МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій

«До захисту допущено»
В.о. завідувача кафедри

_____ Світлана ВАЩЕНКО

____ _____ 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавр

зі спеціальності 122 «Комп'ютерні науки»,

освітньо-професійної програми «Інформаційні технології проектування»

на тему: «Додаток-довідник з цукрового діабету»

Здобувача (ки) групи _ІТ-02_ __Заяц Данило Олександрович_____
(шифр групи) (прізвище, ім'я, по батькові)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ _____Данило Заяц_____
(підпис) (Ім'я та ПРІЗВИЩЕ здобувача)

Керівник ___доц., к.т.н., доц. Світлана ВАЩЕНКО_____
(посада, науковий ступінь, вчене звання, Ім'я та ПРІЗВИЩЕ)

_____
(підпис)

Суми – 2024

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ
В. о. зав. кафедри ІТ

_____Світлана ВАЩЕНКО
«_____» _____ 2024 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА СТУДЕНТУ

Заяц Данилу Олександровичу

1 Тема роботи *Додаток-довідник з цукрового діабету*

керівник роботи *Ващенко Світлана Михайлівна, к.т.н., доцент*,

затверджені наказом по університету від « *07* » травня 2024 р. №0482-VI
2 Строк подання студентом роботи « 26 » травня 2024 р.
3 Вхідні дані до роботи Технічне завдання на розробку додатку-довідника з цукрового діабету

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
Аналіз предметної області, проєктування додатку довідника з цукрового діабету, програмна реалізація додатку-довідника з цукрового діабету

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Актуальність, постановка задачі, порівняння додатків-аналогів, IDEF0 концептуальний рівень, діаграма декомпозиції 1-го рівня, діаграма декомпозиції 2-го рівня, діаграма варіантів використання, логічна модель бази даних, вибір засобів реалізації проєкту, діаграма архітектури додатку, демонстрація результатів роботи, тестування додатку

6. Консультанти розділів роботи:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7.Дата видачі завдання_____10.03.2024_____

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Дослідження предметної області	11.03.2024 – 15.03.2024	
2	Створення дизайну вебдодатку	20.03.2024 – 25.03.2024	
3	Підключення БД до вебдодатку	26.03.2024 – 02.04.2024	
4	Розробка серверної частини вебдодатку	03.04.2024 – 20.05.2024	
5	Тестування вебдодатку	23.05.2024 – 24.05.2024	
6	Запуск вебдодатку	27.05.2024 – 28.05.2024	
7	Відлагодження роботи вебдодатку	29.05.2024 – 31.05.2024	
8	Написання супровідної документації	03.06.2024 – 04.06.2024	
9	Захист проєкту	04.06.2024 – 07.06.2024	

Студент _____ Данило ЗАЯЦ
(підпис)

Керівник роботи _____ к.т.н., доц. Світлана ВАЩЕНКО
(підпис)

АННОТАЦІЯ

Тема кваліфікаційної роботи бакалавра «Розробка додатку-довідника з цукрового діабету».

Пояснювальна записка складається зі вступу, 3 розділів, висновків, списку використаних джерел із 30 найменувань, додатків. Загальний обсяг роботи − 186 сторінок, у тому числі 48 сторінок основного тексту, 4 сторінки списку використаних джерел, 134 сторінок додатків.

Актуальність роботи полягає в поширенні хвороби на цукровий діабет по Україні та світу, постійній потребі хворих в моніторингу показників цукрового діабету. Також актуальність зростає через постійну діджиталізацію суспільства та всіх його сфер по всьому світу.

Мета роботи: створення додатку-довідника з цукрового діабету для інформування пацієнтів та контролю показників по хворобі, з можливістю віддаленої співпраці з лікарем-ендокринологом.

В першому розділі проведено дослідження актуальності проблеми та огляд літературних джерел. Також детально проаналізовано додатки-аналоги та зроблено їх порівняльну характеристику. У результаті було сформовано мету та поставлено задачі на проєкт. За результатами порівняння було сформовано функціональні вимоги. Наприкінці розділу було обрано та обгрунтовано засоби реалізації.

У другому розділі проведено структурно-функціональне моделювання додатку. У результаті створено концептуальні IDEF0 діаграми та їх декомпозицію, логічну модель бази даних, діаграму варіантів використання та архітектури.

У третьому розділі було виконано програмну реалізацію та наведено інструкцію щодо використання розробленого додатку.

Ключові слова: web-додаток, вебдодаток, цукровий діабет, контроль показників, Django.

ЗМІСТ

ВСТУП

Медицина – одна з найважливіших галузей людського суспільства, яка постійно розвивається. Одним із головних рушіїв прогресу є інформаційні технології. Навіть зараз ми можемо переконатися в цьому невпинному прогресі, дивлячись на те, як поєднують людину та технології через вживлення чипів у мозок та оперують за допомогою штучного інтелекту [1]. Але до повноцінною заміни фахівців технологіям ще далеко. Тільки досвідчений лікар-ендокринолог може надати консультацію та слідкувати за здоров'ям хворого, на таку розповсюджену хворобу 21-го століття як цукровий діабет [2]. В час, коли в усіх країнах світу постійно проводять діджиталізацію суспільства [3], ця тема є дуже актуальною.

Об'єктом дослідження є процес комунікації між пацієнтами та лікарями в онлайн форматі.

Предметом дослідження є додаток-довідник для віддаленого скринінгу пацієнтів з цукровим діабетом.

Метою роботи є створення додатку-довідника з цукрового діабету.

Для досягення поставленої мети потрібно вирішити такі задачі:

- провести детальний аналіз додатку-довідника з цукрового діабету та дослідити предметну область догляду за здоров'ям, медицина, цукровий діабет;
- проаналізувати схожі за функціоналом додатки-аналоги для управління цукровим діабетом;
- розробка чітких функціональних вимог та календарного плану проекту;
- здійснення функціонального та структурного моделювання додатку;
- визначення методів та інструментів реалізації розроблюваного додатку;
- концептуальне та логічне моделювання бази даних для зберігання різних даних пацієнтів та лікарів;
- розробка макетів сторінок додатку із зручним та адаптивним сучасним інтерфейсом [4];

- розробка серверної частини додатку-довідника з цукрового діабету;
- тестування додатку-довідника з цукрового діабету.

Додатково передбачити захист чутливої інформації користувачів, як пацієнтів так і лікарів, забезпечення комунікації між пацієнтом і лікарем інструментами web-додатку [5].

Парктична значимість даної розробки полягає втому, що пацієнти дуже легко зможуть знаходити та взаємодіяти зі своїми лікарями, отримувати корисну інформацію по своїй хворобі, так і лікарі в віддаленому форматі можуть переглядати показники та записи своїх пацієнтів з цукровим діабетом, та за потреби відповідати на цікавлячи їх запитання та надавати рекомендації.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд літературних джерел

Опираючись на дослідження Міністерства охорони здоров'я України та Всесвітньої організації охорони здоров'я, частка людей, яка хворіє на діабет першого або другого типів, по Україні та по всьому світу зросла вдвоє [6]. Також варто зазначити, що цукровий діабет почав з'являтися і в людей віком до 20 років [7].

Якщо брати до уваги всі ці показники та невпинну цифровізацію суспільства, то актуальність додатку-довідника для людей, які хворіють на цукровий діабет, важко переоцінити.

Також варто зазначити, що через війну багато людей виїхали з України, серед яких багато хворих на цукровий діабет. Проте, якщо вірити статті The New York Times, українці не задоволені якістю та бюрократичністю медицини в європейських країнах. Це також дуже впливає на актуальність додатку-довідника з цукрового діабету, адже за допомогою нього людина може поспілкуватися та отримати консультацію з лікарем-ендокринологом, до якого вона була прив'язана до виїзду [8].

При огляді різних літературних джерел було проаналізовано багато інформації по рішенню проблем, схожих на задачу розроблюваного додатку та використанню відповідних технологій.

Перш за все – це веб-додаток, отже однією з важливіших проблем буде розробка зручного та інтуїтивного інтерфейсу користувача. Особливу увагу слід приділити семантиці та семантичним тегам в мові розмітки HTML5. Вони не тільки допомогають пошуковим роботам, але і покращують сприйняття сайту за допомогою допоміжних приладів людям з вадами, таким як скрінрідери, тощо. За допомогою стилів CSS та JavaScript буде налаштовуватись зовнішній вигляд та інтерактивність додатку [9, 10].

Особливу увагу варто приділити створенню та проектуванню бази даних для подібного проєкту. Створення бази даних - це окрема наука, яка потребує багато знань. Для розроблюваного додатку-довідника краще за все підійде звичайна реляційна база даних PostgreSQL [11].

Розробка серверної частини додатку буде виконуватися на фреймворці мови програмування Python Django [12]. Цей фреймворк має зручну архітектуру та багато важливих речей, які прискорюють розробку серверної частини додатку, такі як: організація сесій, налаштований захист від багатьох розповсюджених загроз, гнучне налаштування проєкту та розділення його функціоналу по окремим додаткам. Також варто відмітити, що Python – дуже універсальна мова програмування завдяки своїм бібліотекам, які можна застосовувати для розробки додаткового функціоналу чи тестування [13]. Робота зі створеною базою даних додатку буде виконуватися за допомогою Django ORM та його моделей [14].

1.2 Аналіз програмних продуктів – аналогів

Серед інших web-додатків не було знайденого схожого за темою. Розглянемо мобільні застосунки зі схожим функціоналом.

GL index: додаток від Reflective Technologies [15], який може бути корисним людям, хворим на цукровий діабет. Він має зручний інтерфейс та багато різних калькуляторів, наприклад, знаходження індексу маси тіла, кровяного тиску, ваги тощо. Також за допомогою GL index можна розхраховувати глікемічний індекс різної їжі. В додатку є дві версії: платна та безкоштовна. В платній версії додатку відкривається значна частина функціоналу та знімається обмеження на кількості вимірів. Інтерфейс додатку продемонстрований на рисунку 1.1.

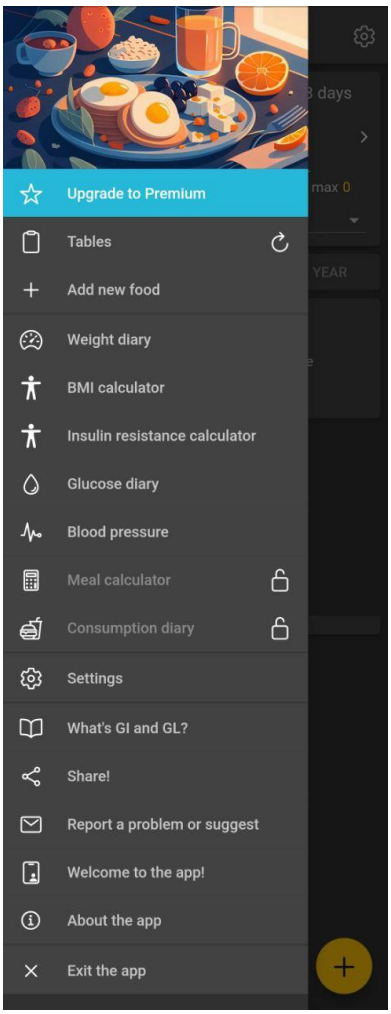


Рисунок 1.1 – Інтерфейс меню додатку GL index

Diabetes M: мобільний додаток для хворих на цукровий діабет з великою кількістю корисного функціоналу. Він включає в себе різноманітні записи: записи їжі, заміри цукру, записи про фізичну активність [16]. Додатково надається багато різних графіків та іншої візуалізації для більшої інформативності, та налаштування різних нагадувань. В Diabetes:M також є платна версія, яка надає додатковий функціонал, наприклад, прив’язування лікаря для віддаленого моніторингу показників пацієнта. Інтерфейс додатку показаний на рисунку 1.2.

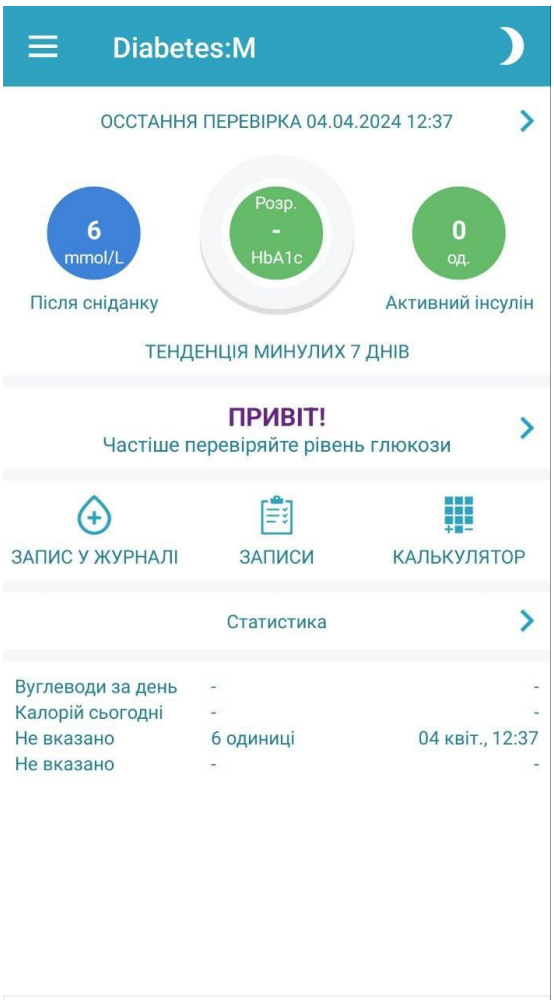


Рисунок 1.2 – Головна сторінка додатку Diabetes:M

Діабет: мобільний додаток схожий за функціоналом на попередні [17]. В ньому також присутньо багато калькуляторів, які можуть бути корисними людям хворим на цукровий діабет. В додатку також є платна версія, яка відкриває додатковий функціонал. Однією з переваг додатку є можливість підключення глюкометру по серійному номеру для моніторингу показників глюкози в крові автоматично. Головне меню додатку продемонстровано на рисунку 1.3.

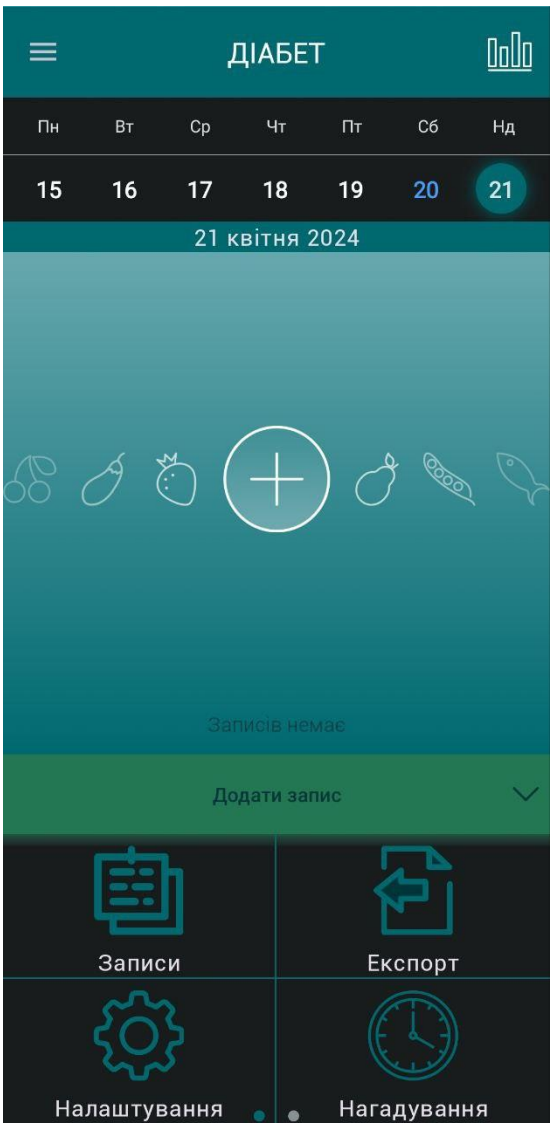


Рисунок 1.3 – Головне меню додатку Діабет

Після аналізу існуючих програмних продуктів аналогів можна побачити, що в кожному з них майже однаковий функціонал, але є і деякі відмінності. Основною є комунікація з лікарем, яка є тільки в додатку Diabetes:M, але тільки на платній основі. В розглянутих додатках багато спільного функціоналу: створення записів по показникам пацієнта, формування звітів та перегляд статистики по записам.

Основною відмінністю розроблюваного додатку-довідника для пацієнтів з цукровим діабетом буде те, що на відміну від простого ведення записів, буде реалізовано блог з корисною інформацію, прив’язку до лікаря на безоплатній основі та комунікація з ним інструментами додатку. Також буде реалізований спільний функціонал, такий як ведення записів пацієнтами та формування звітів.

Узагальнене порівняння додатків-аналогів показано в таблиці 1.1.

Таблиця 1.1 - Порівняння показників програмного забезпечення

Показник	GL index	Diabetes:M	Діабет	Розроблюваний додаток
Інтуітивно зрозумілий інтерфейс	+	+	+	+
Контактна інформація	+	+	+	+
Корисна загальна інформація про діабет	-	-	-	+
Профіль користувачів	-	+	-	+
Прив’язка до лікаря та комунікація з ним	-	+	-	+
Пошук лікарів	-	-	-	+
Ведення записів по хворобі	+	+	+	+
Формування звітів	-	+	+	+
Вартість	Частково безкоштовно	Частково безкоштовно	Частково безкоштовно	Безкоштовно

1.3 Мета та задачі дослідження

Метою даного проекту є створення додатку-довідника для пацієнтів з цукровим діабетом, для отримання корисної інформації по хворобі, ведення записів в картці та комунікації з прив'язаним лікарем-ендокринологом.

Основні задачі по реалізації кваліфікаційної роботи є наступними:

- розробка технічного завдання та визначення функціональних вимог;
- провести функціональне та концептуальне моделювання додатку;
- розробка та проектування бази даних для збереження інформації;
- виконати реалізацію інтерфейсу додатку;
- розробка архітектури та структури розроблюваного додатку;
- виконати розробку серверної частини додатку;
- провести тестування додатку-довідника в можливих "вузьких" місцях.

Вимоги до проекту в цілому, структури web-додатку та функціонування системи описані у технічному завданні на розробку проєкту (додаток А).

Для досгненя всих поставлнених задач та успішного виконання проєкту було провдено планування робіт з аналізом можливих ризиків(додаток Б).

1.4 Методи дослідження та інструменти реалізації

Для реалізації додатку-довідника з цукрового діабету будуть використовуватися різноманітні web-технології.

Клієнтська частина додатку буде розроблюватися за допомогою таких технологій: мова гіпертекстової розмітки HTML5 для формування каркасу сайту та семантично правильної структури, таблиця каскадних стилів CSS сумісно з бібліотекою Bootstrap для визначення стилів сторінки та візуального їх покращення[18]. Для інтерактивності додатку буде використовуватися кросс-

браузерна мова програмування JavaScript. Ці три технології є основою будь-якого web-додатку.

Для серверної частини додатку буде використовуватися мова програмування Python з його web-фреймворком Django, а для формування API [19] буде використовуватися Django REST Framework [20]. Вибір впав на мову програмування через її універсальність та гарну структуру та влаштовані механізми захисту від різного типу атак фреймворку Django.

Для зберігання даних додатку буде використовуватися реляційна база даних PostgreSQL. Запити до неї будуть відбуватися через моделі Django ORM. Зазначений підхід майже нівелює такий вид кіберзагрози як ін’єкція SQL.

2 ПРОЄКТУВАННЯ ДОДАТКУ-ДОВІДНИКА З ЦУКРОВОГО ДІАБЕТУ

2.1 Функціональне моделювання додатку в IDEF0

Функціональна модель у нотації IDEF0 дозволяє на концептуальному рівні визначити основну функцію додатку, механізми реалізації та певні обмеження, які виникнуть в цьому бізнес-процесі та те, що отримується на виході. Діаграма IDEF0 є основою в функціональному аналізі предметної області, яка допомагає краще розуміти її та приймати відповідні рішення при розробці [20]. На рисунку 2.1 продемонстрована діаграма функціональної моделі IDEF0 для основного бізнес-процесу додатку довідника з цукрового діабету, а саме підтримка інформування та контролю показників хворих на цукровий діабет.

Рисунок 2.1 – Діаграма IDEF0 на концептуальному рівні

Вхідні дані:

- медичні показники пацієнта (глюкоза в крові, хлібна одиниця по спожитим вуглеводам, доза інсуліну в ОД;

- дані про пацієнта (дані про конкретного пацієнта зареєстрованого у додатку).

Обмеження:

- політика конфеденційності (політика додатку, яка інформує користувача як та ким будуть використовувати його персональні дані, захист персональних даних);
- правила користування додатком (загальні правила користування додатком);
- довідкова інформація про хворобу (корисна інформація для формування функціоналу з інформування користувачів по хворобі цукрового діабету);
- методика розрахунків медичних показників (розрахунок приблизної дози інсуліну по хлібній одиниці (ХО), де 12г вуглеводів – це 1 ХО, а 1 хлібна одиниця = 1 одиниці інсуліну).

Механізми:

- користувач (основний механізм додатку, без якого неможливі будь-які процеси);
- додаток DiaScreen(серверна частина додатку Django);
- прикладне програмне забезпечення (база даних для збереження та подальшої обробки даних в додатку, браузер для користування додатком).

Вихідні дані:

- оновлені дані в базі даних про показники користувача;
- дані по показникам цукрового діабету, які показуються користувачу в зручному форматі;
- корисна інформація по захворюванню.

Для отримання більш точного розуміння основного бізнес-процесу в додатку-довіднику для хворих на цукровий діабет потрібно розробити діаграму декомпозиції на основі функціональній моделі IDEF0. Для цього потрібно розбити її на менші взаємопов'язані кроки. Входи, виходи, обмеження та механізми залишаються такими ж, але надається більша деталізація.

Неавторизовані користувачі можуть переглядати різну довідкову інформацію про цукровий діабет. Після реєстрації та авторизації пацієнт може перейти в картку та прив'язати лікаря для співпраці. При переході в картку отримує список записів для різних показників, обирає потрібний та заповнює відповідну форму. Дані оброблюються на відображуються в картці в зручному вигляді. Діаграму декомпозиції продемонстровано на рисунку 2.2.

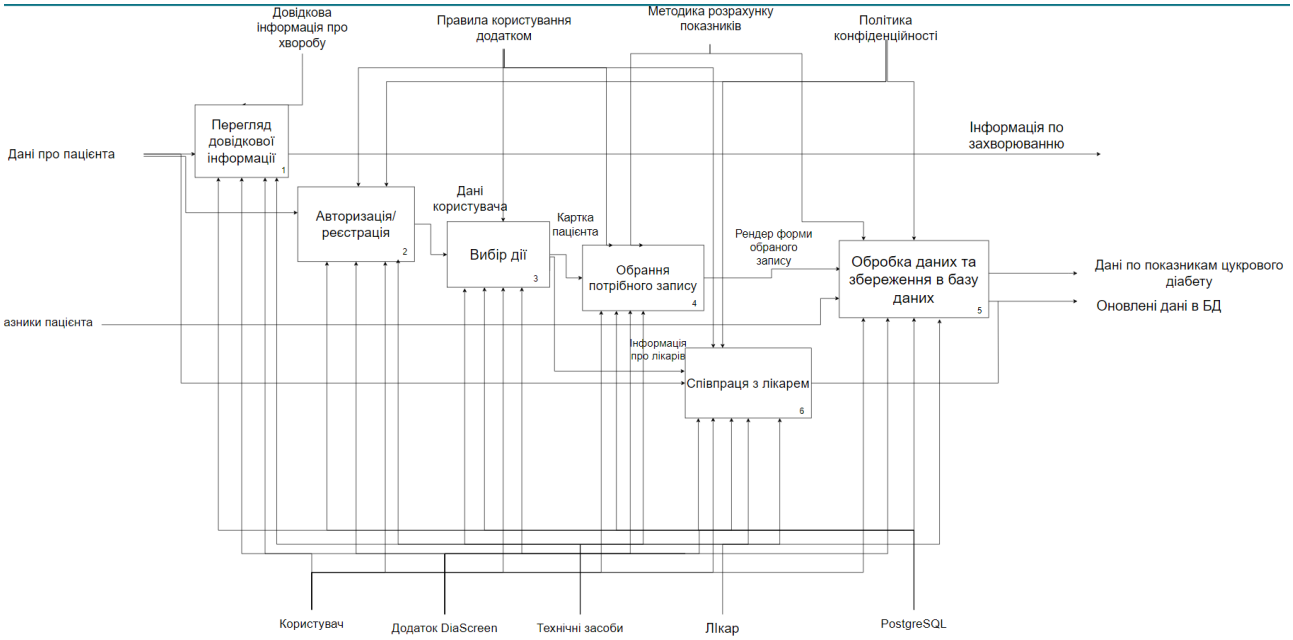


Рисунок 2.2 – Діаграма декомпозиції функціональної моделі IDEF0

Додатково створено діаграму декомпозиції для процесу прив’язки лікаря до пацієнта. В якості входів виступають дані пацієнта та потреба в прив’язці лікаря. Механізми обмеження: правила користування додатком та політика конфіденційності. Механізми виконання не змінені з попередньої моделі. Авторизований пацієнт переходить до профілю лікаря, копіює код прив’язки в буфер обміну, після чого повертається в свій профіль та заповнює форму прив’язки лікаря. На виході отримуємо оновлені дані в БД про прив’язаного лікаря до пацієнта. Діаграма декомпозиції процесу прив’язки лікаря продемонстровано на рисунку 2.3.

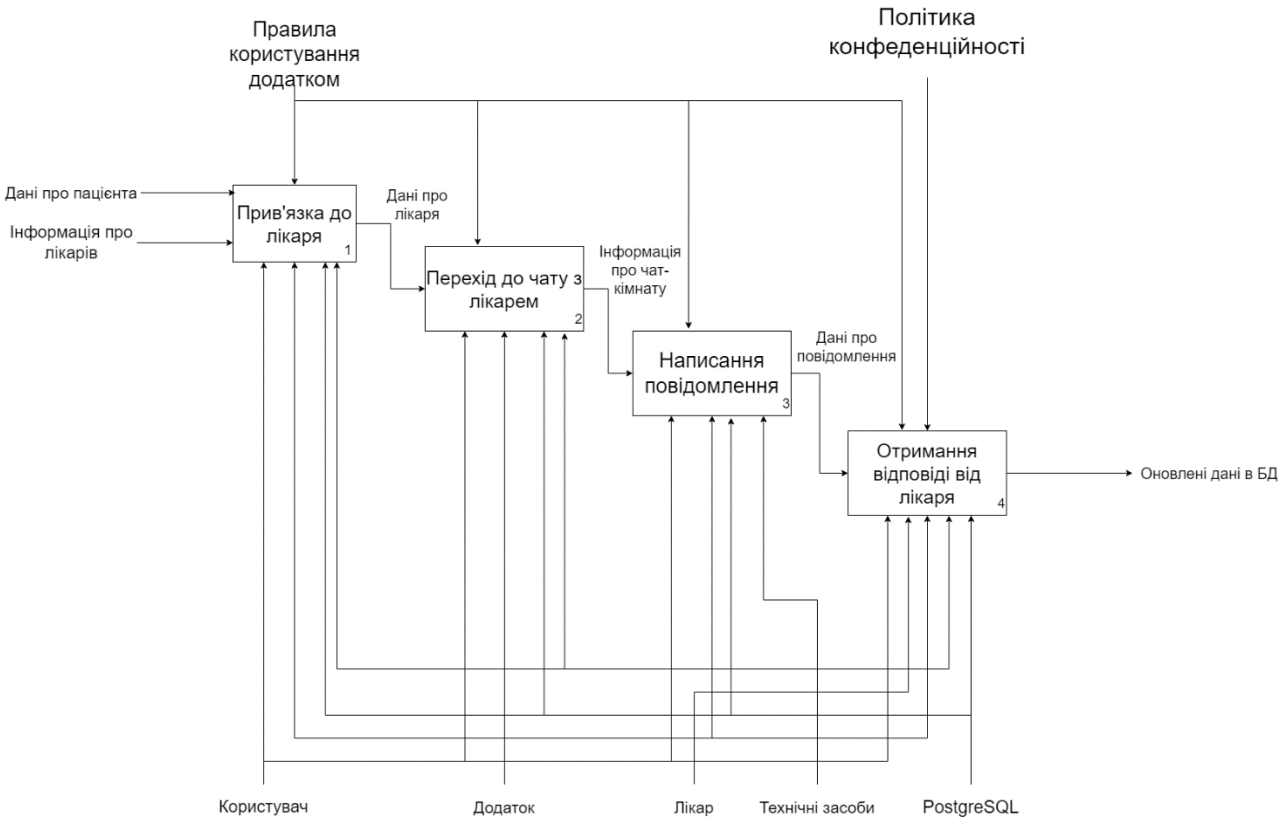


Рисунок 2.3 – Діаграма декомпозиції процесу співпраці з лікарем

Також необхідно декомпозувати блок з прив’язкою до лікаря, бо це складний процес, який складається з декількох кроків. В ньому приймає участь тільки користувач, додаток, технічні засоби та база даних. Декомпозицію блоку прив’язки до лікаря показано на рисунку 2.4.

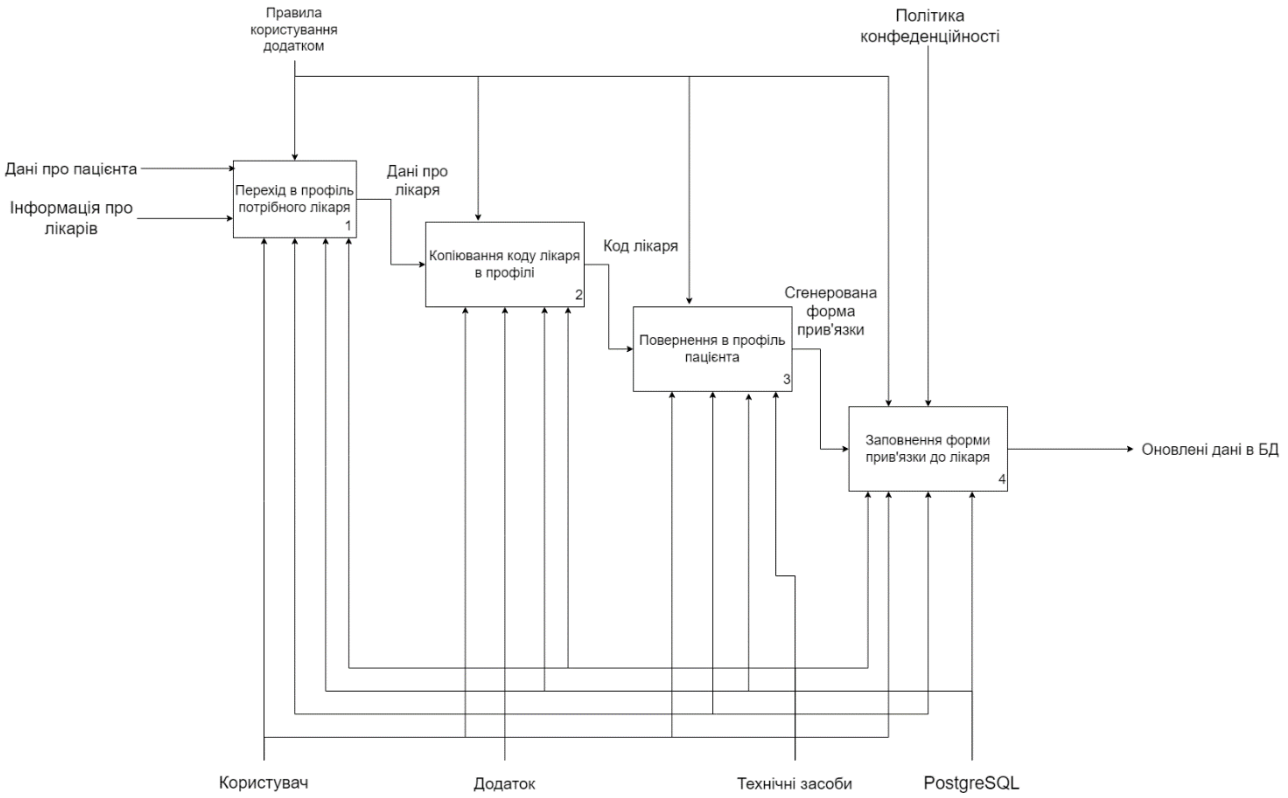


Рисунок 2.4 – Діаграма декомпозиції процесу прив’язки до лікаря

2.2 Проєктування діаграми варіантів використання

Для виділення основних можливостей системи та її функціоналу, а також розподіл його використання по певним акторам, необхідно побудувати діаграму варіантів використання додатку (use-case diagram). Варіанти використання в цій UML діаграмі визначають очікувану поведінку додатку, а не їх власну реалізацію. Ключова концепція діаграми варіантів використання полягає в тому, що ми проєктуємо систему з точки зору користувача [21]. На рисунку 2.5 продемонстровано use-case діаграма для додатку-довідника з цукрового діабету.

Рисунок 2.5 – Діаграма варіантів використання додатку DiaScreen

На діаграмі продемонстровано 5 акторів:

- Гість – неавторизований або незареєстрований користувач;

- Пацієнт – користувач, який зареєстрував аккаунт пацієнта та має відповідну роль в додатку;
- Лікар – користувач, який зареєстрував аккаунт лікаря та має відповідну роль в додатку;
- База даних – відповідає за зберігання усих даних додатку та приймає участь у всіх процесах;
- Адміністратор – користувач з ролями адміністратора та відповідними правами доступа для редагування інформації в додатку, зокрема додавання та редагування статей з корисною інформацією.

2.3 Проєктування діаграми компонентів додатку

При проєктуванні веб-орієнтовного додатку, важливим етапом є побудова UML діаграми компонентів. Вона потрібна для огляду системи в цілому, деталі її організації та взаємозв’язки між компонентами системи. На рисунку 2.6 представлена діаграма компонентів додатку-довідника з цукрового діабету. На ній зображені основні компоненти системи, такі як додатки, конфігураційні компоненти проєкту та база даних. Інші компоненти є модульними та можуть бути замінени за потреби [22].

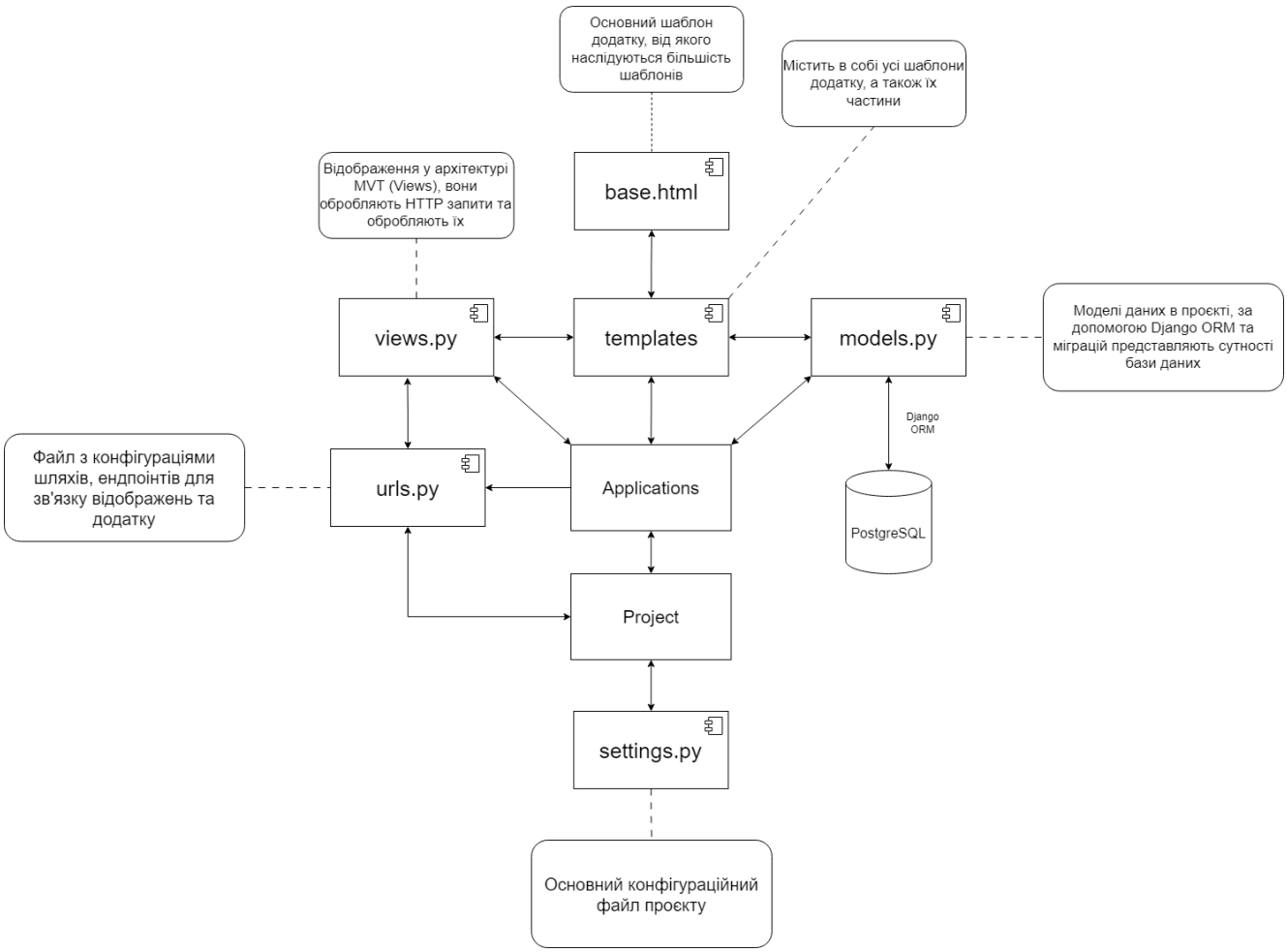


Рисунок 2.6 – Діаграма компонентів додатку-довідника з цукрового діабету

2.4 Проєктування бази даних

Проєктування бази даних є одним з найважливіших етапів при проєктуванні всього додатку, особливо який використовує різні дані для своєї роботи. Створена та нормалізована логічна модель бази даних допомагає розробити оптимізовану систему з використанням різноманітних даних [23]. Логічна модель бази даних представляє з себе набір реляційних таблиць – сутностей, кожна з яких пов’язана між собою певними зв’язками, наприклад, один до одного або один до багатьох. Також визначаються типи даних, що будуть зберігатися, а також первісні або логічні ключі.

У процесі створення логічної моделі бази даних були визначені такі сутності:

- Пацієнти (Patients) – користувачі, які мають роль пацієнта;
- Лікарі (Doctors) – користувач, який зареєстрував аккаунт пацієнта та має відповідну роль в додатку;
- Адреса пацієнта (PatientAdress) – адреса пацієнта;
- Організація (Organization) – зберігання даних про наявну організацію у лікаря;
- Адреса організації (OrgAdress) – адреса організації;
- Відгуки (Feedback) – відгуки пацієнтів про лікаря, один пацієнт може написати тільки один відгун для одного конкретного лікаря;
- Повідомлення чату (ChatMessage) – повідомлення, які надходять від пацієнта до лікаря або навпаки;
- Стаття (Article) – сутність статті з корисною інформаціє про хворобу для інформаційної панелі додатку;
- Заміри глюкози (GlucoseMeasurement) – зберігання даних по замірам глюкози в крові пацієнта;
- Категорія заміру (GlucoseCategoryMeasurement) – категорія запису глюкози;
- Фізична активність (PhysicalActivityMeasurement) – зберігання даних по замірам фізичної активності;
- Тип фізичної активності (PhysicalCategoryMeasurement) – вид фізичної активності;
- Записи по їжі (FoodMeasurement) – зберігання даних по споживанню продуктів харчування та кількості вуглеводів з розрахунком приблизної дози інсуліну;
- Час заміру споживання (FoodCategoryMeasurement) – час заміру їжі;
- Список їжі (FoodItems) – єднальна сутність для записів та страви, представляє собою список усіх спожитих страв;
- Страва (FoodItem) – зберігання даних по спожитим стравам;

- Записи по ін’єкціям інсуліну (InsulineMeasurement) – зберігання даних по записам інсуліну та його дози;
- Тип ін’єкції по часу (InsulineCategoryMeasurement) – категорія по часу ін’єкції;
- Аналізи (Analysis) – зберігання файлів з певними аналізами пацієнтів;
- Тип аналізу (AnalysisType) – типи аналізів.

На рисунку 2.7 продемонстровано фізичну модель бази даних у контексті тієї СУБД, яку було обрано для виконання практичної реалізації проєкту.

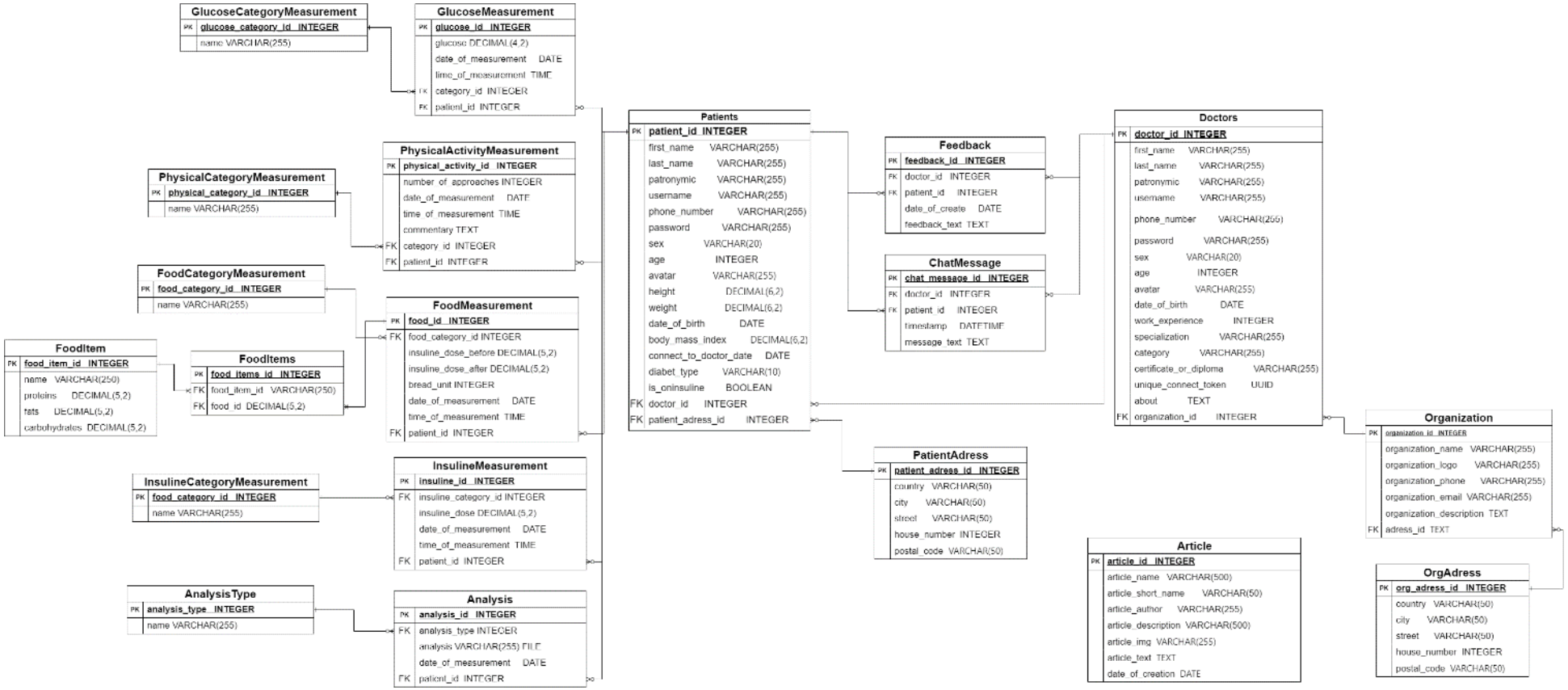


Рисунок 2.7 – Фізична модель база даних

2.5 Архітектура додатку

Серверна частина розробляється на Python Django. Архітектура Django представляє собою патерн MVT (Model-View-Template). Користувач взаємодіє з шаблонами, дані з шаблону передаються на сервер через шляхи маршрутизації (Routing), після чого оброблюються в відображеннях (View). Самі відображення оброблюють дані через цикл запитів та відповідей HTTP (request-response cycle) та взаємодіють з моделями даних (Models), які представляють собою певні сутності в додатку, які зберігаються в базі даних за допомогою ORM [24]. Після обробки дані через контекст відправляються назад в шаблон. Діаграму архітектури додатку продемонстровано на рисунку 2.8.

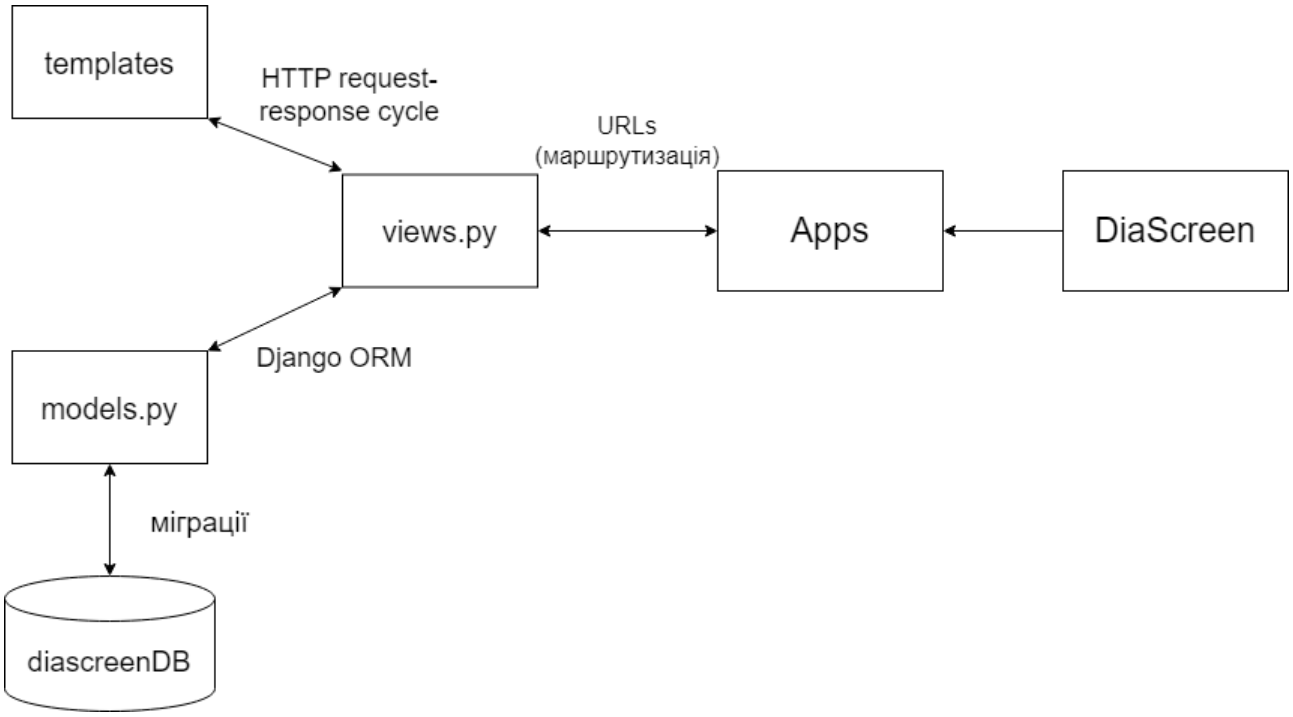


Рисунок 2.8 – Архітектура MVT додатку-довідника з цукрового діабету DiaScreen

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ДОДАТКУ-ДОВІДНИКА З ЦУКРОВОГО ДІАБЕТУ

3.1 Програмна реалізація

Для початку роботи над додатком було створено репозиторій Git для контролю версій. Це дуже гарна практика для розробки проєкту будь-якої складності, яка дозволяє контролювати зміни та зберігати їх у віддаленому репозиторії, до якого можна отримати доступ з будь-якого пристрою [25]. Створений репозиторій проєкту продемонстровано на рисунку 3.1.

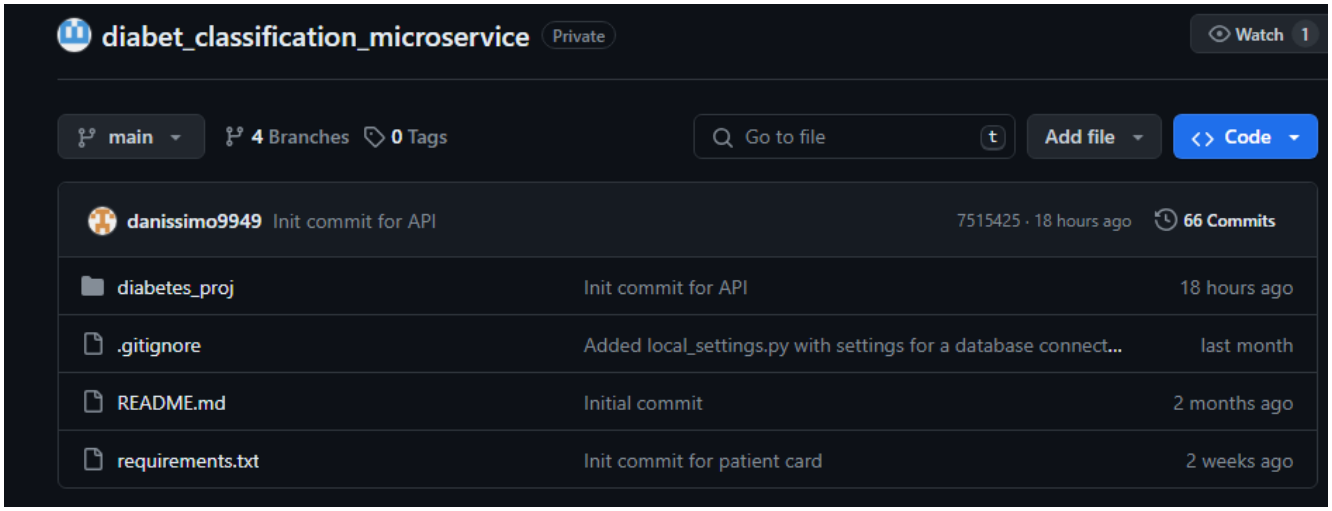


Рисунок 3.1 – Віддалений репозиторій додатку-довідника з цукрового діабету

Наступним етапом йшло створення структури проєкту Django та налаштування віртуального середовища для встановлення усіх необхідних залежностей. Проєкт містить багато додатків, які потрібні щоб розділити реалізацію функцій, для кращої підтримки. Структура додатку Django показано на рисунку 3.2.

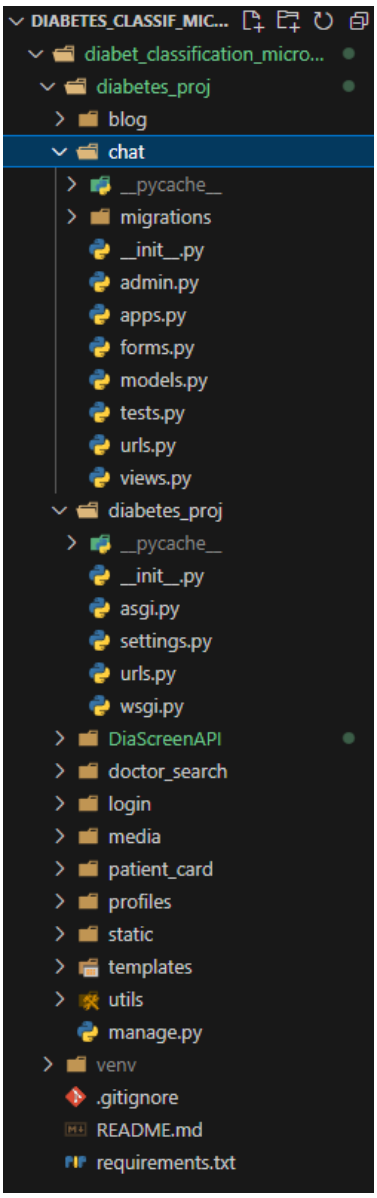


Рисунок 3.2 – Структура проєкту Django

Клієнтська частина: клієнтська частина розроблювалась на HTML, CSS та JS з використанням бібліотеки Bootstrap для швидкої стилізації. Розробка клієнтьскої частини велась одночасно з розробкою серверної частини додатку. Для неї створювались шаблони (Templates), наприклад, базовий шаблон base.html, від якого наслідувалась основна частина інших шаблонів за допомогою шаблонної мови Django Template Language [26]. Його продемонстровано на рисунку 3.3.

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>DiaScreen - онлайн скринінг</title>
    <link rel="stylesheet" href="{% static 'css/reset.css' %}">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMh
    <link rel="stylesheet" href="{% static 'css/index.css' %}">
    <link rel="stylesheet" href="{% static 'css/media.css' %}">
    {% block styles %}
    {% endblock styles %}
</head>
<body>
    {% include 'partials/_header.html' %}
    <main>
        {% block content %}
        {% endblock content %}
    </main>
    {% include 'partials/_footer.html' %}

    {% block scripts %}
    {% endblock scripts %}
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeH
    <script src="https://code.jquery.com/jquery-3.5.1.js " integrity="sha256-QWo7LDvxbWT2tbbQ97B53yJnYU3WhH/C8ycbRAkjPDc=" crossorigin=" anonymous "></script>
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js " integrity="sha384-9/reFTGAW83EW2RDu2S0VKaIzap3H66lZH81PoYlFhbGU+6BZp6G7niu735Sk7lN " cros
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js " integrity="sha384-B4gt1jrGC7Jh4AgTPSdUtOBvfO8shuf57BaghqFfPlYxofvL8/KUEfYiJOMMV+rV " cr
</body>
</html>
```

Рисунок 3.3 – Основний шаблон base.html

Наслідування шаблонів використовують зазначені блоки в базовому шаблоні та просто їх перевизначають за потреби. Наприклад, на рисунку 3.4 продемонстровано наслідування шаблону пошуку лікарів.

```
{% extends 'base.html' %}
{% load static %}

{% block styles %}
<link rel="stylesheet" href="{% static 'css/search.css' %}">
{% endblock styles %}

{% block content %}
<section id="section"  class="my-3">
    <div class="container">
        <div class="row">
            <div class="col-md-3">
              <aside>
                  <form method="get" class="search__form">
                    {{ search_form.as_p }}
                  <button type="submit" class="btn btn-primary submit_filter_btn">Пошук</button>
                  </form>
                  <h2 class="fs-4 filter__title">Розширені фільтри пошуку</h2>
                  <form method="get" class="filter__form" novalidate>
                     {{ filter_form.as_p }}
                     <button type="submit" class="btn btn-primary submit_filter_btn">Примінити фільтри</button>
                  </form>
              </aside>
            </div>
            <div class="col-md-9">
              <div class="row" id="doctor_row">
                  {% for doctor in object_list %}
                  <div class="card mx-4 mb-3" style="max-width: 540px;">
                    <a href="{% url 'doctor_profile' pk=doctor.pk %}" class="doctor-link"><div class="row g-0">
                      <div class="col-md-4">
                        <img src="{{ doctor.avatar.url }}" class="img-fluid" alt="аватар доктора">
                      </div>
                      <div class="col-md-8">
                        <div class="card-body">
                          <h5 class="card-title">{{ doctor.get_full_name }}</h5>
                          <p class="card-text">{{ doctor.specialization }}</p>
                          <p class="card-text"><small class="text-body-secondary">{{ doctor.age }}</small></p>
```

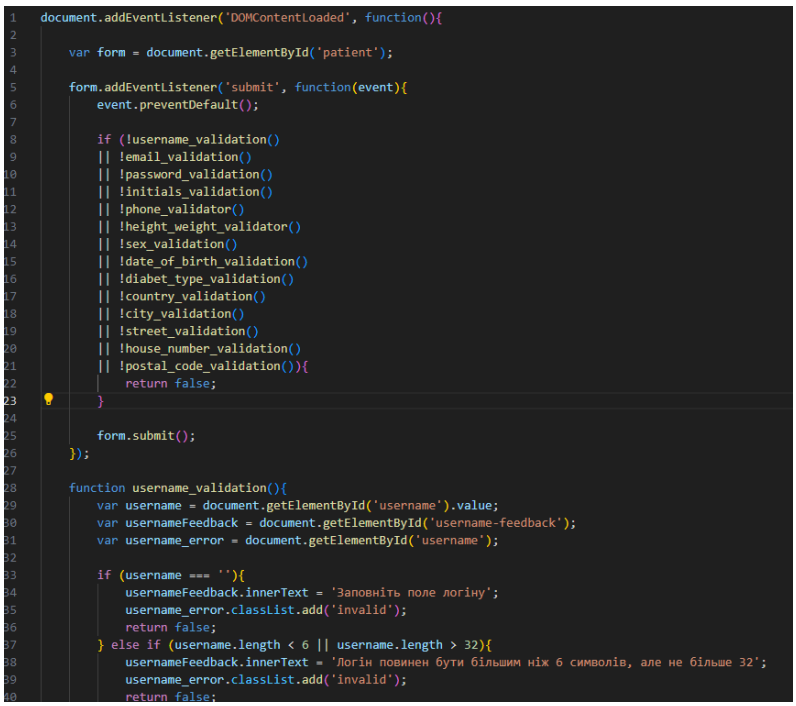
Рисунок 3.4 – Шаблон пошуку лікарів

Стилізація розмітки здебільшого відбувалась за допомогою сіток Bootstrap та інших стілів. Проте також використовувалися каскадні таблиці стилів CSS. На рисунку 3.5 показана стилізація сторінки картки пацієнта, зокрема записів по категоріям.

```css
.meas_glucose__list{
    min-width: 350px;
    max-height: 300px;
    overflow: hidden;
    overflow-y: auto;
    flex-wrap: wrap;
}

.meas_fit__list{
    min-width: 600px;
    max-height: 300px;
    overflow: hidden;
    overflow-y: auto;
    flex-wrap: wrap;
}

.meas__block{
    background-color: #4169E1;
}

.meas__block, .category-block, .time-block, .glucose-block, .bad{
    display: flex;
    justify-content: center;
    align-items: center;
    height: 40px;
    width: 100%;
    border-radius: 5px;
    padding: 5px 10px;
    margin-right: 10px;
    color: #FFF;
}

.category-block {
    background-color: #4169E1;
    margin-left: 10px;
}
```

Рисунок 3.5 – Стилізація картки пацієнта

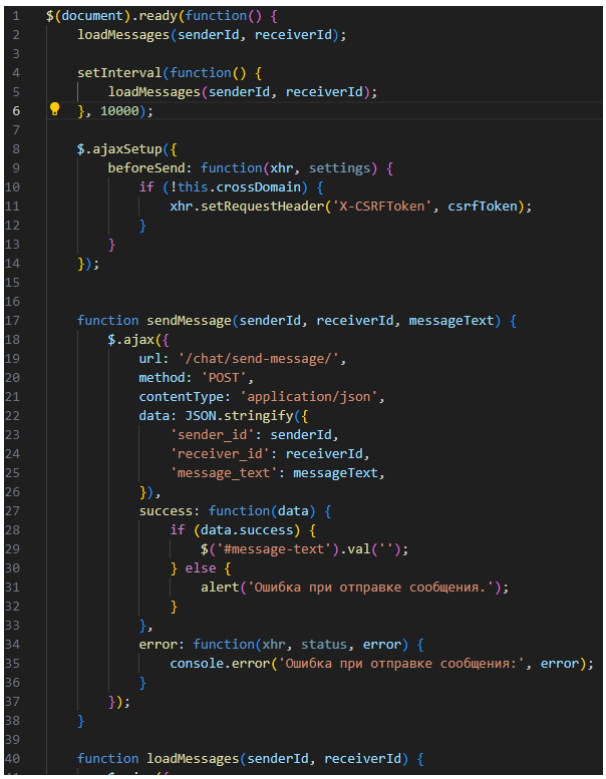
Для реалізації валідації форм реєстрації, розробки чату та іншої інтерактивності використовувалась мова програмування JavaScript. Приклад валідації форми показано на рисунку 3.6.



```
document.addEventListener('DOMContentLoaded', function(){

    var form = document.getElementById('patient');

    form.addEventListener('submit', function(event){
        event.preventDefault();

        if (!username_validation()
        || !email_validation()
        || !password_validation()
        || !initials_validation()
        || !phone_validator()
        || !height_weight_validator()
        || !sex_validation()
        || !date_of_birth_validation()
        || !diabet_type_validation()
        || !country_validation()
        || !city_validation()
        || !street_validation()
        || !house_number_validation()
        || !postal_code_validation()){
            return false;
        }

        form.submit();
    });

    function username_validation(){
        var username = document.getElementById('username').value;
        var usernameFeedback = document.getElementById('username-feedback');
        var username_error = document.getElementById('username');

        if (username === ''){
            usernameFeedback.innerText = 'Заповніть поле логіну';
            username_error.classList.add('invalid');
            return false;
        } else if (username.length < 6 || username.length > 32){
            usernameFeedback.innerText = 'Логін повинен бути більшим ніж 6 символів, але не більше 32';
            username_error.classList.add('invalid');
            return false;
```

Рисунок 3.6 – Валідація форми пацієнта

За аналогією було провалідовано форму реєстрації доктора. Для реалізації часту використовувались AJAX запити, які взаємодіють з серверною частиною та вигружають на сторінку повідомлення без її перезавантаження для імітації чату в реальному часі [27]. Реалізація чату продемонстрована на рисунку 3.7.



```
$(document).ready(function() {
    loadMessages(senderId, receiverId);

    setInterval(function() {
        loadMessages(senderId, receiverId);
    }, 10000);

    $.ajaxSetup({
        beforeSend: function(xhr, settings) {
            if (!this.crossDomain) {
                xhr.setRequestHeader('X-CSRFToken', csrfToken);
            }
        }
    });


    function sendMessage(senderId, receiverId, messageText) {
        $.ajax({
            url: '/chat/send-message/',
            method: 'POST',
            contentType: 'application/json',
            data: JSON.stringify({
                'sender_id': senderId,
                'receiver_id': receiverId,
                'message_text': messageText,
            }),
            success: function(data) {
                if (data.success) {
                    $('#message-text').val('');
                } else {
                    alert('Ошибка при отправке сообщения.');
                }
            },
            error: function(xhr, status, error) {
                console.error('Ошибка при отправке сообщения:', error);
            }
        });
    }

    function loadMessages(senderId, receiverId) {
```

Рисунок 3.7 – Реалізація чату за допомогою AJAX запитів

Серверна частина: після створення проєкту Django та налаштування основних моментів було створено декілька додатків для розділення по ним функціоналу та налаштовано маршрутизацію по ним, як показано на рисунку 3.8.

```
from django.contrib import admin
from django.views.generic import TemplateView
from django.conf import settings
from django.conf.urls.static import static
from django.urls import path, include
from profiles.views import send_support_ticket

urlpatterns = [
    path('', TemplateView.as_view(template_name = 'index.html'), name='home'),
    path('admin/', admin.site.urls),
    path('support/', send_support_ticket, name='support'),
    path('accounts/', include('login.urls')),
    path('doctor-search/', include('doctor_search.urls')),
    path('profile/', include('profiles.urls')),
    path('blog/', include('blog.urls')),
    path('card/', include('patient_card.urls')),
    path('chat/', include('chat.urls')),
    path('api/', include('DiaScreenAPI.urls')),
    path("__debug__/", include("debug_toolbar.urls")),
] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)

```

Рисунок 3.8 – Налаштування маршрутизації проєкту

Після попереднього налаштування треба почати створювати базу даних за допомогою Django ORM (Object Relational Model). Цей механізм дозволяє створювати звичайні класи сутностей мовою програмування та за допомогою міграцій перетворювати їх на звичайні реляційні таблиці в базі даних [28]. Використання ORM є гарною практикою та покриває одну з можливих загроз – SQL ін'єкції, так як чистих SQL запитів не використовується, а вся робота проходить виключно з об'єктами. Процес створення моделей показано на рисунку 3.9.

```python
class Patient(DiaScreenUser):
    """
    Choices for diabet type options
    """
    FIRST_TYPE = '1'
    SECOND_TYPE = '2'
    NULL_TYPE = 'null'
    DIABETES_TYPE_CHOICES = (
        (FIRST_TYPE, '1 тип'),
        (SECOND_TYPE, '2 тип'),
        (NULL_TYPE, 'Відсутній'),
    )

    height = models.DecimalField(max_digits=6, decimal_places=2)
    weight = models.DecimalField(max_digits=6, decimal_places=2)
    body_mass_index = models.DecimalField(max_digits=6, decimal_places=2, blank=True, null=True)
    connect_to_doctor_date = models.DateTimeField(blank=True, null=True)
    diabet_type = models.CharField(max_length=10, choices=DIABETES_TYPE_CHOICES, default=NULL_TYPE)
    is_oninsuline = models.BooleanField(db_index=True, null=True)
    doctor_id = models.ForeignKey(Doctor, on_delete=models.SET_NULL, blank=True, null=True)
    adress_id = models.ForeignKey(Adress, on_delete=models.SET_NULL, blank=True, db_index=True, null=True)

    def calculate_BMI(self):
        """
        Calculate body mass index for patient from his measurements
        """
        return self.weight / (self.height ** 2)


    def save(self, *args, **kwargs):
        """
        Overriding abstract user metod from django.contrib.auth.models which add additional
        functional when this model is save to database
        """
        if self.weight and self.height:
            self.body_mass_index = self.calculate_BMI()
```

Рисунок 3.9 – Створення моделі пацієнта

Усі поля сутності виступають атрибутами класу та прив’язані до певних полів, які визначають метод зберігання даних в базі даних, а також певні обмеження. Усі інші моделі створювались за аналогією.

Після створення моделей було виконано міграції для формування таблиць в базі даних PostgreSQL, які відповідають даним моделям. Усі таблиці бази даних, створені таким чином, показані на рисунку 3.10.

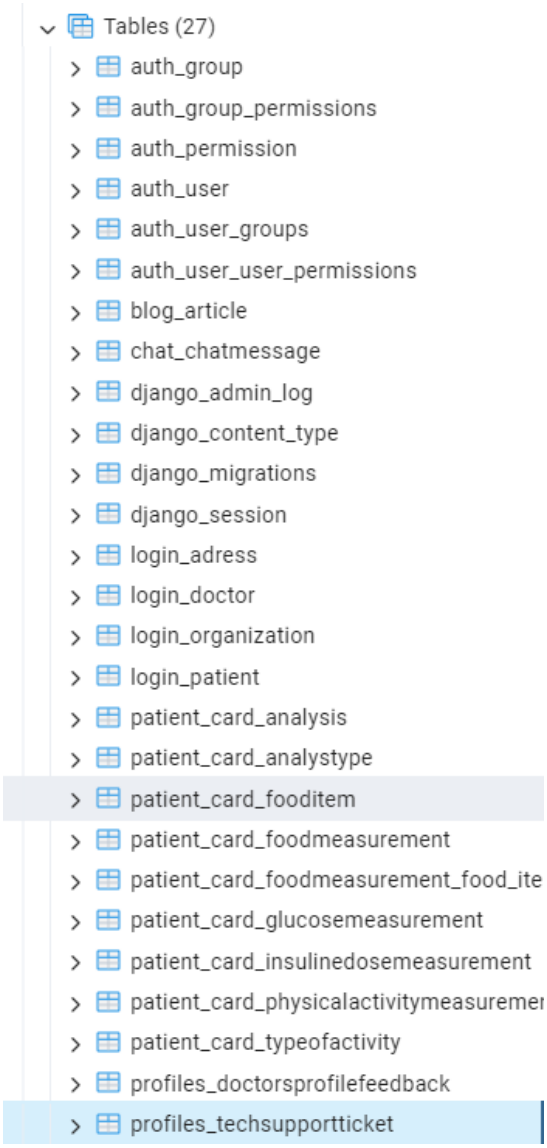


Рисунок 3.10 – Список таблиць бази даних проєкту

Для створення або використання даних з бази даних використовується шар відображень (View). Вони слугують для зв’язку моделей та шаблонів та можуть отримувати та віддавати дані через HTTP запити. До прикладу візьмемо сторінку

з корисною інформацією по хворобі цукрового діабету. За нього відповідає відображення продемонстроване на рисунку 3.11.

```
class InformationPanel(generic.ListView, AdminRoleMixin):
    model = Article
    template_name = 'info_panel.html'
    context_object_name = 'article_list'
    paginate_by = 9

    def get_context_data(self, **kwargs: Any):
        context = super().get_context_data(**kwargs)
        context['article_creation_form'] = ArticleCreationForm()
        context['is_admin'] = self.get_admin_role(self.request)
        return context

    def post(self, request, *args, **kwargs):
        article_form = ArticleCreationForm(request.POST, request.FILES)
        if article_form.is_valid():
            article_form.save()
            return redirect('information')
        return redirect('information')
```

Рисунок 3.11 – Відображення для сторінки з корисною інформацією

Це відображення-клас, яке працює з моделлю статей (Article) та формує з нього список. В ньому реалізовано два методи. Перший відповідає за відправку даних контексту в шаблон, для подальшого їх використання в ньому. Другий метод відповідає за обробку HTTP Post методу та заповнює даними з цього запиту форму створення статті та створює відповідний запис в базі даних. Сама форма представляє собою клас, який наслідується від forms.ModelForm та відповідає за зв’язок моделі та даних з методу POST. Приклад форми створення статті показано на рисунку 3.12.

```
class ArticleCreationForm(forms.ModelForm):
    class Meta:
        model = Article
        fields = '__all__'
```

Рисунок 3.12 – Форма створення нової статті

В класі форми визначається модель та визначаються поля, які будуть задіяні в формі.

Так як дані форми були передані в контекст шаблону, то в шаблоні відповідно її можна відобразити так, як показано на рисунку 3.13.

```
<div class="modal-body">
    <form method="post" enctype="multipart/form-data">
        {% csrf_token %}
        {{ article_creation_form.as_p }}
        <button type="submit" class="btn btn-primary">Зберегти</button>
    </form>
</div>
```

Рисунок 3.13 – Рендер форми в шаблоні

Усі інші форми додатку були створені та відображені за аналогією до попередньої. Додатково в усіх формах використовуються лише POST запити з використанням csrf токену, що додатково захищає сайт від підробки даних користувачів за допомогою генерації секретного токену [29].

3.2 Використання розробленого продукту

Після відкриття сайту користувач потрапляє на головну сторінку додатку, яку продемонстровано на рисунку 3.14.

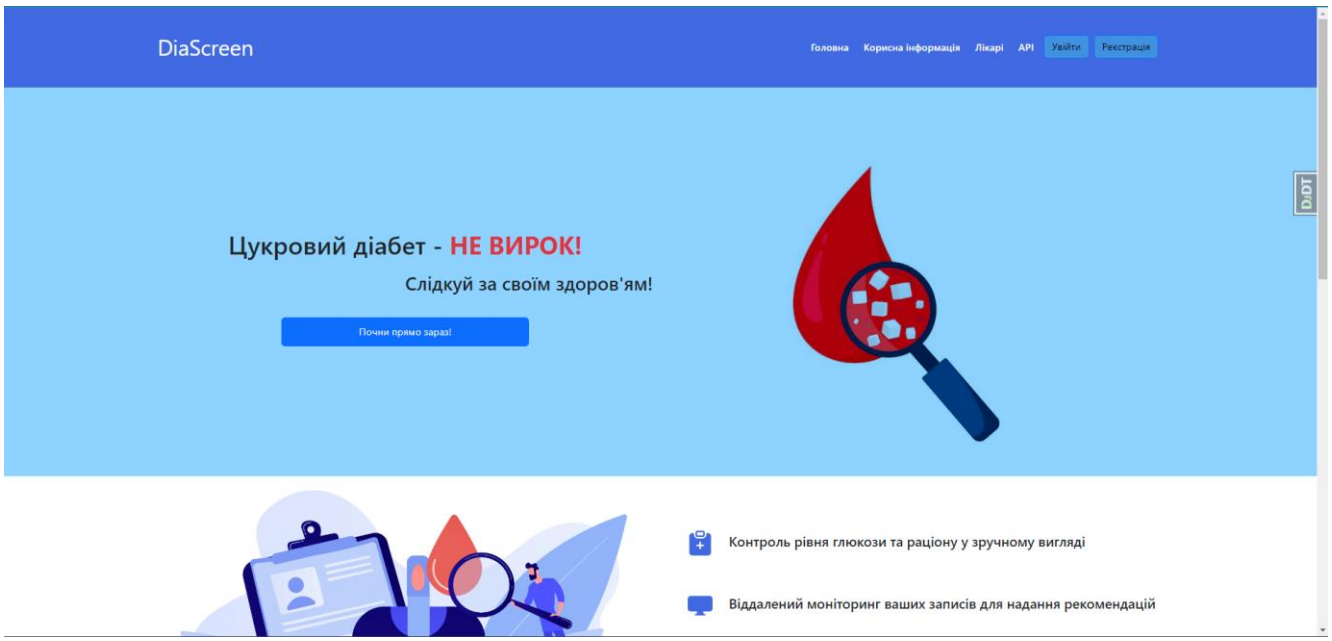


Рисунок 3.14 – Головна сторінка додатку-довідника з цукрового діабету

Незареєстрований користувач може переглянути необхідну інформацію про додаток на головній сторінці, а також перейти до довідкової сторінки з корисною інформацією у вигляді статей (рис. 3.15).

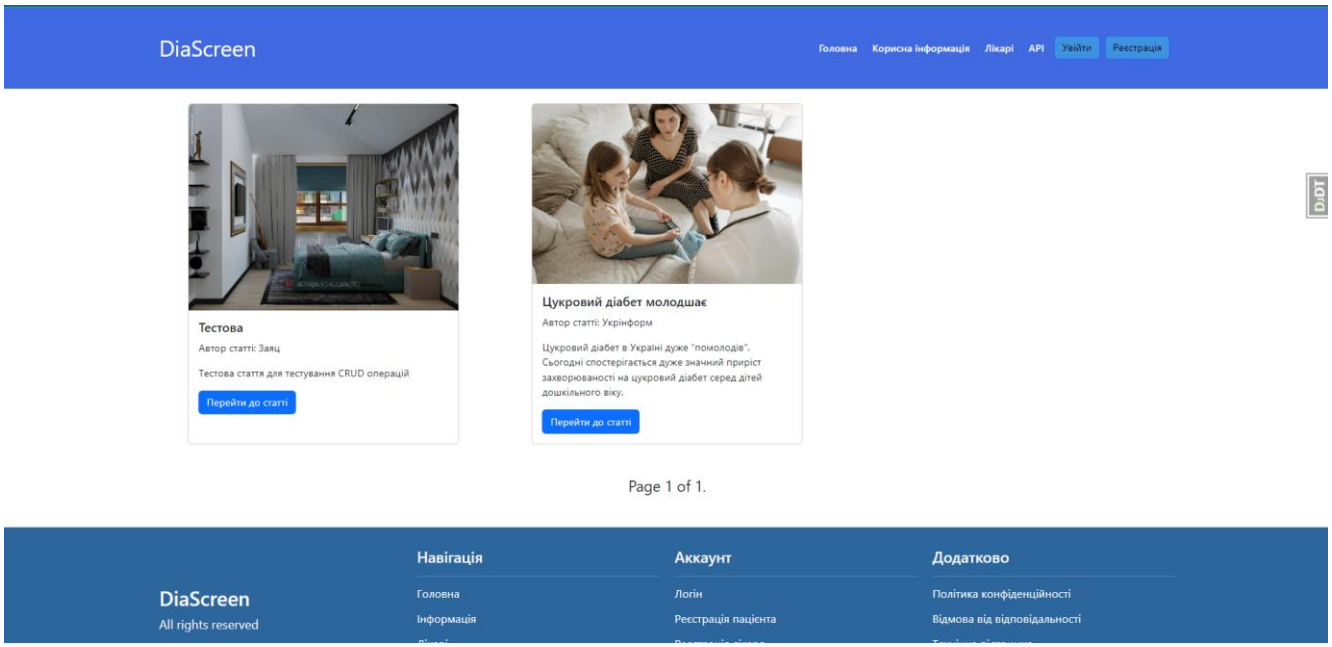


Рисунок 3.15 – Сторінка з довідковою інформацією

Далі користувач може увійти в свій аккаунт або зареєструвати новий. При виборі реєстрації відкриється сторінка з вибором типу реєстрації, як пацієнт чи

лікар (рис. 3.16). Після вибору варіанта користувача перекине на відповідну форму реєстрації, яку продемонстровано на рисунку 3.17.

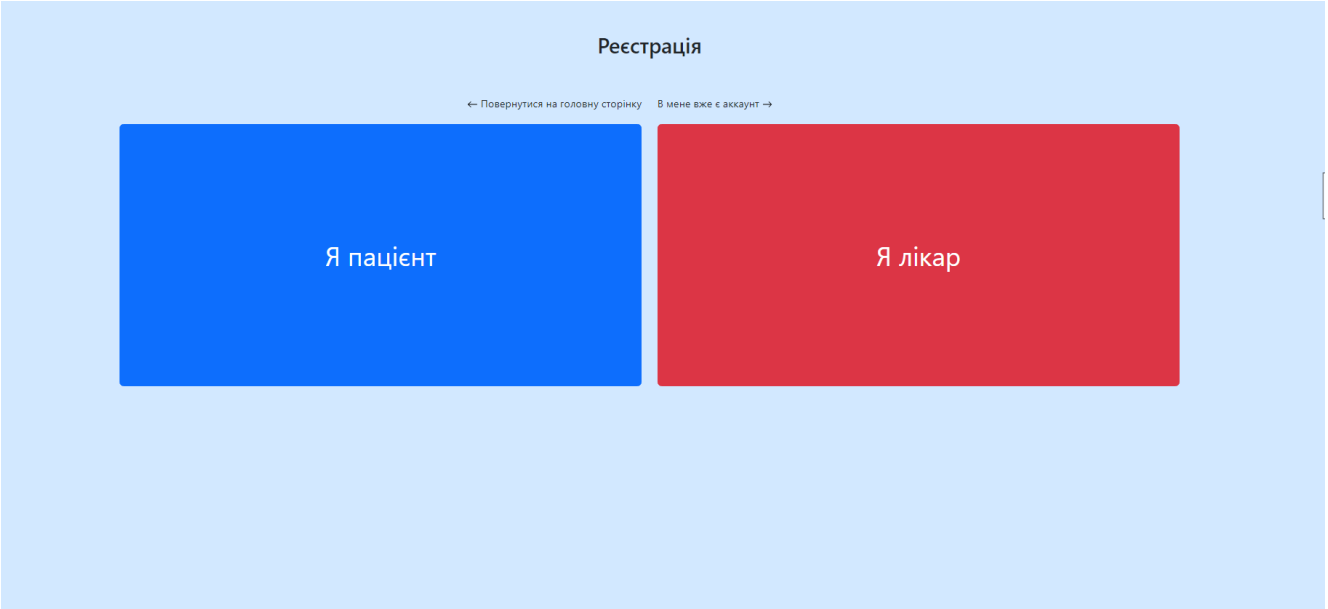


Рисунок 3.16 – Вибір типу реєстрації

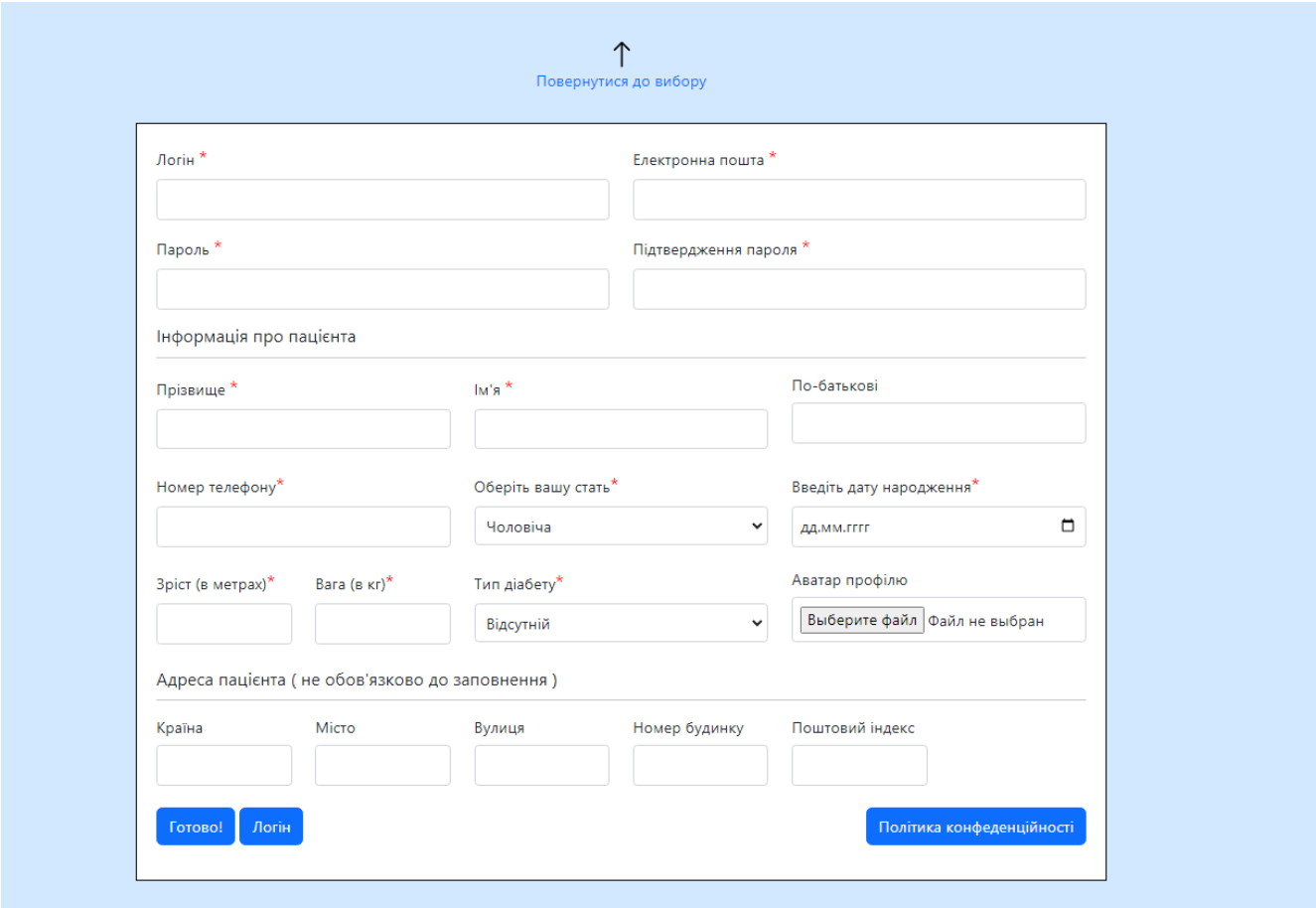


Рисунок 3.17 – Реєстраційна форма пацієнта

У лікаря аналогічна реєстраційна форма. Якщо користувач вже зареєстрований він переходить на сторінку авторизації (рис. 3.18). Додатково передбачено та реалізовано механізм відновлення паролю, користувач вводить свою пошту та через SMTP протокол отримує повідомлення з посиланням на відновлення та змінює пароль.

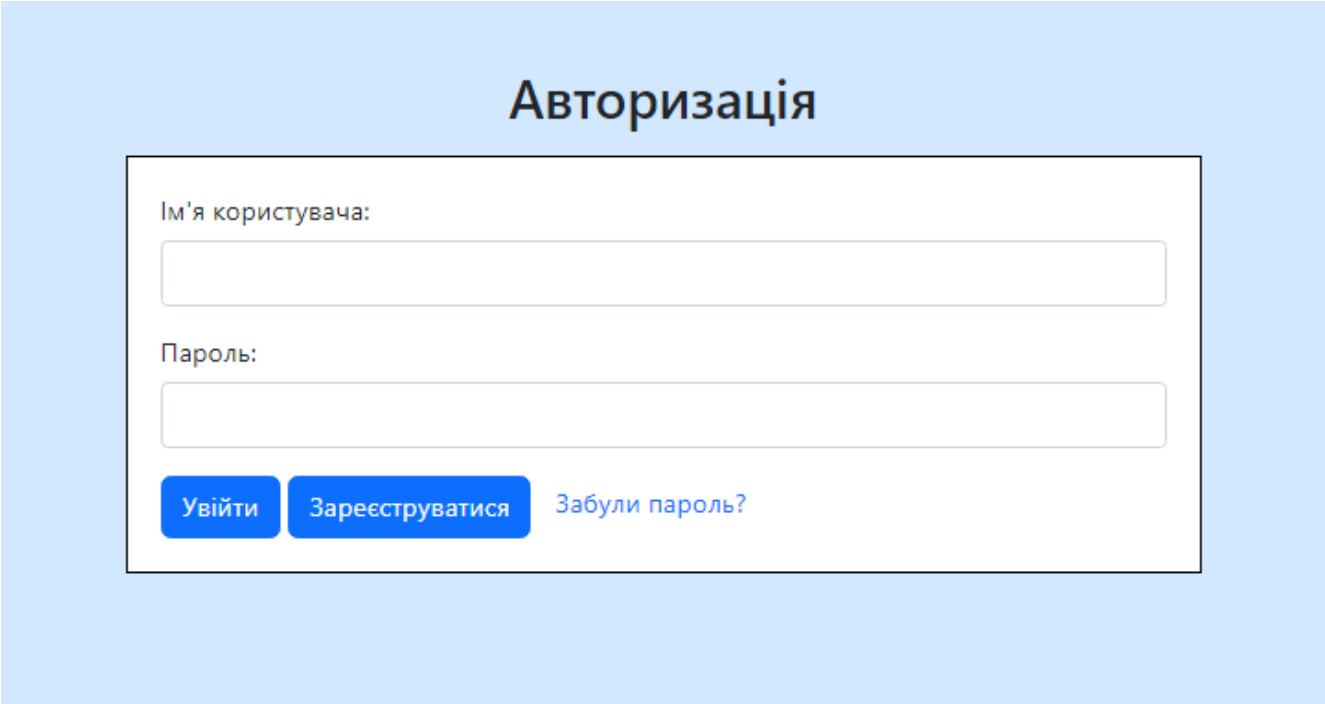


Рисунок 3.18 – Сторінка авторизації додатку

Після авторизації у користувача є 2 опції: перехід та налаштування профіля з прив’язкою лікаря чи перехід до картки для ведення записів. При переході в профіль користувача, пацієнт бачить дані свого акаунту з можливостю їх подальшого редагування за допомогою форми, а також бачить форму прив’язки лікаря. Профіль користувача продемонстровано на рисунку 3.19.

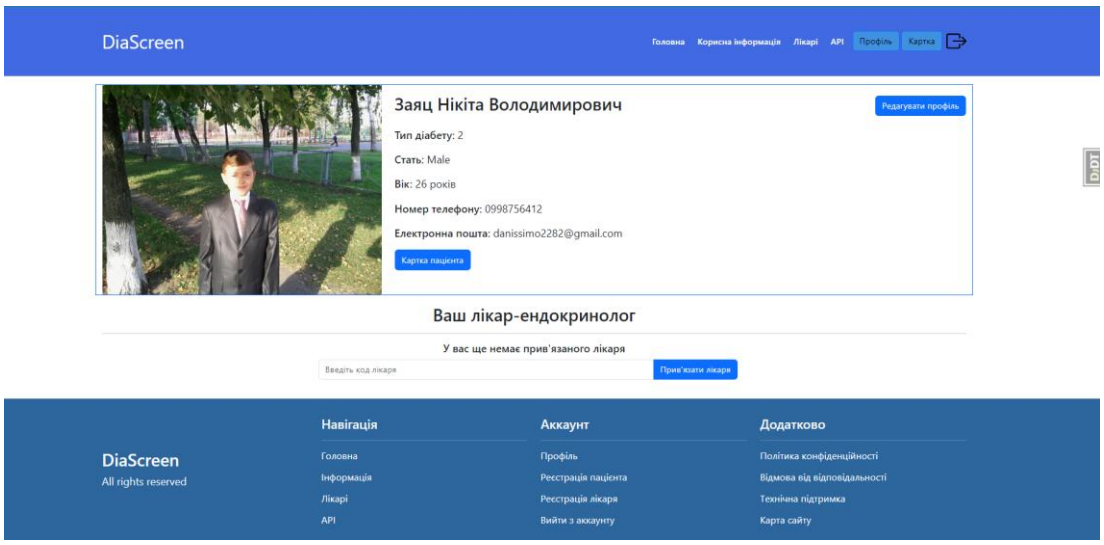


Рисунок 3.19 – Профіль пацієнта

Для прив’язки лікаря потрібно вставити в форму прив’язки його код, знайти цей код можна у профілі лікаря. Користувачу треба перейти на сторінку пошуку лікарів та знайти потрібного ендокринолога. В профілі лікаря треба натиснути кнопку отримати код лікаря і його буде скопійовано в буфер обміну. Профіль лікаря продемонстровано на рисунку 3.20.

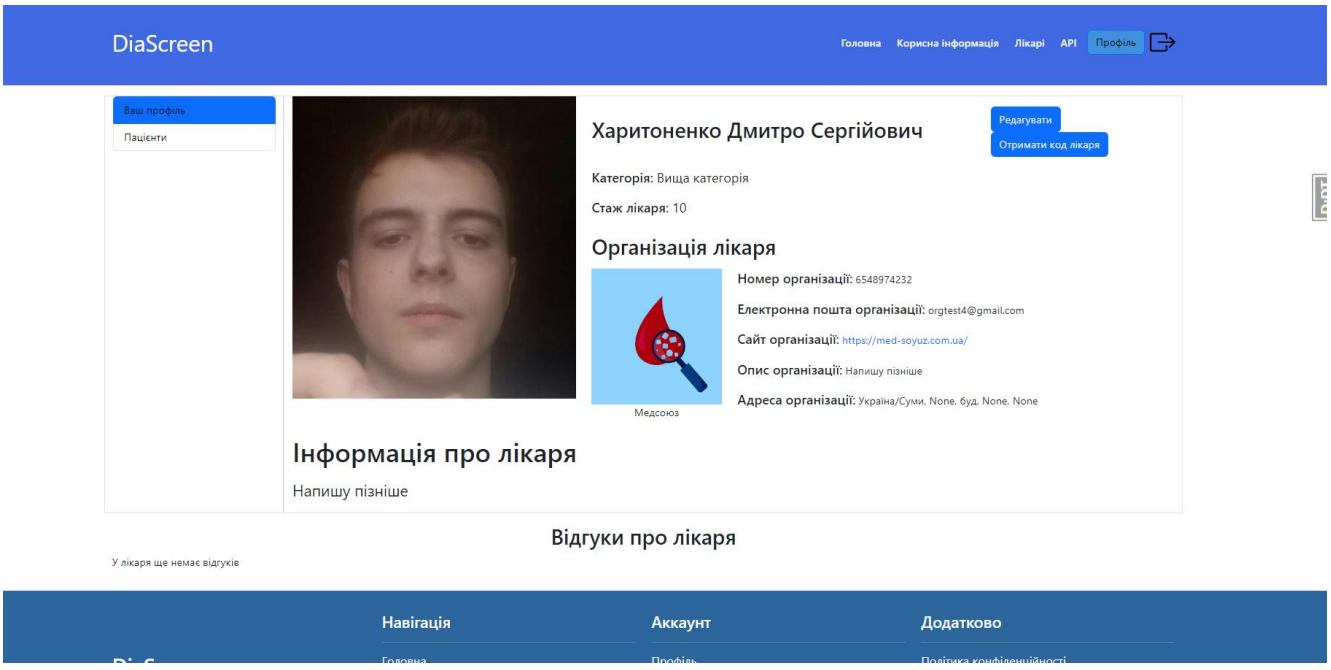


Рисунок 3.20 – Профіль лікаря

Скопійований код потрібно вставити в форму прив’язки лікаря і його буде успішно прив’язано. Профіль пацієнта з прив’язаним лікарем продемонстровано на рисунку 3.21.

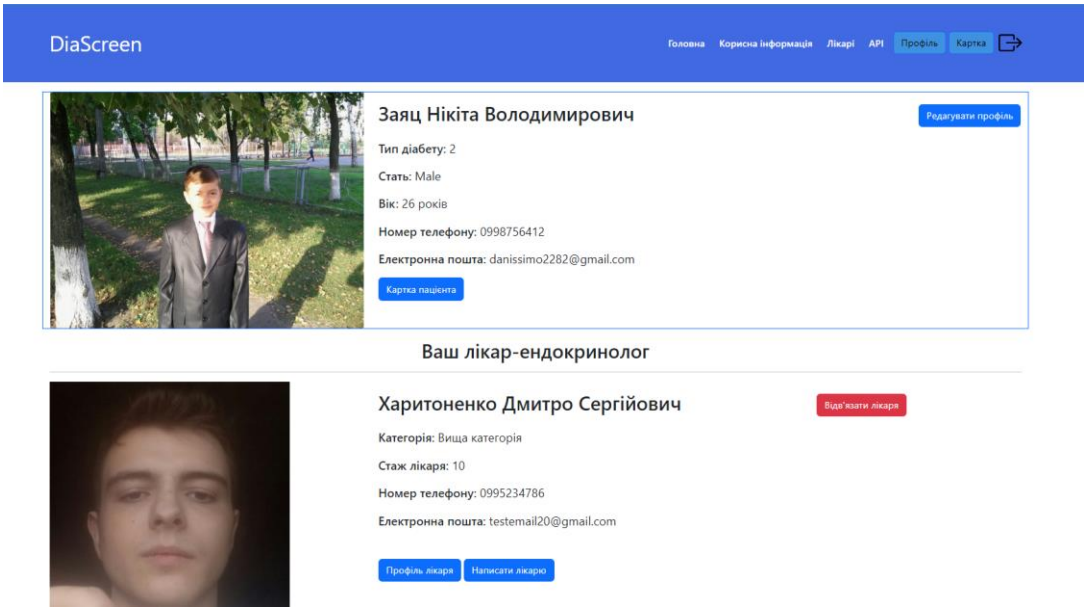


Рисунок 3.21 – Профіль з прив’язаним лікарем

В будь-який момент часу можна відв’язатися від лікаря або отримати консультацію за допомогою чату. Щоб відкрити чат з лікарем потрібно натиснути на кнопку написати лікаря. Вікно кімнати текстового чату продемонстровано на рисунку 3.22.

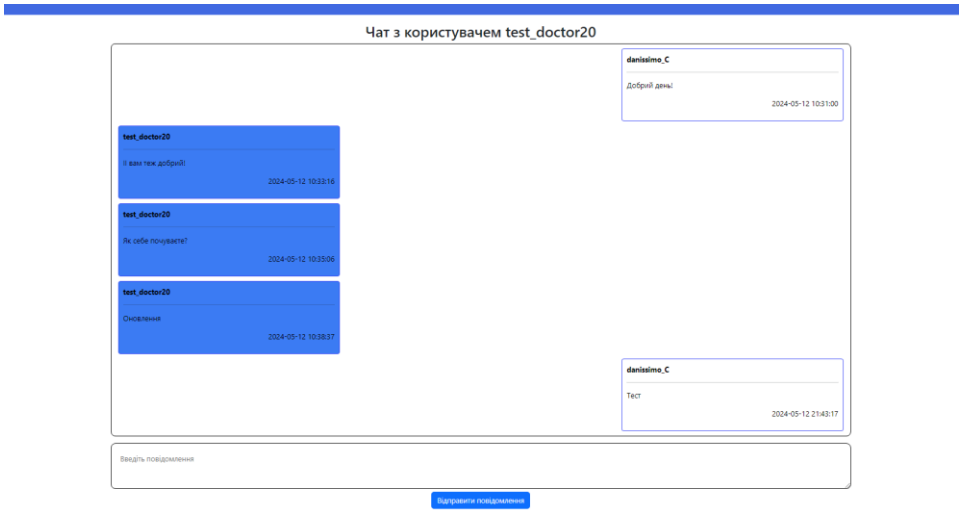


Рисунок 3.22 – Чат кімната з лікарем

Ще однією опцією пацієнта в профілі є перехід в його власну картку для ведення записів. Для захисту даних передбачено, щоб тільки конкретний пацієнт та його прив’язаний лікар бачили цю картку. В картці також є меню вибору потрібного запису: заміри глюкози, прийоми їжі, тощо. Сторінка картки пацієнта показана на рисунку 3.23.

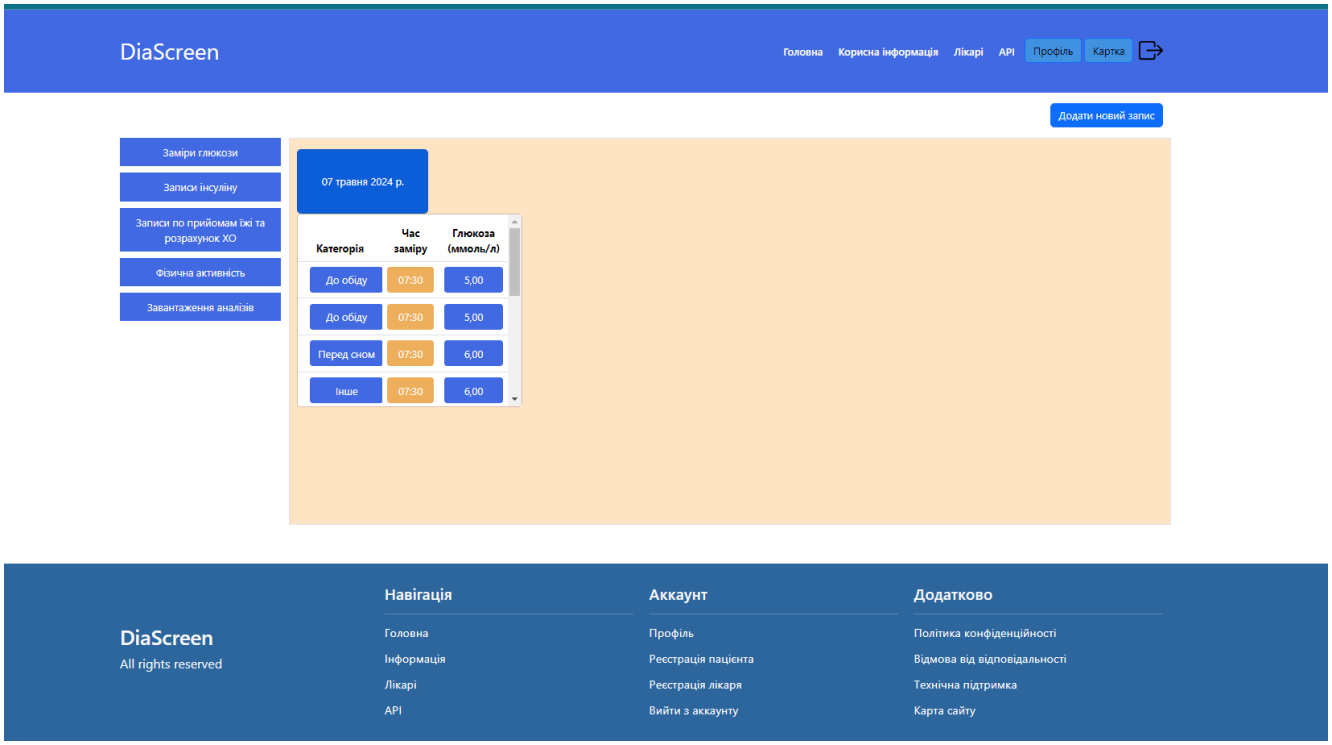


Рисунок 3.23 – Картка пацієнта з записом

Записи групуються по конкретній даті, щоб не перенавантажувати сторінку. Для додавання нового запису треба натиснути кнопку «Додати новий запис». Відкриється модальне вікно з формой запису, як показано на рисунку 3.24.

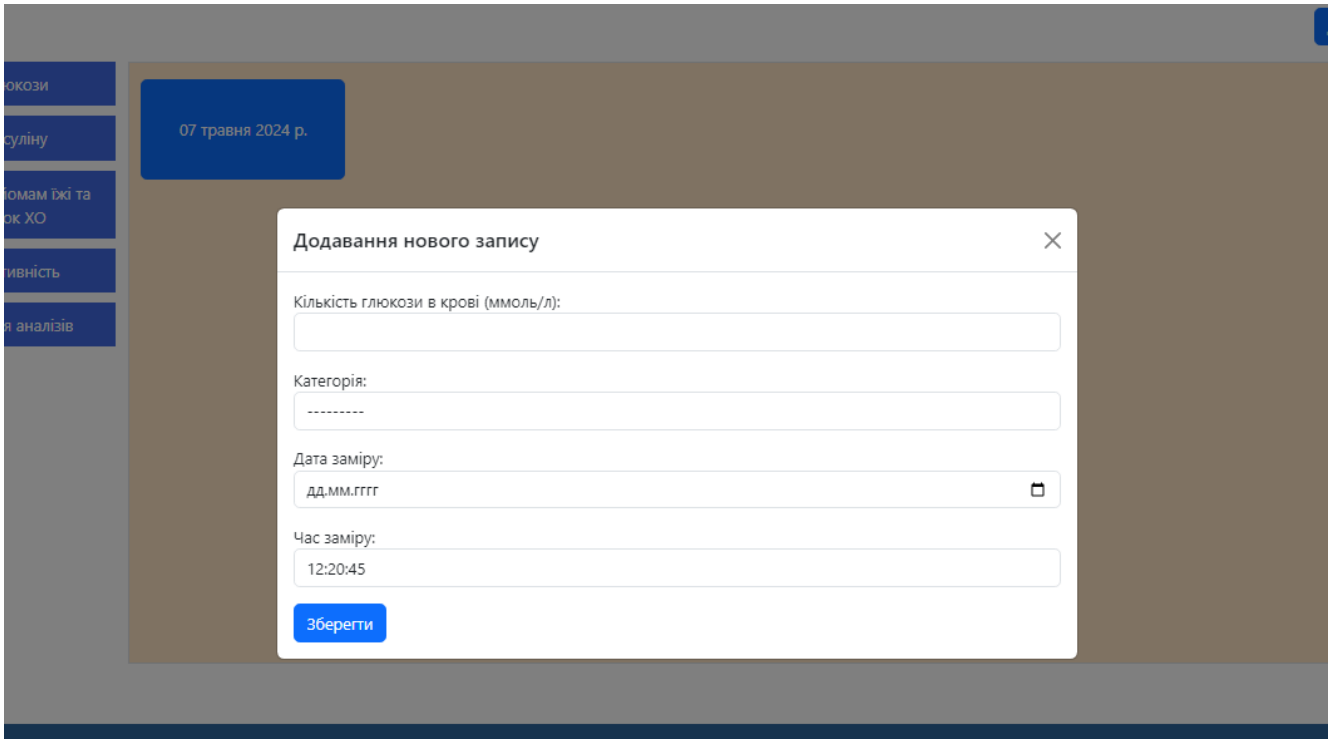


Рисунок 3.24 – Додавання нового запису в картку пацієнта

3.3 Тестування розробки

Одним із головних етапів життєвого циклу проєкту є тестування. Тести можуть бути різними та на різних етапах, а також покривати різні обсяги додатку.

Розроблений додаток-довідник з цукрового діабету тестувався декількома методами. Клієнтська частина тестувалася за допомогою методу чорного ящику, а саме перехід між сторінками, спрацювання необхідних кнопок, видимість контенту для різних ролей. Приклади тест репортів по тестуванню клієнтської частини додатку показано в таблиці 3.1.

Таблиця 3.1 – Приклад тест-репортів по тестуванню клієнтської частини

№	Умова	Очікуваний результат	Фактичний результат	0/1
1	Перехід в профіль лікаря	Відображення профілю лікаря, можливість написання лікарю, для пацієнта недоступний список інших пацієнтів	Профіль лікаря відображається коректно, недоступні для користувача пункти не відображаються	1
2	Написання відгуків лікарю від одного пацієнта	Один пацієнт може залишити тільки один відгук для лікаря	Перший відгук успішно додається, усі наступні ні	1
3	Додавання нового запису в картці по однаковій даті	Новий блок дати з’являтися не буде, запис з’явиться під попереднім	Запис зберігаються та правильно групуються по датам, запис по однаковій даті додається в неї нижче попереднього	1
4	Написання нового повідомлення лікарю	Повідомлення успішно відправиться та відобразиться на сторінці з чатом	Повідомлення успішно відображається в чат-кімнаті з правильними даними, текстом та часом. Також повідомлення має вигляд відправленного	1
5	Отримання нового повідомлення	Написане лікарем повідомлення відобразиться на сторінці з чатом у вигляді отриманого	Повідомлення успішно відображається в чат-кімнати та має вигляд отриманого.	1
6	Копіювання коду лікаря	В профілі лікаря при натисканні на кнопку «Отримати код лікаря» повинен копіюватися його унікальний код в буфер обміну	Унікальний код лікаря успішно копіюється до буферу обміну	1

Продовження таблиці 3.1.

№	Умова	Очікуваний результат	Фактичний результат	0/1
7	Редагування особистого профілю	При натисканні на кнопку редагування профілю повинна відкриватися сторінка з формою. Після внесення змін, вони повинні відображатися в профілі	Сторінка з формами редагування успішно відкривається, оновлені дані відображуються в профілі користувача	1

На серверній частині розрахунки, а також деякі методи покриті Unit-тестами [30]. Приклад написаних Unit-тестів продемонстровано на рисунку 3.25.

```python
from django.core.exceptions import ValidationError
from django.db import connection
from login.models import Patient

from datetime import datetime

class DataBaseConnectionTests(TestCase):
    def test_db_connection(self):
        try:
            cursor = connection.cursor()
        except Exception as e:
            self.fail(f"Database connection error: {str(e)}")


class AgeCalculationTests(TestCase):
    def test_valid_year_calculation(self):

        date_of_birth = datetime(year=2002, day=22, month=11)
        instance = Patient(date_of_birth=date_of_birth)

        age = instance.calculate_age()
        self.assertEqual(age, 21)


class BodyIndexCalculationTests(TestCase):
    def test_calculate_body_mass_index(self):
        height = 1.93
        weight = 85
        expected_result = weight / (height ** 2)
        instance = Patient(height=height, weight=weight)

        body_mass_index = instance.calculate_BMI()
        self.assertEqual(body_mass_index, expected_result)
```

Рисунок 3.25 – Приклади Unit-тестування додатку

До прикладу, в тестах показаних на рисунку 3.25 тестується підключення до бази даних, розрахунок віку людини по даті народження та метод розрахунку індексу маси тіла, які наявні в моделі пацієнта.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи бакалавра було розроблено веб-орієнтовний додаток-довідник з цукрового діабету для інформування пацієнтів та контролю їх показників по хворобі, з можливістю віддаленої співпраці з лікарем.

При дослідженні предметної області медицини та цукрового діабету, було зв'ясовано актуальність розроблюваного проєкту та задачі, які треба було розв'язати. Також було проведено порівняльний аналіз додатків-аналогів для виділення основних відмінностей та формування на їх основі функціональних вимог до проєкту.

В ході проєктування додатку було створено функціональні та структурні моделі різного типу. Для розуміння основних процесів додатку було створено IDEF0 діаграму та декомпозиції різних рівнів. Для розуміння того, як додаток можуть використовувати різні клієнти, було спроєктовано модель варіантів використання. Також було побудовано модель бази даних, для демонстрації збереження даних та їх зв’язків в додатку. Додатково було побудовано моделі компонентів та архітектури проєкту.

Для програмної реалізації було обрано технології HTML, CSS, Bootstrap та JavaScript. Для серверної частини використовувалася мова прогармування Python з веб-фреймворком Django, а для збереження даних використовувалася PostgreSQL.

В ході тестування додатку було проведено тестування клієнтської частини додатку методами чорного ящику, а саме порівняння очікуваного результату з фактичним та створення відповідних тест-репортів. Додатково було створено UNIT-тести, для покриття проблемних місць в коді. По всім тестам було отримано позитивний результат, все працює вірно.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. How technology is changing the world of medicine - hunimed. Hunimed. URL: https://www.hunimed.eu/news/technology-changing-world-medicine/ (дата звернення: 20.04.2024).

2. Diabetes. World Health Organization (WHO). URL: https://www.who.int/news-room/fact-sheets/detail/diabetes (дата звернення: 20.04.2024).

3. Ордатій Н. Діджіталізація: плюси і мінуси. DoctoRoomUA. URL: https://room.doctorthinking.org (дата звернення: 20.04.2024).

4. Nicoara R. How to be a web developer: a complete beginner's guide on what to know and where to start 1st ed. edition : навчальний посібник. Apress, 2023. 230 p.

5. Web development best practices 2024: expert strategies revealed. Web Development Best Practices 2024: Expert Strategies Revealed. URL: https://atlasiko.com/blog/web-development/web-developmentbestpractices/ (дата звернення: 20.04.2024).

6. В Україні 2 млн 325 тис людей з діабетом - Міжнародна діабетична федерація. Атлас: Діабет в Україні. URL: https://diabetesatlas.com.ua/ua/ (дата звернення: 21.04.2024).

7. Поширеність діабету серед молоді значно зросла з 2001 року. Атлас: Діабет в Україні. URL: https://diabetesatlas.com.ua/ua/diabet-u-sviti/tpost/1idm9a84t1-poshirenst-dabetu-sered-molod-znachno-zr (дата звернення: 21.04.2024).

8. Meheut C. For ukrainian refugees, seeing the doctor can be worth a risky trip home. New york times. 2023. 14 листоп. URL: https://www.nytimes.com/2023/11/14/world/europe/ukraine-refugees-return-doctor.html (дата звернення: 21.04.2024).

9. Frain B. Responsive Web Design with HTML5 and CSS3 - Second Edition: Build responsive and future-proof websites to meet the demands of modern web users. Packt Publishing, 2017. 312 с.

10. Vickler A. Javascript: Javascript Front End Programming. Independently Published, 2021.

11. Schonig H.-J. Mastering PostgreSQL 13: Build, administer, and maintain database applications efficiently with PostgreSQL 13, 4th Edition. Packt Publishing, 2020. 476 с.

12 . Writing your first Django app, part 1 | Django documentation. Django Project. URL: https://docs.djangoproject.com/en/5.0/intro/tutorial01/ (дата звернення: 22.04.2024).

13. Маттес Е. Пришвидшений курс Python. Практичний, проєктно-орієнтований вступ до програмування : підручник. 2-ге вид. Львів : Вид-во Старого Лева, 2021. 600 с.

14. Making queries | Django documentation. Django Project. URL: https://docs.djangoproject.com/en/5.0/topics/db/queries/ (дата звернення: 22.04.2024).

15. Technologies R. Glycemic Index. Diabetes diary – Додатки в Google Play. Android Apps on Google Play. URL: https://play.google.com/store/apps/details?id=com.creamsoft.mygi&hl=uk (дата звернення: 22.04.2024).

16. Sirma Medical Systems. Diabetes:M. Версія 9.0.6. Sirma Medical Systems, 2013. URL: https://diabetes-m.com (дата звернення 22.04.2024).

17. Solutions H. Діабет – Додатки в Google Play. Android Apps on Google Play.
URL: https://play.google.com/store/apps/details?id=ru.hintsolutions.diabets&hl=uk (дата звернення: 22.04.2024).

18. Get started with Bootstrap. Bootstrap · The most popular HTML, CSS, and JS library in the world. URL: https://getbootstrap.com/docs/5.3/getting-started/introduction/ (дата звернення: 22.04.2024).

19. Що таке API?. Highload.today - медіа для розробників. URL: https://highload.today/uk/shho-take-api-navishho-vin-neobhidnij-i-yaku-koristnese/ (дата звернення: 22.04.2024).

20. The Complete Guide To Understand IDEF Diagram | EdrawMax Online. Edrawsoft. URL: https://www.edrawmax.com/article/the-complete-guide-to-understand-idef-diagram.html (дата звернення: 13.05.2024).

21. What is Use Case Diagram?. Ideal Modeling & Diagramming Tool for Agile Team Collaboration. URL: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/ (дата звернення: 15.05.2024).

22. Morales Jr. What is A UML Component Diagram. MindOnMap | Free Mind Mapping Tool to Draw Ideas Easily Online. URL: https://www.mindonmap.com/uk/blog/uml-component-diagram/ (дата звернення: 15.05.2024).

23. Maxym Zosym. Моделювання даних (Data Modelling). *Maxym Zosym*. URL: https://www.maxzosim.com/data-modelling/ (дата звернення: 17.05.2024).

24. Maple C. Understanding Django MVT architecture and view functions| Django Full Course for Beginners | Lesson… Medium. URL: https://medium.com/@CodeMaple/understanding-django-mvt-architecture-and-view-functions-django-full-course-for-beginners-lesson-39c8da093b44 (дата звернення: 15.05.2024).

25.Що таке Git?. *QualityAssuranceGroup*. URL: https://qagroup.com.ua/publications/shcho-take-git/ (дата звернення: 18.05.2024).

26. The Django template language | Django documentation. *Django Project*. URL: https://docs.djangoproject.com/en/5.0/ref/templates/language/ (дата звернення: 18.05.2024).

27. freeCodeCamp.org. AJAX Tutorial: What AJAX Is and How to Use it. *freeCodeCamp.org*. URL: https://www.freecodecamp.org/news/ajax-tutorial/ (дата звернення: 18.05.2024).

28. Отримуємо максимум від Django ORM. *DevZone*. URL: https://devzone.org.ua/post/otrymuyemo-maksymum-vid-django-orm (дата звернення: 18.05.2024).

29. What Is Cross-Site Request Forgery (CSRF) and How Does It Work? | Synopsys. *Synopsys | EDA Tools, Semiconductor IP and Application Security Solutions*. URL: https://www.synopsys.com/glossary/what-is-csrf.html#:~:text=A%20CSRF%20token%20is%20a,token%20for%20every%20user%20session. (дата звернення: 18.05.2024).

30. Юніт тести: процес написання та інструменти для запуску. *FoxmindEd*. URL: https://foxminded.ua/yunit-testy/ (дата звернення: 18.05.2024).

ДОДАТОК А

ТЕХНІЧНЕ ЗАВДАННЯ

на розробку інформаційної системи
«Web-додаток для віддаленого скринінгу пацієнтів з цукровим діабетом»

ПОГОДЖЕНО:

Доцент кафедри інформаційних технологій

_____ Ващенко С.М.

Студент групи ІТ-02

_____ Заяц Д.О.

Суми 2024

1. Призначення й мета створення web-додатку

1.1 Призначення web-додатку

Web-додаток повинен надавати користувачам можливість вести та переглядати записи по цукровому діабету, отримати консультацію з лікаря онлайн, мати можливість доступу до корисної загальної інформації щодо захворювання.

1.2 Мета створення web-додатку

Надання зручного простору ознайомлення та отримання консультацій по цукровому діабету.

1.3 Цільова аудиторія

До цільової аудиторії web-додатку можна віднести людей з цукровим діабетом, які хочуть вести записи по хворобі в онлайн форматі. Також до цільової аудиторії входять лікарі-ендокринологів, які хочуть вести віддалений скринінг своїх пацієнтів.

2 Вимоги до web-додатку

2.1 Вимоги до web-додатку в цілому

2.1.1 Вимоги до структури й функціонування web-додатку

Web-додаток має бути доступним в мережі Інтернет під доменним іменем gloss.zzz.com.ua. Web-додаток повинен складатися з розділів, які залежать один від одного.

2.1.2 Вимоги до персоналу

Від персоналу web-додатку можуть вимагатися поверхневі знання з HTML та CSS, а також реляційних баз даних та їх адміністрування.

2.1.3 Вимоги до збереження інформації

Уся інформація надана у web-додатку буде зберігатися у реляційній базі даних PostgreSQL. Особиста інформація від клієнта, така як паролі, показники тощо, повинна бути захищена.

2.1.4 Вимоги до розмежування доступу

Створений web-додаток повинен бути загальнодоступним.

Права доступа у web-додатку розподіляються по трьом групам користувачів: звичайні користувачі, як зареєстровані так і не зареєстровані, або пацієнти, лікарі та адміністратори.

Незареєстровані користувачі можуть переглядати загальну інформацію про додаток, блог з корисною інформацією про цукровий діабет, матимуть змогу шукати лікарів.

Користувач після реєстрації отримає роль пацієнта, якщо реєструвався як пацієнт. В нього є такі права доступа: заповнення та налаштування свого власного профілю, перегляд та ведення своєї картки пацієнта, завантаження аналізів, записи замірів цукру та інше, також пацієнт може зв’язатися зі своїм лікарем.

Лікарі реєструються окремо від пацієнтів та мають схожі права доступа. Відмінність полягає у наявності свого власного лікарського профілю, де лікар може переглянути інформацію по прив’язаним до нього пацієнтам, надавання пацієнтам поради по їх введеним показникам, генерація унікального коду лікаря для прив’язки пацієнтів.

Адміністратор web-додатку може переглядати усю інформацію та змінювати дані, крім чутливої інформації користувача.

Адміністративна панель створюється автоматично засобами фреймворку Django, а доступ до неї відбувається за унікальними логінами та паролями адміністраторів.

2.2 Структура web-додатку

2.2.1 Загальна інформація про структуру web-додатку

Структура web-додатку складається з невеликої кількості взаємопов’язаних сторінок, які також представлені у навігаційній панелі головної сторінки та у підвалі додатку.

Сторінки розроблюваного web-додатку:

Головна – на сторінці представлена основна інформація про web-додаток, його переваги та інструкція про початок користування додатком.

Корисна інформація – довідник, у вигляді блогу, який містить інформацію, яка може бути корисною для людей з цукровим діабетом.

Пошук лікарів – сторінка, на якій користувачі матимуть змогу знайти лікаря по своїм побажанням та зв’язатися з ним.

API – додаткова сторінка, в якій міститься невелика документація до API, яке пропонує web-додаток, наприклад, реєстрація лікарів через інший web-додаток. Буде доступна для адміністратора додатку.

Профіль – сторінка з інформацією про користувача, відрізняється залежно від ролі користувача. В профілі також розміщується картка користувача, його лікар та інша потрібна інформація.

Сторінка реєстрації – на сторінці проходить реєстрація в web-додатку, як пацієнтів так і лікарів.

Аутентифікація – сторінка для логіну користувачів у web-додаток.

2.2.2 Навігація

Навігація по web-додатку відбувається через різні посилання. Основні посилання знаходять в хедері сайту: головна сторінка, блог з інформацією, пошук лікарів, профіль та API.

Додаткова інформація знаходиться в підвалі сайту, а також дублюється навігація хедеру.

2.2.3 Наповнення web-додатку (контент)

Заповнення додатку контентом буде відбуватися динамічно з даних, які вносять користувачі та використовуючи інформацію з бази даних.

За наповнення контентом блогу з корисною інформацією відповідають адміністратори web-додатку.

2.2.4 Дизайн та структура додатку

Стиль web-додатку має бути сучасним, зручним та стриманим. Кольорова гамма буде складатися з відтінків синього та білого кольорів, щоб відповідати дизайнам сайтів зі сфери healthcare.

Розташування елементів на головній сторінці web-додатку схематично показано на рисунку А.1.

Рисунок А.1 – Схема головної сторінки

Макети інших сторінок продемонстровано на рисунках А.2 – А.6.

Рисунок А.2 – Макет сторінки реєстрації

Рисунок А.3 – Сторінка аутентифікації

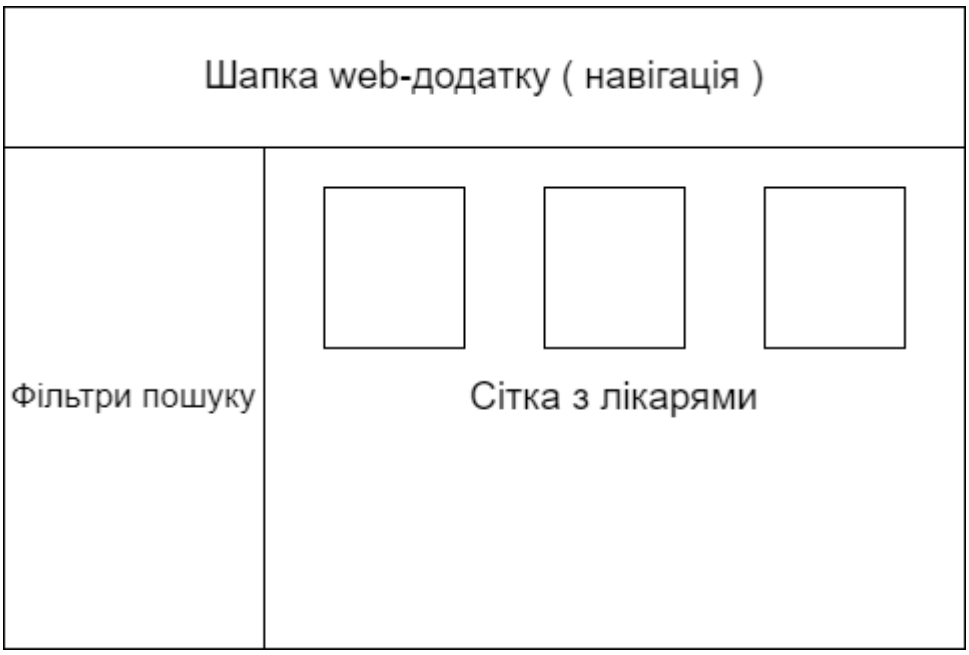


Рисунок А.4 – Макет сторінки пошуку лікарів

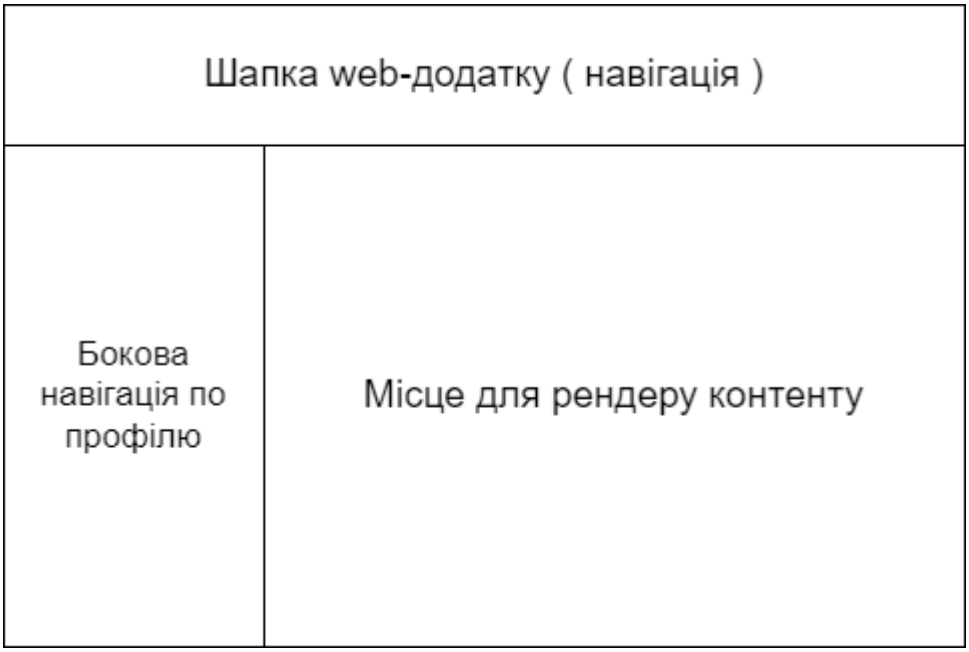


Рисунок А.5 – Схема сторінки профілю

2.2.5 Система навігації (карта web-додатку)

Карта web-додатку зображена на рисунку А.6.

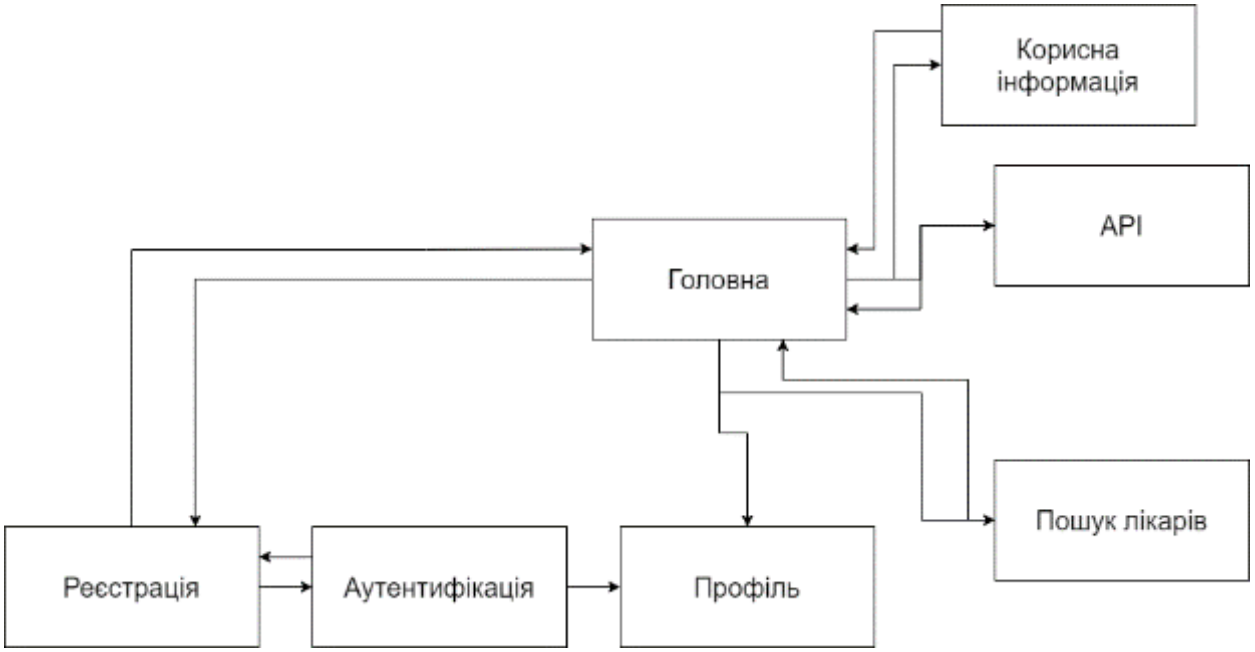


Рисунок А.2 – Карта web-додатку

2.3 Вимоги до функціонування системи

2.3.1 Потреби користувача

Потреби користувача, визначені на основі рішення замовника, представлені у таблиці А.1.

Таблиця А.1 – Потреби користувача

ID	Потреби користувача	Джерело
UN-01	Перегляд цікавої інформації по цукровому діабету	Клієнт
UN-02	Перегляд та пошук потрібного лікаря	Клієнт
UN-03	Можливість прив’язки до обраного лікаря	Клієнт
UN-04	Ведення своєї картки хвороби	Клієнт
UN-05	Записи по замірам цукру в картку пацієнта	Клієнт

Продовження табл. А.1

UN-06	Завантаження різноманітних аналізів у картку	Клієнт
UN-07	Записи про фізичні навантаження	Клієнт
UN-08	Записи про поставлені уколи та дози інсуліна	Клієнт
UN-09	Можливість консультації з лікарем	Клієнт
UN-10	Перегляд та налаштування профілю	Клієнт
UN-11	Перегляд інформації про свого лікаря	Клієнт
UN-12	Можливість отримання зворотнього зв’язку	Клієнт
UN-13	Перегляд інформації про всіх пацієнтів	Клієнт
UN-14	Перегляд інформації по кожному пацієнту	Клієнт
UN-15	Редагування інформації	Адміністратор

2.3.2 Функціональні вимоги

На основі потреб користувача були визначені такі функціональні вимоги:

- реєстрація та авторизація користувачів, різна для пацієнта і лікаря;
- перегляд блогу з корисною інформацією по захворюванню;
- налаштування профілю користувача;
- можливість звернутися до технічної підтримки сайту;
- пошук потрібного лікаря-ендокринолога (по ПІБ або лікарні);
- можливість онлайн-консультації зі своїм лікарем через чат в додатку;
- функціональна прив’язка до конкретного лікаря;
- завантаження різноманітних аналізів пацієнта в форматі pdf;
- ведення записів по замірам показника глюкози в крові;
- ведення записів про прийоми їжі та фізичну активніть: калорійність їжі, глікемічний індекс, види фізичнх вправ, тривалість тощо;
- перегляд лікарем інформації про конкретного пацієнта та його картки хвороби;

2.3.3 Системні вимоги

Даний розділ визначає, розподіляє та вказує на системні вимоги, визначені розробником. Їх перелік наведений в таблиці А.2.

Таблиця А.2 – Системні вимоги

ID	Системні вимоги	Пріоритет	Опис
SR-01	Модуль авторизації та реєстрації користувачів	M	Надає можливість реєстрації для нових користувачів та авторизації для існуючих. Передбачити окрему реєстрацію для пацієнтів та лікарів.
SR-02	Перегляд блогу з корисною інформацією	M	Забезпечення доступу до блогу з корисною інформацією по цукровому діабету для всіх груп користувачів
SR-03	Налаштування профілю користувача	M	Користувачі повинні мати можливість налаштування свого профілю під власні потреби
SR-04	Модуль пошуку лікарів	M	Надати пацієнту змогу знайти потрібного йому лікаря, або зв’язатися з іншим лікарем у будь-який час
SR-05	Модуль консультації	M	Передбачити чат з прив’язаним лікарем в реальному часі
SR-06	Наявність модулю зворотнього зв’язку	M	Надає можливість, в разі потреби, зв’язатися з технічною підтримкою сайту
SR-07	Модуль картки пацієнта	M	Користувач матиме змогу вносити дані в свою картку та заповнювати її, як вважає потрібним
SR-08	Розрахунок вуглеводів та хлібних одиниць	S	Додатково потрібно розраховувати вуглеводи, які вжив пацієнт та розраховувати приблизну хлібну одиницю
SR-09	Доступ лікаря до інформації пацієнтів	M	Забезпечити лікарів доступом до інформації конкретного прив’язаного до них пацієнта для подальшого аналізу та прийняття рішень по хворобі

Продовження табл. А.2.

SR-10	API Diascreen	S	Система буде надавати необхідну документацію та ендпоінти, для можливості створення інших додатків.
SR-11	Захист персональних даних	M	В додатку повинен бути забезпечений високий рівень захисту

Умовні позначення в таблиці А.2:

Must have (M) – вимоги, які повинні бути реалізовані в системі;

Should have (S) – вимоги, які мають бути виконані, але вони можуть почекати своєї черги;

Could have (C) – вимоги, які можуть бути реалізовані, але вони не є центральною ціллю проекту.

2.4 Вимоги до видів забезпечення

2.4.1 Вимоги до інформаційного забезпечення

Реалізація web-додатку відбувається з використанням:

- HTML5, CSS3 та Bootstrap, JavaScript
- Python Django 5.0, Django REST Framework, Django Channels
- PostgreSQL, Redis

2.4.2 Вимоги до лінгвістичного забезпечення

Web-додаток має бути виконаний українською мовою.

2.4.3 Вимоги до програмного забезпечення

Програмне забезпечення клієнтської частини повинно бути кросбраузерним та працювати в усіх сучасних браузерах.

3 Склад і зміст робіт зі створення web-додатку

Докладний опис етапів роботи зі створення web-додатку наведено в таблиці А.3.

Таблиця А.3 – Етапи створення web-додатку

№	Склад і зміст робіт	Строк розробки (у робочих днях)
1	Постановка цілей та задач	3 дні
2	Складання технічного завдання	2 дні
3	Концептуальне моделювання	5 днів
4	Створення макету дизайну web-додатку	5 днів
5	Верстка макету	3 дні
6	Моделювання та розробка бази даних	6 днів
7	Налаштування GIT репозиторію та проекту Django	1 день
8	Розробка модулів web-додатку	20 днів
9	Тестування web-додатку	3 дні
10	Завершення роботи	1 день
	Загальна тривалість робіт	49 днів

4 Вимоги до складу й змісту робіт із введення web-додатку в експлуатацію

Після тестування web-додатку його треба розмістити на хостингу. Для цього потрібно придбати доменне ім’я, на якому буде розміщено додаток в мережі Інтернет. Перед розміщенням web-додатку в мережі інтернет треба оптимізувати структуру проекту та бази даних. Додатково налаштувати проект Django під деплоїнг.

ДОДАТОК Б

ПЛАНУВАННЯ РОБІТ

Деталізація мети проєкту методом SMART. Продуктом дипломного проекту є web-додаток довідник з цукрового діабету.

Результати деталізації мети методом SMART розміщені у таблиці Б.1.

Таблиця Б.1 – Деталізація мети проєкту методом SMART

Specific	Розробити додаток-довідник з цукрового діабету з можливістю віддаленого скринінгу пацієнтів.
Measurable	Результатом роботи проекту є web-додаток розміщений на хостингу в мережі Інтернет.
Achievable	Мета є досяжною, є затверджене технічне завдання та знання з обраних технологій, а саме web-технологій HTML, CSS та JS, а також Python Django.
Relevant	У розробників в наявності є всі необхідні програмні засоби та додатній рівень кваліфікації для створення проекту.
Time-bound	Завершити розробку та запустити "DiaScreen" не пізніше 1 червня 2024 року.

Планування змісту робіт. WBS (Work Breakdown Structure – ієрархічна структура робіт) – це графічний вигляд елементів проєкту, які згруповані ієрархією у єдине ціле з продуктом проєкту. Побудуємо структуру WBS для створюваного проєкту, з детальним описом робіт, які потрібно виконати для досягнення поставлених задач. Діаграма WBS проєкту «DiaScreen» представлена на рисунку Б.1.

Планування структури виконавців. Наступним етапом після створення декомпозиції робіт WBS розробимо структуру виконавців OBS. На рисунку Б.2 представлено організаційну структуру планування проєкту. Список виконавців, що функціонують в проєкті описано в таблиці Б.2

Таблиця Б.2 – Список виконавців проєкту

Роль	ПІБ	Проєктна роль
Developer	Заяц Д.О.	Виконує front-end та back-end розробку.
Проєктувальник	Заяц Д.О.	Виконує проєктування бази даних та розробляє структуру web-додатку.
Керівник проєкту	Ващенко С.М.	Формує завдання на розробку проєкту.
Тестувальник	Заяц Д.О.	Відповідає за тестування функціоналу та дизайну web-додатку
Менеджер проекту	Заяц Д.О.	Відповідає за виконання термінів, збір та аналіз даних.

Рисунок Б.1 – WBS-структура робіт проєкту

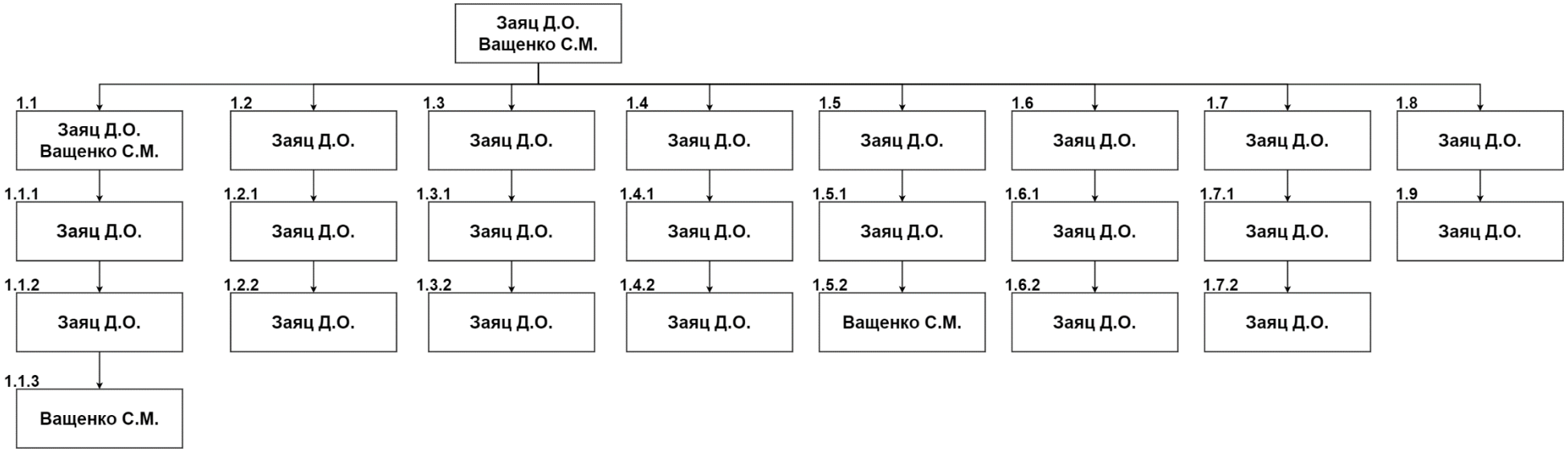


Рисунок Б.2 – OBS-структура робіт проєкту

Діаграма Ганта. Побудова календарного графіку (діаграми Ганта) є одним з важливих етапів планування проєкту, що виглядає як розклад виконання робіт з реальним розподілом дат. Завдяки йому можна отримати достовірне уявлення про тривалість процесів з обмеженнями у ресурсах, урахуванням вихідних днів та свят. Календарний графік проєкту представлено на рисунках Б.3-Б.4.

1			**Web-додаток довідник з цукрового діабету "DiaScreen"**	**65 днів**	**Нд 10.03.24**	**Ср 05.06.24**		**Заяц Д.О.;Ващенко С.М.**
1.1			**Дослідження предметної області**	**5 днів**	**Пн 11.03.24**	**Пт 15.03.24**		**Ващенко С.М.;Заяц Д.О.**
1.1.1			Визначення актуальності дослідження	1 день	Пн 11.03.24	Пн 11.03.24		Заяц Д.О.
1.1.2			Визначення функціональних та нефункціональних вимог	2 днів	Вт 12.03.24	Ср 13.03.24	3	Заяц Д.О.
1.1.3			Організація робочого процесу та планування задач	2 днів	Чт 14.03.24	Пт 15.03.24	4	Ващенко С.М.
1.2			**Створення дизайну web-додатку**	**4 днів**	**Ср 20.03.24**	**Пн 25.03.24**	**2**	**Заяц Д.О.**
1.2.1			Розробка дизайну web-додатку	2 днів	Ср 20.03.24	Чт 21.03.24		Заяц Д.О.
1.2.2			Верстка сторінок web-додатку	2 днів	Пт 22.03.24	Пн 25.03.24	7	Заяц Д.О.
1.3			**Підключення БД до web-додатку**	**6 днів**	**Вт 26.03.24**	**Вт 02.04.24**	**6**	**Заяц Д.О.**
1.3.1			Розробка структури та моделювання бази даних	4 днів	Вт 26.03.24	Пт 29.03.24		Заяц Д.О.
1.3.2			Наповнення контентом web-додатку	2 днів	Пн 01.04.24	Вт 02.04.24	10	Заяц Д.О.
1.4			**Розробка серверної частини web-додатку**	**35 днів**	**Ср 03.04.24**	**Пн 20.05.24**	**9**	**Заяц Д.О.**
1.4.1			Розробка функціоналу додатку	25 днів	Ср 03.04.24	Пн 06.05.24		Заяц Д.О.
1.4.2			Розробка додаткового функціоналу	10 днів	Вт 07.05.24	Пн 20.05.24	13	Заяц Д.О.
1.5			**Тестування web-додатку**	**4 днів**	**Вт 21.05.24**	**Пт 24.05.24**	**12**	**Заяц Д.О.;Ващенко С.М.**
1.5.1			Alpha-тестування	2 днів	Вт 21.05.24	Ср 22.05.24		Заяц Д.О.
1.5.2			Beta-тестування	2 днів	Чт 23.05.24	Пт 24.05.24	16	Ващенко С.М.
1.6			**Запуск web-додатку**	**2 днів**	**Пн 27.05.24**	**Вт 28.05.24**	**15**	**Заяц Д.О.**
1.6.1			Розміщення на хостингу	1 день	Пн 27.05.24	Пн 27.05.24		Заяц Д.О.
1.6.2			Збір відгуків про функціональність	1 день	Вт 28.05.24	Вт 28.05.24	19	Заяц Д.О.
1.7			**Відлагодження роботи web-додатку**	**3 днів**	**Ср 29.05.24**	**Пт 31.05.24**	**18**	**Заяц Д.О.**
1.7.1			Перевірка та налагодження діагностичних функцій	2 днів	Ср 29.05.24	Чт 30.05.24		Заяц Д.О.
1.7.2			Налагодження працездатності	1 день	Пт 31.05.24	Пт 31.05.24	22	Заяц Д.О.
1.8			Написання супровідної документації	2 днів	Пн 03.06.24	Вт 04.06.24	21	Заяц Д.О.
1.9			Реліз проекту	1 день	Ср 05.06.24	Ср 05.06.24	24	Заяц Д.О.

Рисунок Б.3 – Календарний план проєкту

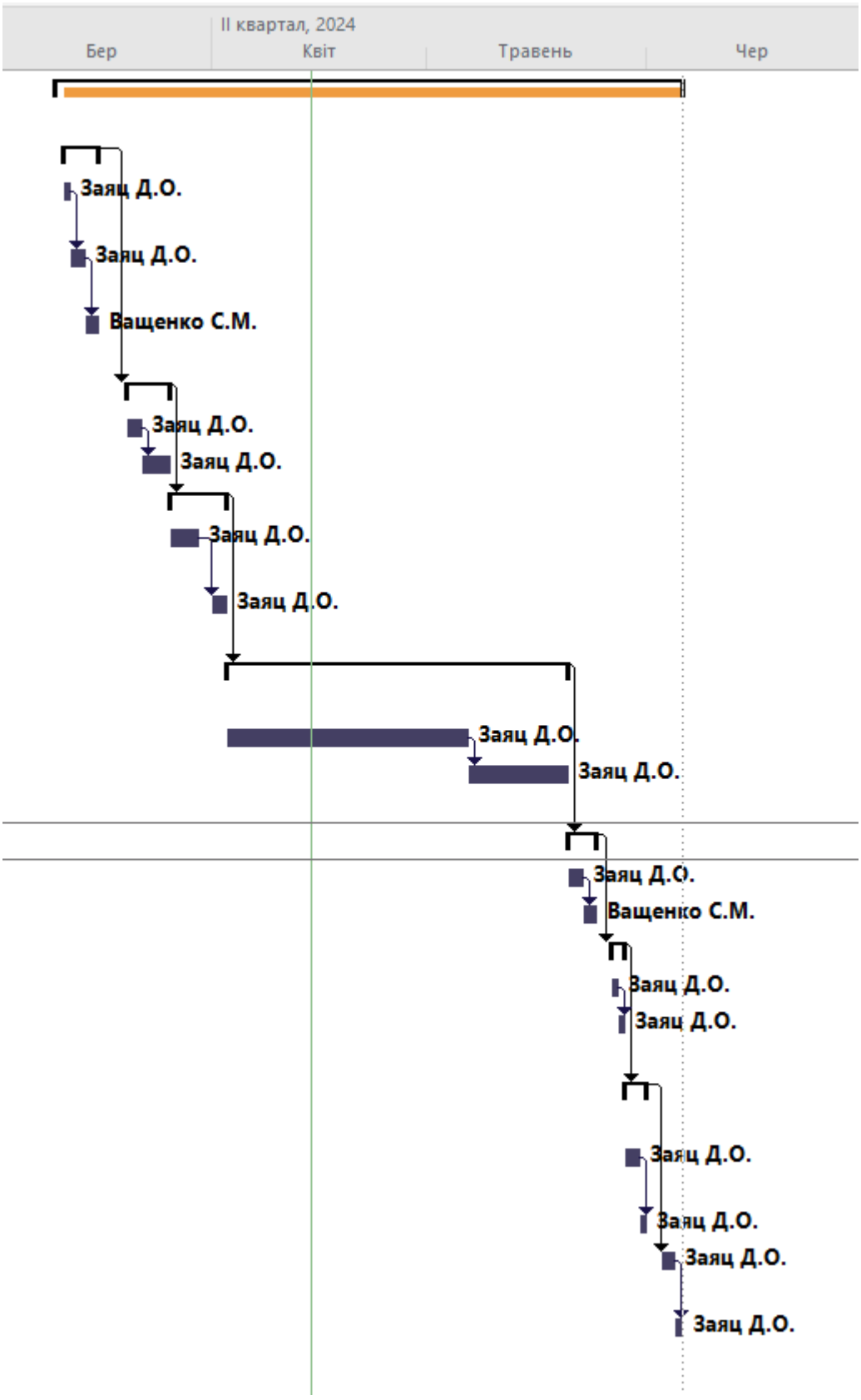


Рисунок Б.4 – Діаграма Ганта проєкту

Управління ризиками проєкту. Для успішної розробки проєкту потрібно визначити можливі ризики та провести їх аналіз. Ризики, які несуть в собі велику загрозу потрібно виправити якнайшвидше. У таблиці Б.3 надано перелік ризиків даного проєкту. Результати оцінки ризиків надано у таблиці Б.4.

Таблиця Б.3 – Ризики проєкту

№ ризику	Назва (опис) ризику
1	Технічні проблеми розробки: проблеми з програмним та апаратним забезпеченням, серверним обладнанням тощо.
2	Затримка в розробці: Хвороба, відключення світла та інше.
3	Втрата даних: несприятливі ситуації що призводять до втрати даних.
4	Зміна в вимогах користувачів: внесення змін у вимоги під час розробки.
5	Відсутність підтримки певних технологій: на різних ОС по іншому працюють технології.
6	Зміни у законодавстві: зміни в законодавстві, що впливають на проєкт, наприклад, зміна закону про збереження персональних даних.
7	Внесення змін до бази даних під час розробки може вплинути на вже існуючий функціонал, викликати конфлікти або вимагати перегляду коду та запитів.
8	Труднощі в масштабуванні: затримки в роботі при великому навантаженні на додаток
9	Проблеми з інтеграцією додаткового функціоналу: несумісність з різними технологіями, нестабільна робота
10	Недостача часу на тестування: непомічені дефекти, нестабільна робота додатку

Таблиця Б.4 – Визначення ймовірності, впливу та рангу ризиків проєкту

№ ризику	Назва (опис) ризику	Ймовірність (0,1-0,9)	Вплив (0,05-0,8)	Ранг
1	Технічні проблеми розробки	0.1	0.4	0.04

2	Затримки в розробці	0.1	0.2	0.02
3	Втрата даних	0.1	0.8	0.08
4	Зміна в вимогах користувачів	0.1	0.2	0.02
5	Відсутність підтримки певних технологій	0.3	0.4	0.12
6	Зміни у законодавстві	0.1	0.4	0.04
7	Внесення змін до бази даних	0.2	0.4	0.08
8	Труднощі в масштабуванні	0.1	0.4	0.04
9	Проблеми з інтеграцією додаткового функціоналу	0.3	0.4	0.12
10	Недостача часу на тестування	0.05	0.1	0.005

За показниками продемонстрованими в таблиці Б.4 розробимо матрицю ймовіреності та впливу ризиків на проєкт. Ця матриця допоможе розробити стратегії реагування для усунення та зменшення впливу ризиків. У результаті планування заходів реагування на ризики проєкту було отримано матрицю ймовірності виникнення та впливу ризиків, яку продемонстровано в таблиці Б.5. Зеленим кольором на матриці позначають прийнятні ризики, жовтим – виправдані, а червоним – недопустимі.

Таблиця Б.5 – Матриця ймовірності та впливу згідно проєкту

Ймовірність ризику (Й)	Вплив загрози (ризику)				
	Дуже малий	Малий	Середній	Великий	Дуже великий
	0.05	0.1	0.2	0.4	0.8
0.9	0,045	0,09	0,18	0,36	0,72

0.7	0,035	0,07	0,14	0,28	0,56
0.5	0,025	0,05	0,10	0,20	0,40
0.3	0,015	0,03	0,06	0,12R5,9	0.24
0.1	0,005 R10	0,01	0,02R2,4	0,04R1.6,8	0,08R3,7

Класифікація ризиків проєкту за рівнем, відповідно до отриманого значення індексу, представлена у таблиці Б.6. У таблиці Б.7 описано ризики та стратегії реагування на кожен із них.

Таблиця Б.6 – Шкала оцінювання за рівнем ризику

№	Назва	Межі	Ризики, які входять (номера)
1	Прийнятні	0,005≤R≤0,05	1,2,4,6,8,10
2	Виправдані	0,05<R≤0,14	3,7,5,9
3	Недопустимі	0,14<R≤0,72	

Таблиця Б.7 – Ризики проєкту та стратегії реагування

ID ризику	Статус ризику	Опис ризику	Ймовірність виникнення	Вплив ризику	Ранг ризику	Тип стратегії реагування	План А	План Б
1	Новий	Технічні проблеми розробки: проблеми з програмним та апаратним забезпеченням, серверним обладнанням тощо.	0.1	0.4	0.04	ЗМЕНШЕННЯ	Вибір надійного апаратного обладнання, ліцензійне програмне забезпечення	Закупівля нового обладнання або налагодження старого
2	Новий	Затримка в розробці: Хвороба, відключення світла та інше.	0.1	0.2	0.02	ЗМЕНШЕННЯ	Розробка гнучкого графіку та робочих процедур.	Запуск віддаленої роботи, резервного плану робіт.
3	Новий	Втрата даних: несприятливі ситуації що призводять до втрати даних.	0.1	0.8	0.08	ЗМЕНШЕННЯ	Застосування найсучасніших методів резервування та захисту даних	Миттєве відновлення втрачених даних за допомогою резервних копій
4	Новий	Зміна в вимогах користувачів: внесення змін у вимоги під час розробки.	0.1	0.2	0.02	ЗМЕНШЕННЯ	Використовувати AGILE підходи для швидкого адаптування під зміни у вимогах	Закріплення основних вимог, які не можуть бути змінені

Продовжен

5	Новий	Відсутність підтримки певних технологій	0.3	0.4	0.12	ЗМЕНШЕННЯ	Проведення детального аналізу технологічного стеку проєкту перед розробкою	Використання альтернативних технологій
6	Новий	Зміни у законодавстві: зміни в законодавстві, що впливають на проєкт, наприклад, зміна закону про збереження персональних даних.	0.1	0.4	0.04	ЗМЕНШЕННЯ	Регулярне вивчення та аналіз змін в законодавстві, що стосується обробки та збереження персональних даних	Залучення юридичних консультантів для рішення проблеми
7	Новий	Низька зацікавленість користувачів: невиправдані очікування щодо функціоналу додатку.	0.2	0.4	0.08	ЗМЕНШЕННЯ	Впровадження функціоналу, який цікавий користувачам на ринку	Організація постійних опитувань користувачів для пропозицій
8	Новий	Труднощі в масштабуванні	0.1	0.4	0.04	ЗМЕНШЕННЯ	Регулярне тестування масштабованості системи, виявлення та вирішення труднощів на ранніх етапах розробки дозволяє уникнути проблем при зростанні навантаження.	Використання хмарних рішень та сервісів дозволяє ефективно масштабувати інфраструктуру в залежності від потреб проєкту, зменшуючи необхідність у власних ресурсах та обладнанні.
9	Новий	Проблеми з інтеграцією додаткового функціоналу: несумісність з різними технологіями, нестабільна робота	0.3	0.4	0.12	ЗМЕНШЕННЯ	Тестування інтеграції на різних етапах розробки	Розробка резервного варіанту інтеграції
10	Новий	Недостача часу на тестування: непомічені дефекти, нестабільна робота додатку	0.05	0.1	0.005	ЗМЕНШЕННЯ	Розробити детальний план тестування заздалегіть	Зменшення функціоналу або відкладення деяких функцій на етапи після випуску проєкту

ДОДАТОК В

ЛІСТИНГ КОДУ

Папка проєкту

settings.py

```
import os
from pathlib import Path

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')

# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/5.0/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-^qrj!a@s!^2q@64m6%iw-um@&3vinvkjq$5v@v#xby(^w%tfil'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []


# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'login',
    'doctor_search',
    'profiles',
    'blog',
    'patient_card',
    'chat',
    'debug_toolbar',
    'DiaScreenAPI',
    'rest_framework',
]
```

```
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
    "debug_toolbar.middleware.DebugToolbarMiddleware",
]

ROOT_URLCONF = 'diabetes_proj.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, 'templates'), 'myapp/templates'],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

INTERNAL_IPS = [
    # ...
    "127.0.0.1",
    # ...
]

WSGI_APPLICATION = 'diabetes_proj.wsgi.application'


# Database
# https://docs.djangoproject.com/en/5.0/ref/settings/#databases

# DATABASES = {
#     'default': {
#         'ENGINE': 'django.db.backends.postgresql',
#         'NAME': 'diascreenDB',
#         'USER': 'postgres',
#         'PASSWORD': 'danegodiab#020423!',
#         'HOST': 'localhost',
#         'PORT': '5432'
#     }
# }
```

```
# Local settings for local database connection

try:
  from utils.local_settings import DATABASES
except ImportError:
  pass


# Password validation
# https://docs.djangoproject.com/en/5.0/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]

LOGOUT_REDIRECT_URL = 'login'
LOGIN_URL = 'login'

# Internationalization
# https://docs.djangoproject.com/en/5.0/topics/i18n/

LANGUAGE_CODE = 'uk'

TIME_ZONE = 'Europe/Kiev'

USE_I18N = True

USE_TZ = False


# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/5.0/howto/static-files/

STATIC_URL = '/static/'

STATICFILES_DIRS = [
```

```
    os.path.join(BASE_DIR, 'static'),  # Для статических файлов в директории
проекта
    os.path.join(BASE_DIR, 'login/static'),  # Для статических файлов в директории
вашего приложения
]

# Default primary key field type
# https://docs.djangoproject.com/en/5.0/ref/settings/#default-auto-field

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'


EMAIL_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'
EMAIL_HOST = 'smtp.gmail.com'
EMAIL_PORT = 587
EMAIL_USE_TLS = True
EMAIL_HOST_USER = 'danissimo228337@gmail.com'
EMAIL_HOST_PASSWORD = 'syjcssaanbxpeiob'
```

urls.py

```
from django.contrib import admin
from django.views.generic import TemplateView
from django.conf import settings
from django.conf.urls.static import static
from django.urls import path, include
from profiles.views import send_support_ticket

urlpatterns = [
    path('', TemplateView.as_view(template_name = 'index.html'), name='home'),
    path('admin/', admin.site.urls),
    path('privacy-policy', TemplateView.as_view(template_name =
'privacy_policy.html'), name='policy'),
    path('support/', send_support_ticket, name='support'),
    path('accounts/', include('login.urls')),
    path('doctor-search/', include('doctor_search.urls')),
    path('profile/', include('profiles.urls')),
    path('blog/', include('blog.urls')),
    path('card/', include('patient_card.urls')),
    path('chat/', include('chat.urls')),
    path('api/', include('DiaScreenAPI.urls')),
    path("__debug__/", include("debug_toolbar.urls")),
] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

base.html

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>DiaScreen - онлайн скринінг</title>
    <link rel="stylesheet" href="{% static 'css/reset.css' %}">
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH"
crossorigin="anonymous">
    <link rel="stylesheet" href="{% static 'css/index.css' %}">
    <link rel="stylesheet" href="{% static 'css/media.css' %}">
    {% block styles %}
    {% endblock styles %}
</head>
<body>
    {% include 'partials/_header.html' %}
    <main>
        {% block content %}
        {% endblock content %}
    </main>
    {% include 'partials/_footer.html' %}

    {% block scripts %}
    {% endblock scripts %}
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
integrity="sha384-YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz"
crossorigin="anonymous"></script>
    <script src="https://code.jquery.com/jquery-3.5.1.js " integrity="sha256-
QWo7LDvxbWT2tbbQ97B53yJnYU3WhH/C8ycbRAkjPDc=" crossorigin=" anonymous "></script>
    <script
src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js "
integrity="sha384-9/reFTGAW83EW2RDu2S0VKaIzap3H66lZH81PoYlFhbGU+6BZp6G7niu735Sk7lN
" crossorigin="anonymous "></script>
    <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js "
integrity="sha384-B4gt1jrGC7Jh4AgTPSdUtOBvfO8shuf57BaghqFfPlYxofvL8/KUEfYiJOMMV+rV
" crossorigin="anonymous "></script>
</body>
</html>
```

_header.html

```
{% load static %}
```

```
<header id="header">
    <div class="container">
            <nav class="navbar navbar-expand-lg navbar-light text-center text-
light">
                <a href="{% url 'home' %}" class="navbar-brand"
id="logo">DiaScreen</a>
                <button class="navbar-toggler ms-auto" type="button" data-
toggle="collapse" data-target="#navbarNav" aria-controls="navbarNav" aria-
expanded="false" aria-label="Toggle navigation">
                    <span class="navbar-toggler-icon"></span>
                </button>
                <div class="collapse navbar-collapse" id="navbarNav">
                    <ul class="navbar-nav ms-auto fw-medium column-gap-2">
                        <li class="nav-item active">
                          <a class="nav-link" href="{% url 'home' %}">Головна</a>
                        </li>
                        <li class="nav-item">
                          <a class="nav-link" href="{% url 'information'
%}">Корисна інформація</a>
                        </li>
                        <li class="nav-item">
                          <a class="nav-link" href="{% url 'doctors_search'
%}">Лікарі</a>
                        </li>
                        <li class="nav-item">
                          <a class="nav-link" href="#">API</a>
                        </li>
                        {% if not user.is_authenticated %}
                        <li class="nav-btn">
                            <a href="{% url 'login'%}" class="btn btn-primary
header__button" type="button">Увійти</a>
                        </li>
                        <li class="nav-btn">
                          <a href="{% url 'registration'%}" class="btn btn-primary
header__button" type="button">Реєстрація</a>
                        </li>
                        {% endif %}
                        <li class="nav-item">
                              {% if 'Doctors' in user.groups.all|join:", " %}
                              <a class="btn btn-primary header__button" href="{%
url 'doctor_profile' pk=user.pk %}">
                                  Профіль
                              </a>
                              {% elif 'Patients' in user.groups.all|join:", " %}
                              <a class="btn btn-primary header__button" href="{%
url 'patient_profile' pk=user.pk %}">
                                  Профіль
                              </a>
                              <a class="btn btn-primary header__button" href="{%
url 'patient_card' patient_id=user.pk %}">
```

```
                                Картка
                            </a>
                            {% endif %}
                        </li>
                        {% if user.is_authenticated %}
                        <li class="nav-btn">
                          <form action="{% url 'logout' %}" method="post">
                            {% csrf_token %}
                            <button type="submit" class="logout-btn">
                              <img src="{% static 'img/svg/box-arrow-right.svg' %}"
alt="Кнопка виходу">
                            </button>
                          </form>
                        </li>
                        {% endif %}
                    </ul>
                </div>
            </nav>
        </div>
</header>
```

_footer.html

```
<footer id="footer">
    <div class="container">
        <div class="row text-light align-items-center">
            <div class="col-md-3">
                <h2>DiaScreen</h2>
                <p class="fs-5">All rights reserved</p>
            </div>
            <div class="col-md-3">
                <h3 class="fs-4 mt-2">Навігація</h3><hr>
                <p><a class="footer__link" href="{% url 'home' %}">Головна</a></p>
                <p><a class="footer__link" href="{% url 'information'
%}">Інформація</a></p>
                <p><a class="footer__link" href="{% url 'doctors_search'
%}">Лікарі</a></p>
                <p><a class="footer__link" href="#">API</a></p>
            </div>
            <div class="col-md-3">
                <h3 class="fs-4 mt-2">Аккаунт</h3><hr>
                {% if user.is_authenticated %}
                <p><a class="footer__link" href="#">Профіль</a></p>
                {% else %}
                <p><a class="footer__link" href="{% url 'login' %}">Логін</a></p>
                {% endif %}
```

```
                <p><a class="footer__link" href="{% url 'patient_register'
%}">Реєстрація пацієнта</a></p>
                <p><a class="footer__link" href="{% url 'doctor_register'
%}">Реєстрація лікаря</a></p>
                <p><a class="footer__link" href="{% url 'logout' %}">Вийти з
аккаунту</a></p>
            </div>
            <div class="col-md-3">
                <h3 class="fs-4 mt-2">Додатково</h3><hr>
                <p><a class="footer__link" href="{% url 'policy' %}">Політика
конфіденційності</a></p>
                <p><a class="footer__link" href="#">Відмова від
відповідальності</a></p>
                <p><a class="footer__link" href="{% url 'support' %}">Технічна
підтримка</a></p>
                <p><a class="footer__link" href="#">Карта сайту</a></p>
            </div>
        </div>
    </div>
</footer>
```

index.html

```
{% extends 'base.html' %}
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>DiaScreen - онлайн скринінг</title>
</head>
<body>
    {% block content %}
    <main>
        <section class="section__promo">
            <div class="container">
                <div class="row align-items-center">
                    <div class="col-md-6 col-12 promo__text text-center">
                        <h1 class="text-center promo__title">Цукровий діабет -
<strong class="text-danger">НЕ ВИРОК!</strong></h1>
                        <h3 class="text-end promo__text">Слідкуй за своїм
здоров'ям!</h3>
                        <a href="#" class="btn btn-primary promo__button">Почни
прямо зараз!</a>
                    </div>
                    <div class="col-md-6 col-12 promo__img">
                        <img src="{% static 'img/blood.png' %}" class="promo__img">
```

```
                </div>
            </div>
        </div>
    </section>
    <section class="func_section">
        <div class="container">
            <div class="row align-items-center border-bottom border-primary">
                <div class="col-md-6 col-12">
                    <img src="{% static 'img/func.jpg' %}" class="img-fluid"
alt="functional image">
                </div>
                <div class="col-md-6 col-12 align-self-center mx-auto">
                    <ul class="func__list fw-medium">
                        <li class="func__item"><span>
                            <img src="{% static 'img/svg/clipboard-plus-
fill.svg' %}" alt="рисунок блокнота">
                        </span>Контроль рівня глюкози та раціону у зручному
вигляді</li>
                        <li class="func__item"><span>
                            <img  src="{% static 'img/svg/display-fill.svg' %}"
alt="рисуонок монітору">
                        </span>Віддалений моніторинг ваших записів для надання
рекомендацій</li>
                        <li class="func__item"><span>
                            <img  src="{% static 'img/svg/chat-right-text-
fill.svg' %}" alt="рисуонок іконки чату">
                        </span>Чат та консультації з вашим особистим
лікарем</li>
                        <li class="func__item"><span>
                            <img  src="{% static 'img/svg/cloud-arrow-down-
fill.svg' %}" alt="рисуонок завантаження">
                        </span>Завантаження та перегляд аналізів у зручному
форматі</li>
                    </ul>
                </div>
            </div>
        </div>
    </section>
    <section class="mt-4">
       <div class="container">
            <div class="row align-items-center">
                <div class="col-12 text-center">
                   <h3>Як почати роботу?</h3>
                </div>
            </div>
       </div>
       <div class="container mt-3 mb-5">
        <div class="row align-items-center" style="min-height: 600px;">
            <div class="col-xxl-4 col-md-6 col-12 align-self-start">
                <div class="card mx-auto guide__card">
```

```
                        <div class="card-body">
                          <h5 class="card-title">
                            <span><img src="{% static 'img/svg/1-circle-fill.svg'
%}" class="step__card__svg" alt="first step"></span>
                            <br>Зареєструйтесь в додатку</h5><hr>
                          <p class="card-text fs-5 text-wrap">Пройдіть просту
реєстрацію в додатку та налаштуйте свій власний профіль.</p>
                          <a href="{% url 'registration' %}" class="card-
link">Реєстрація</a>
                        </div>
                    </div>
                </div>
                <div class="col-xxl-4 col-md-6 col-12 align-self-center">
                    <div class="card mx-auto guide__card">
                        <div class="card-body">
                          <h5 class="card-title">
                            <span><img src="{% static 'img/svg/2-circle-fill.svg'
%}" class="step__card__svg" alt="first step"></span>
                            <br>Прив'яжіться до вашого лікаря
                            </h5><hr>
                          <p class="card-text fs-5 text-wrap">Зв'яжіться з обраним
лікарем, він згенерує ваш унікальний ключ для приєднання до нього.</p>
                          <a href="{% url 'doctors_search' %}" class="card-
link">Пошук лікарів</a>
                        </div>
                    </div>
                </div>
                <div class="col-xxl-4 col-md-6 col-12 align-self-end">
                    <div class="card mx-auto guide__card">
                        <div class="card-body">
                          <h5 class="card-title">
                            <span><img src="{% static 'img/svg/3-circle-fill.svg'
%}" class="step__card__svg" alt="first step"></span>
                            <br>Почніть вести свою власну картку
                            </h5><hr>
                          <p class="card-text fs-5 text-wrap">У свою власну картку
записуйте дані по хворобі, заміри цукру та інше. Робіть це як для себе, так і для
моніторингу лікарем.</p>
                        </div>
                    </div>
                </div>
            </div>
          </div>
        </section>
    </main>
    {% endblock %}
</body>
</html>
```

```
* {
    padding: 0px;
    margin: 0px;
    border: none;
}

*,
*::before,
*::after {
    box-sizing: border-box;
}

/* Links */

a, a:link, a:visited  {
    text-decoration: none;
}

a:hover  {
    text-decoration: none;
}

/* Common */

aside, nav, footer, header, section, main {
    display: block;
}

h1, h2, h3, h4, h5, h6, p {
    font-size: inherit;
    font-weight: inherit;
}

ul, ul li {
    list-style: none;
}

img {
    vertical-align: top;
}

img, svg {
    max-width: 100%;
    height: auto;
}

address {
  font-style: normal;
}
```

```
/* Form */

input, textarea, button, select {
    font-family: inherit;
    font-size: inherit;
    color: inherit;
    background-color: transparent;
}

input::-ms-clear {
    display: none;
}

button, input[type="submit"] {
    display: inline-block;
    box-shadow: none;
    background-color: transparent;
    background: none;
    cursor: pointer;
}

input:focus, input:active,
button:focus, button:active {
    outline: none;
}

button::-moz-focus-inner {
    padding: 0;
    border: 0;
}

label {
    cursor: pointer;
}

legend {
    display: block;
}
```

media.css

```
@media (max-width: 768px) {
    .navbar-nav {
        margin-top: 10px;
    }

    .navbar-nav .nav-item {
```

```
        margin-bottom: 10px;
        border-bottom: 1px solid black;
    }

    .nav-btn{
        margin-bottom: 10px;
    }

    .header__button{
        width: 40%;
        text-align: center;
    }

    .navbar-nav .nav-link {
        color: #333;
    }

    .promo__title{
        text-align: center;
    }

    .promo__text{
        text-align: center;
    }

    .guide__card{
        margin-bottom: 20px;
    }
}
```

index.css

```
html, body {
    height: 100%;
    margin: 0;
    padding: 0;
}

body {
    display: flex;
    flex-direction: column;
}

#section {
    flex: 1;
}

footer{
```

```
    margin-top: auto;
}

#header{
    background-color: #4169E1;
    padding: 25px 0 25px 0;
}


#logo{
    font-size: 35px;
}

.container{
    max-width: 1680px;
    margin: 0 auto;
    padding: 0 30px;
}

#logo{
    color: #fff;
}

.nav-link{
    color:#fff;
}

.nav-link:hover{
    opacity: 1;
}


.header__button{
    color: black;
    background-color: #4191E1;
}


.section__promo{
    min-height: 400px;
    background-image: url('../img/fon.png');
    background-attachment: fixed;
    background-size: contain;
}

.promo__text{
    margin-top: 20px;
    font-size: 30px;
}

.promo__button{
```

```
    padding-top: 10px;
    padding-bottom: 10px;
    width: 50%;
    margin-top: 30px;
}

.promo__img{
    height: auto;
    max-width: 100%;
}

.guide__card{
    width: 18rem;
    border: 1px solid #4169E1;
    transition: transform 0.3s ease;
}

.guide__card:hover{
    transform: translateY(-20px);
    box-shadow: 0 40px 8px 10px rgba(0, 0, 0, 0.1);
}

.step__card__svg{
    width: 45px;
    height: 45px;
    margin-bottom: 10px;
}

.func__list{
    font-size: 22px;
}

.func__item:not(:last-child){
    margin-bottom: 60px;
}

.func__item span{
    margin-right: 20px;
}




#footer{
    background-color: #2D659D;
    padding: 25px 0 25px 0;
    clear: both;
}

.footer__column{
   align-items: start;
```

```
   text-align: left;
}

.footer__link{
    font-size: 18px;
    display: inline-block;
    color: #fff;
    line-height: 1;
    text-decoration: none;
    cursor: pointer;
}
.footer__link:after {
    background-color: #4191E1;
    display: block;
    content: "";
    height: 2px;
    width: 0%;
    -webkit-transition: width .3s ease-in-out;
    -moz--transition: width .3s ease-in-out;
    transition: width .3s ease-in-out;
}
.footer__link:hover:after,
.footer__link:focus:after {
    width: 100%;
}
```

Додаток login

urls.py

```
from django.urls import path
from django.contrib.auth import views as auth_views
from . import views


urlpatterns = [
    path('registration', views.registration, name='registration'),
    path('registration/patients', views.register_patient, name='patient_register'),
    path('registration/doctors', views.register_doctor, name='doctor_register'),
    path('login', views.user_login, name='login'),
    path('reset-password', auth_views.PasswordResetView.as_view(template_name =
'login/reset-password.html'), name='password_reset'),
    path('reset-password-send',
auth_views.PasswordResetDoneView.as_view(template_name = 'login/reset-password-
send.html'), name='password_reset_done'),
```

```
    path('reset/<uidb64>/<token>',
auth_views.PasswordResetConfirmView.as_view(template_name = 'login/reset-password-
confirm.html'), name='password_reset_confirm'),
    path('reset-password-complete',
auth_views.PasswordResetCompleteView.as_view(template_name = 'login/reset-password-
complete.html'), name='password_reset_complete'),
    path('logout', auth_views.LogoutView.as_view(next_page = 'login'),
name='logout'),
]
```

models.py

```
from django.db import models
from django.contrib.auth.models import User, AbstractUser, Group
from django.utils.translation import gettext_lazy as _
from django.utils import timezone
from datetime import datetime
import uuid
from django.contrib.auth.hashers import make_password



class Adress(models.Model):
    """
    Abstract model for users and organizations adresses
    """
    country = models.CharField(max_length=50, blank=True, null=True)
    city = models.CharField(max_length=50, blank=True, null=True)
    street = models.CharField(max_length=50, blank=True, null=True)
    house_number = models.IntegerField(blank=True, null=True)
    postal_code = models.CharField(max_length=50, blank=True, null=True)

    class Meta:
        verbose_name = "Adress"
        verbose_name_plural = "Adresses"


class Organization(models.Model):
    """
    Organization model for Doctor.This is organization where doctor is currently working
    """
    organization_name = models.CharField(max_length=255, db_index=True, blank=True,
null=True)
    organization_logo = models.ImageField(upload_to='organization-logos/',blank=True, null=True)
    organization_phone = models.CharField(max_length=255, blank=True, null=True)
    organization_email = models.EmailField(_("Електронна пошта"), blank=True, null=True)
    organization_description = models.TextField(blank=True, null=True)
    organization_website_url = models.URLField(blank=True, null=True)
```

```
    adress_id = models.ForeignKey(Adress, on_delete=models.SET_NULL, db_index=True,
null=True, blank=True)

    class Meta:
        ordering = ['organization_name']
        verbose_name = "Organization"
        verbose_name_plural = "Organizations"


class DiaScreenUser(User):
    """
    Choices for sex options for user
    """
    MALE = 'Чоловіча'
    FEMALE = 'Жіноча'
    SEX_CHOICES = (
        (MALE, 'Чоловічий'),
        (FEMALE, 'Жіночий'),
    )

    phone_number = models.CharField(max_length=255, unique=True)
    date_of_birth = models.DateField()
    sex = models.CharField(max_length=20, choices=SEX_CHOICES)
    age = models.IntegerField(db_index=True, blank=True, null=True)
    patronymic = models.CharField(max_length=255, blank=True)
    avatar = models.ImageField(default='placeholder-img.png', upload_to='profile-pics/',
blank=True, null=True)

    def calculate_age(self):
        """
        Calculating a patient age by his date of birth
        """
        if self.date_of_birth.month > datetime.now().month:
            return datetime.now().year - self.date_of_birth.year - 1
        else:
            return datetime.now().year - self.date_of_birth.year

    def save(self, *args, **kwargs):
        """
        Overriding abstract user metod from django.contrib.auth.models which add additional
        functional when this model is save to database
        """
        if self.pk is None:
            self.password = make_password(self.password)
        else:
            old_instance = type(self).objects.get(pk=self.pk)
            if self.password != old_instance.password:
                self.password = make_password(self.password)

        if self.date_of_birth:
            self.age = self.calculate_age()
```

```
        super().save(*args, **kwargs)

    class Meta:
        ordering = ['last_name', 'age']
        abstract = True


class Doctor(DiaScreenUser):
    """
    Doctor model
    """
    work_experience = models.IntegerField(db_index=True)
    specialization = models.CharField(max_length=255)
    category = models.CharField(max_length=255)
    certificate_or_diploma = models.FileField(upload_to='certificates/')
    unique_connect_token = models.UUIDField(default=uuid.uuid4, editable=True, unique=True)
    about = models.TextField(blank=True, null=True)
    organization_id = models.ForeignKey(Organization, on_delete=models.SET_NULL,
db_index=True, blank=True, null=True)

    def __str__(self):
        return self.first_name + " " + self.last_name

    def save(self, *args, **kwargs):
        """
        Overriding abstract user metod from django.contrib.auth.models which add additional
        functional when this model is save to database
        """
        super().save(*args, **kwargs)
        group = Group.objects.get(name='Doctors')
        group.user_set.add(self)

    class Meta:
        ordering = ['work_experience']
        verbose_name = "Doctor"
        verbose_name_plural = "Doctors"


class Patient(DiaScreenUser):
    """
    Choices for diabet type options
    """
    FIRST_TYPE = '1'
    SECOND_TYPE = '2'
    NULL_TYPE = 'null'
    DIABETES_TYPE_CHOICES = (
        (FIRST_TYPE, '1 тип'),
        (SECOND_TYPE, '2 тип'),
        (NULL_TYPE, 'Відсутній'),
    )
```

```
    height = models.DecimalField(max_digits=6, decimal_places=2)
    weight = models.DecimalField(max_digits=6, decimal_places=2)
    body_mass_index = models.DecimalField(max_digits=6, decimal_places=2, blank=True,
null=True)
    connect_to_doctor_date = models.DateTimeField(blank=True, null=True)
    diabet_type = models.CharField(max_length=10, choices=DIABETES_TYPE_CHOICES,
default=NULL_TYPE)
    is_oninsuline = models.BooleanField(db_index=True, null=True)
    doctor_id = models.ForeignKey(Doctor, on_delete=models.SET_NULL, blank=True, null=True)
    adress_id = models.ForeignKey(Adress, on_delete=models.SET_NULL, blank=True,
db_index=True, null=True)

    def calculate_BMI(self):
        """
        Calculate body mass index for patient from his measurements
        """
        return self.weight / (self.height ** 2)


    def save(self, *args, **kwargs):
        """
        Overriding abstract user metod from django.contrib.auth.models which add additional
        functional when this model is save to database
        """
        if self.weight and self.height:
            self.body_mass_index = self.calculate_BMI()
        if self.diabet_type == self.FIRST_TYPE:
            self.is_oninsuline = True
        if self.doctor_id and not self.connect_to_doctor_date:
            self.connect_to_doctor_date = timezone.now()
        super(Patient,self).save(*args, **kwargs)
        group = Group.objects.get(name='Patients')
        group.user_set.add(self)


    def __str__(self):
        return self.first_name + " " + self.last_name

    class Meta:
        ordering = ['connect_to_doctor_date', 'diabet_type', 'is_oninsuline']
        verbose_name = "Patient"
        verbose_name_plural = "Patients"
```

forms.py

```
from django import forms
from .models import Patient, Adress, Doctor
from django.utils.translation import gettext_lazy as _
```

```
class CustomDiascreenUserValidator:
    def clean_username(form, username):
        if Patient.objects.filter(username=username).exists() or
Patient.objects.filter(username=username).exists():
            raise forms.ValidationError("Користувач з таким логіном вже існує")
        if (len(username) < 6 or len(username) > 32):
            raise forms.ValidationError("Логін занадто короткий")
        return username

class PatientForm(forms.ModelForm):
    validator = CustomDiascreenUserValidator()

    class Meta:
        model = Patient
        fields = [
            "username",
            "email",
            "password",
            "phone_number",
            "date_of_birth",
            "sex",
            "first_name",
            "last_name",
            "patronymic",
            "height",
            "weight",
            "diabet_type",
            "avatar"
        ]
        labels = {
            "username": _('Логін'),
            "email": _('Електронна пошта'),
            "password": _('Пароль'),
            "phone_number": _('Номер телефону'),
            "date_of_birth": _('Дата народження'),
            "sex": _('Стать'),
            "first_name": _('Ім`я'),
            "last_name": _('Прізвище'),
            "patronymic": _('По-батькові'),
            "height": _('Зріст'),
            "weight": _('Вага'),
            "diabet_type": _('Тип діабету'),
            "avatar": _('Аватар профілю'),
        }

    def clean_username(self):
        username = self.cleaned_data.get('username')
        return self.validator.clean_username(username)
```

```
class DoctorForm(forms.ModelForm):
    validator = CustomDiascreenUserValidator()

    class Meta:
        model = Doctor
        fields = [
            "username",
            "email",
            "password",
            "phone_number",
            "date_of_birth",
            "sex",
            "first_name",
            "last_name",
            "patronymic",
            "work_experience",
            "specialization",
            "category",
            "certificate_or_diploma",
            "about",
            "avatar"
        ]
        labels = {
            "username": _('Логін'),
            "email": _('Електронна пошта'),
            "password": _('Пароль'),
            "phone_number": _('Номер телефону'),
            "date_of_birth": _('Дата народження'),
            "sex": _('Стать'),
            "first_name": _('Ім`я'),
            "last_name": _('Прізвище'),
            "patronymic": _('По-батькові'),
            "work_experience": _('Стаж роботи'),
            "specialization": _('Спеціальность'),
            "category": _('Категорія лікаря'),
            "certificate_or_diploma": _('Сертифікат або диплом'),
            "about": _('Про себе'),
            "avatar": _('Аватар профілю'),
        }

    def clean_username(self):
        username = self.cleaned_data.get('username')
        return self.validator.clean_username(username)



class AdressForm(forms.ModelForm):
    class Meta:
        model = Adress
        fields = "__all__"
```

admin.py

```python
from django.contrib import admin
from .models import Adress, Organization, Patient, Doctor

class AdressAdmin(admin.ModelAdmin):

    list_display = [
        'country',
        'city',
        'street',
        'house_number',
        'postal_code',
    ]
    list_filter = ['country','city','street']
    search_fields = ['city','street']


class OrganizationAdmin(admin.ModelAdmin):

    list_display = [
        'organization_name',
        'organization_phone',
        'organization_email',
        'organization_website_url',
        'adress_id',
    ]
    list_filter = ['organization_name']
    search_fields = ['organization_name']


class PatientAdmin(admin.ModelAdmin):
    list_display = [
        'first_name',
        'last_name',
        'password',
        'age',
        'email',
        'avatar',
        'username',
        'height',
        'weight',
        'body_mass_index',
        'diabet_type',
        'is_oninsuline',
        'doctor_id',
        'adress_id',
    ]
```

```
    list_filter = ['diabet_type', 'is_oninsuline']
    search_fields = ['first_name', 'last_name', 'username','email']


class DoctorAdmin(admin.ModelAdmin):
    list_display = [
        'first_name',
        'last_name',
        'password',
        'email',
        'avatar',
        'username',
        'work_experience',
        'specialization',
        'category',
        'unique_connect_token',
        'organization_id',
    ]
    list_filter = ['work_experience']
    search_fields = ['first_name', 'last_name', 'username','email']


admin.site.register(Adress, AdressAdmin)
admin.site.register(Organization, OrganizationAdmin)
admin.site.register(Patient, PatientAdmin)
admin.site.register(Doctor, DoctorAdmin)
```

views.py

```
from django.http import HttpResponse
from django.shortcuts import render, redirect
from django.urls import reverse_lazy
from django.contrib.auth.decorators import login_required
from django.contrib.auth.forms import AuthenticationForm, PasswordResetForm
from django.contrib.auth import login as auth_login


from .forms import PatientForm, AdressForm, DoctorForm

def registration(request):
    return render(request, 'login/registration.html')


def register_patient(request):
    if request.method == "POST":
        patient_form = PatientForm(request.POST)
        address_form = AdressForm(request.POST)
        if patient_form.is_valid() and address_form.is_valid():
            patient = patient_form.save(commit=False)
```

```
            address = address_form.save()
            patient.address = address
            patient.save()
            return redirect('home')
        else:
            return render(request, 'login/patient_form.html', {
                'patient_form': patient_form,
                'address_form': address_form,
            })
    else:
        patient_form = PatientForm()
        address_form = AdressForm()
        return render(request, 'login/patient_form.html', {
            'patient_form': patient_form,
            'address_form': address_form,
        })


def register_doctor(request):
    if request.method == "POST":
        doctor_form = DoctorForm(request.POST, request.FILES)
        if doctor_form.is_valid():
            doctor_form.save()
            return redirect('home')
        else:
            return render(request, 'login/doctor_form.html', {'doctor_form':
doctor_form})
    else:
        doctor_form = DoctorForm()

    return render(request, 'login/doctor_form.html', {'doctor_form': doctor_form})


def user_login(request):
    if request.method == 'POST':
        form = AuthenticationForm(request, request.POST)
        if form.is_valid():
            auth_login(request, form.get_user())
            return redirect('home')
    else:
        form = AuthenticationForm()
    return render(request, 'login/login.html', {'form': form})
```

tests.py

```
from django.test import TestCase
from django.core.exceptions import ValidationError
from django.db import connection
from login.models import Patient
```

```
from datetime import datetime

class DataBaseConnectionTests(TestCase):
    def test_db_connection(self):
        try:
            cursor = connection.cursor()
        except Exception as e:
            self.fail(f"Database connection error: {str(e)}")


class AgeCalculationTests(TestCase):
    def test_valid_year_calculation(self):

        date_of_birth = datetime(year=2002, day=22, month=11)
        instance = Patient(date_of_birth=date_of_birth)

        age = instance.calculate_age()
        self.assertEqual(age, 21)


class BodyIndexCalculationTests(TestCase):
    def test_calculate_body_mass_index(self):
        height = 1.93
        weight = 85
        expected_result = weight / (height ** 2)
        instance = Patient(height=height, weight=weight)

        body_mass_index = instance.calculate_BMI()
        self.assertEqual(body_mass_index, expected_result)
```

registration.html

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>DiaScreen - реєстрація користувачів</title>
    <link rel="stylesheet" href="{% static 'css/reset.css'%}">
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH"
crossorigin="anonymous">
    <link rel="stylesheet" href="{% static 'css/registration.css'%}">
</head>
<body>
```

```
    <main>
        {% block content %}
        <section class="option__choice" id="options">
            <div class="container">
                <div class="row">
                    <div class="col-12 text-center">
                        <h1 class="fs-2">Реєстрація</h1>
                    </div>
                </div>
                <div class="row">
                    <div class="col-md-6 col-12 text-end">
                        <a class="nav-link" href="{% url 'home' %}"><span>
                            <img class="link-to-main" src="{% static 'img/arrow-
left.svg' %}" alt="Стрілка повернення на головну">
                        </span>Повернутися на головну сторінку</a>
                    </div>
                    <div class="col-md-6 col-12 text-start">
                        <a class="nav-link" href="{% url 'login' %}">В мене вже є
аккаунт
                        <span>
                            <img class="link-to-login" src="{% static 'img/arrow-
right.svg' %}" alt="Стрілка вправо для логіну">
                        </span></a>
                    </div>
                </div>
                <div class="row">
                    <div class="col-md-6">
                        <a href="{% url 'patient_register' %}" id="patient__btn"
class="btn btn-primary patient__choice">
                            Я пацієнт
                        </a>
                    </div>
                    <div class="col-md-6">
                        <a href="{% url 'doctor_register' %}" id="doctor__btn"
class="btn btn-danger doctor__choice">
                            Я лікар
                        </a>
                    </div>
                </div>
            </div>
        </section>
        {% endblock content %}
    </main>
</body>
</html>
```

patient_form.html

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Реєстрація пацієнта</title>
    <link rel="stylesheet" href="{% static 'css/reset.css'%}">
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH"
crossorigin="anonymous">
    <link rel="stylesheet" href="{% static 'css/patient.css'%}">
</head>
<body>
    <main>
        <section class="patient__form align_items-center" id="patient__form">
            <div class="container">
                <div class="row align_items-center text-center mb-3">
                    <div class="col-12 mb-0">
                        <img src="{% static 'img/arrow-up.svg' %}" alt="Стрілка
вверх">
                    </div>
                    <div class="col-12 text-center">
                        <a class="reg__link" id="linkUP_doctor" href="{% url
'registration' %}">Повернутися до вибору</a>
                    </div>
                </div>
                <div class="row align_items-center text-center mb-3">
                    {% if patient_form.errors %}
                    <div class="alert alert-danger">
                        <strong>Помилка:</strong>
                        <ul>
                        {% for field, error_list in patient_form.errors.items %}
                        {% for error in error_list %}
                            <li>{{ field }}: {{ error }}</li>
                        {% endfor %}
                        {% endfor %}
                        </ul>
                     </div>
                    {% endif %}
                </div>
                <div class="row align_items-center">
                    <div class="col-12">
                        <div class="form__background">
                            <form method="post" id="patient"
enctype="multipart/form-data" novalidate>
                                {% csrf_token %}
                                <fieldset>
```

```
                                    <div class="row">
                                        <div class="col-md-6 mb-2">
                                            <label for="username">Логін <span
class="label_span">*</span></label>
                                            <input type="text" id="username"
name="username" required>
                                            <div id="username-feedback"
class="error"></div>
                                        </div>
                                        <div class="col-md-6 mb-2">
                                            <label for="email">Електронна пошта
<span class="label_span">*</span></label>
                                            <input type="email"  id="email"
name="email" required>
                                            <div id="email-feedback"
class="error"></div>
                                        </div>
                                    </div>
                                    <div class="row">
                                        <div class="col-md-6 mb-2">
                                            <label for="password">Пароль <span
class="label_span">*</span></label>
                                            <input type="password" id="password"
name="password" required>
                                            <div id="password-feedback"
class="error"></div>
                                        </div>
                                        <div class="col-md-6 mb-2">
                                            <label for="password2">Підтвердження
пароля <span class="label_span">*</span></label>
                                            <input type="password" id="password2"
required>
                                            <div id="password1-feedback"
class="error"></div>
                                        </div>
                                    </div>
                                </fieldset>
                                <fieldset>
                                    <legend>Інформація про пацієнта</legend><hr>
                                    <div class="row">
                                        <div class="col-md-4 mb-3">
                                            <label for="last-name">Прізвище <span
class="label_span">*</span></label>
                                            <input type="text" id="last-name"
name="last_name" required>
                                            <div id="lastname-feedback"
class="error"></div>
                                        </div>
                                        <div class="col-md-4 mb-3">
```

```
                                        <label for="first_name">Ім'я <span
class="label_span">*</span></label>
                                        <input type="text" id="first-name"
name="first_name" required>
                                        <div id="firstname-feedback"
class="error"></div>
                                    </div>
                                    <div class="col-md-4 mb-3">
                                        <label for="patronymic">По-
батькові</label>
                                        <input type="text" id="patronymic"
name="patronymic">
                                        <div id="patronymic-feedback"
class="error"></div>
                                    </div>
                                </div>
                                <div class="row">
                                    <div class="col-md-4 mb-3">
                                        <label for="phone_number">Номер
телефону<span class="label_span">*</span></label>
                                        <input type="phone" id="phone_number"
name="phone_number">
                                        <div id="phone_number-feedback"
class="error"></div>
                                    </div>
                                    <div class="col-md-4 mb-3">
                                        <label for="sex">Оберіть вашу
стать<span class="label_span">*</span></label>
                                        <select id="sex" name="sex">
                                            <option
value="Male">Чоловіча</option>
                                            <option
value="Female">Жіноча</option>
                                        </select>
                                    </div>
                                    <div class="col-md-4 mb-3">
                                        <label for="date_of_birth">Введіть дату
народження<span class="label_span">*</span></label>
                                        <input type="date" id="date_of_birth"
name="date_of_birth">
                                    </div>
                                </div>
                                <div class="row">
                                    <div class="col-md-2 mb-3">
                                        <label for="height">Зріст (в
метрах)<span class="label_span">*</span></label>
                                        <input type="text" id="height"
name="height">
                                        <div id="height-feedback"
class="error"></div>
```

```
                                    </div>
                                    <div class="col-md-2 mb-3">
                                        <label for="weight">Вага (в кг)<span
class="label_span">*</span></label>
                                        <input type="text" id="weight"
name="weight">
                                        <div id="weight-feedback"
class="error"></div>
                                    </div>
                                    <div class="col-md-4 mb-3">
                                        <label for="diabet_type">Тип
діабету<span class="label_span">*</span></label>
                                        <select id="diabet_type"
name="diabet_type">
                                            <option
value="null">Відсутній</option>
                                            <option value="1">1 тип</option>
                                            <option value="2">2 тип</option>
                                        </select>
                                    </div>
                                    <div class="col-md-4 mb-3">
                                        <label for="avatar">Аватар
профілю</label>
                                        <input type="file" id="avatar"
name="avatar">
                                    </div>
                                </div>
                            </fieldset>
                            <fieldset class="adresses">
                                <legend>Адреса пацієнта ( не обов'язково до
заповнення )</legend><hr>
                                <div class="row">
                                    <div class="col-md-2 mb-3">
                                        <label for="country">Країна</label>
                                        <input type="text" id="country"
name="country">
                                        <div class="country-feedback
error"></div>
                                    </div>
                                    <div class="col-md-2 mb-3">
                                        <label for="city">Місто</label>
                                        <input type="text" class="" id="city"
name="city">
                                        <div class="city-feedback error"></div>
                                    </div>
                                    <div class="col-md-2 mb-3">
                                        <label for="country">Вулиця</label>
                                        <input type="text" id="street"
name="street">
```

```
                                        <div class="street-feedback
error"></div>
                                    </div>
                                    <div class="col-md-2 mb-3">
                                        <label for="house_number">Номер
будинку</label>
                                        <input type="text" id="house_number"
name="house_number">
                                        <div class="house-feedback
error"></div>
                                    </div>
                                    <div class="col-md-2 mb-3">
                                        <label for="postal_code">Поштовий
індекс</label>
                                        <input type="text" id="postal_code"
name="postal_code">
                                        <div class="postal-feedback
error"></div>
                                    </div>
                                </div>
                            </fieldset>
                            <div class="row align-item-center">
                                <div class="col-md-6 mb-3">
                                    <button class="btn btn-primary"
type="submit">Готово!</button>
                                    <a href="{% url 'login' %}" class="btn btn-
primary">Логін</a>
                                </div>
                                <div class="col-md-6 text-end">
                                    <a class="btn btn-primary"
href="#">Політика конфеденційності</a>
                                </div>
                            </div>
                        </form>
                    </div>
                </div>
            </div>
        </section>
    </main>
    <script src="{% static 'js/patient_validation.js' %}"></script>
</body>
</html>
```

login.html

```
{% load static %}
<!DOCTYPE html>
```

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Логін</title>
    {% block styles %}
    <link rel="stylesheet" href="{% static 'css/reset.css' %}">
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH"
crossorigin="anonymous">
    <link rel="stylesheet" href="{% static 'css/login.css' %}">
    {% endblock styles %}
</head>
<body>
    {% block content %}
    <main>
        <section id="section">
            <div class="container">
                <div class="row">
                    <div class="col-12">
                        <h2 class="text-center mb-3">Авторизація</h2>
                        <div class="form__background mb-5">
                            <form method="post">
                                <div class="form__errors">
                                    {{ form.errors }}
                                </div>
                                {% csrf_token %}
                                {{ form.as_p }}
                                <button type="submit" class="btn btn-primary
form__button">Увійти</button>
                                <a href="{% url 'registration' %}" class="btn btn-
primary form__button">Зареєструватися</a>
                                <a href="{% url 'password_reset' %}"
class="form__button">Забули пароль?</a>
                            </form>
                        </div>
                    </div>
                </div>
            </div>
        </section>
    </main>
    {% endblock content %}
</body>
</html>
```

reset-password.html

```
{% extends 'login/login.html' %}
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Зміна паролю</title>
</head>
<body>
    {% block content %}
    <main>
        <section id="section">
            <div class="container">
                <div class="row">
                    <div class="col-12">
                        <h2 class="text-center mb-3">Зміна паролю</h2>
                        <p class="text-center mb-3 fs-6">Вкажіть вашу електронну
пошту на яку буде вислано посилання на зміну паролю</p>
                        <div class="form__background mb-5">
                            <form method="post">
                                {% csrf_token %}
                                {{ form.as_p }}
                                <button type="submit" class="btn btn-
primary">Відновити пароль</button>
                                <a href="{% url 'login' %}" class="btn btn-
primary">Логін</a>
                            </form>
                        </div>
                    </div>
                </div>
            </div>
        </section>
    </main>
    {% endblock content %}
</body>
</html>
```

registration.css

```
body{
    background-color: #D2E8FF;
}

.container{
    max-width: 1680px;
    margin: 0 auto;
    padding: 0 30px;
```

```
}

.option__choice{
    height: 100vh;
    padding-top: 50px;
}

.nav-link{
    margin-top: 50px;
}

.patient__choice, .doctor__choice{
    display: flex;
    justify-content: center;
    align-items: center;
    font-size: 40px;
    margin-top: 20px;
    width: 100%;
    height: 400px;
}

.patient__form{
    margin-top: 50px;
}

.doctor__form{
    margin-top: 50px;
}

.form__background{
    background-color: rgb(255, 255, 255);
    padding: 20px;
    max-width: 1000px;
    height: auto;
    margin: 0 auto;
    border: 1px solid black;
}

label{
    display: block;
}

legend{
    font-size: 18px;
}

input,
select {
    width: 100%;
    padding: 8px;
```

```
    margin: 6px 0;
    border: 1px solid #ced4da;
    border-radius: 4px;
    box-sizing: border-box;
}

.invalid{
    border: 1px solid red;
}

.error{
    color: red;
    font-size: 16px;
}


@media (max-width: 768px) {
    .patientform__background {
        width: 100%;
        padding: 10px;
    }
}
```

login.css

```
body{
    background-color: #D2E8FF;
}

.container{
    max-width: 1680px;
    margin: 0 auto;
    padding: 0 30px;
}

#section{
    height: 100vh;
    display: flex;
    justify-content: center;
    align-items: center;
}

input,
select {
    display: block;
    width: 100%;
    padding: 8px;
    height: 40px;
```

```
    margin: 6px 0;
    border: 1px solid #ced4da;
    border-radius: 4px;
    box-sizing: border-box;
}

form{
    margin: 0 auto;
}

.form__background{
    background-color: rgb(255, 255, 255);
    padding: 20px;
    width: 650px;
    height: auto;
    margin: 0 auto;
    border: 1px solid black;
}

.form__button:not(:first-of-type){
    margin-left: 10px;
}
```

patient_validation.js

```
document.addEventListener('DOMContentLoaded', function(){

    var form = document.getElementById('patient');

    form.addEventListener('submit', function(event){
        event.preventDefault();

        if (!username_validation()
        || !email_validation()
        || !password_validation()
        || !initials_validation()
        || !phone_validator()
        || !height_weight_validator()
        || !sex_validation()
        || !date_of_birth_validation()
        || !diabet_type_validation()
        || !country_validation()
        || !city_validation()
        || !street_validation()
        || !house_number_validation()
        || !postal_code_validation()){
            return false;
        }
```

```
        form.submit();
    });

    function username_validation(){
        var username = document.getElementById('username').value;
        var usernameFeedback = document.getElementById('username-feedback');
        var username_error = document.getElementById('username');

        if (username === ''){
            usernameFeedback.innerText = 'Заповніть поле логіну';
            username_error.classList.add('invalid');
            return false;
        } else if (username.length < 6 || username.length > 32){
            usernameFeedback.innerText = 'Логін повинен бути більшим ніж 6
символів, але не більше 32';
            username_error.classList.add('invalid');
            return false;
        } else if (!isNaN(parseInt(username[0]))){
            usernameFeedback.innerText = 'Першим символом не повинно бути число';
            username_error.classList.add('invalid');
            return false;
        }
        usernameFeedback.innerText = '';
        username_error.classList.remove('invalid');
        return true;
    }

    function email_validation(){
        var email = document.getElementById('email').value;
        var emailFeedback = document.getElementById('email-feedback');
        var email_error = document.getElementById('email');

        var re = /\S+@\S+\.\S+/;
        if (!re.test(email)){
            emailFeedback.innerText = 'Некоректна адреса електронної пошти';
            email_error.classList.add('invalid');
            return false;
        }
        emailFeedback.innerText = '';
        email_error.classList.remove('invalid');
        return true;
    }

    function password_validation(){
        var password = document.getElementById('password').value;
        var password_repeat = document.getElementById('password2').value;
        var passwordError = document.getElementById('password');
        var passwordRepError = document.getElementById('password2');
        var passwordFeedback = document.getElementById('password-feedback');
```

```
        var passwordRepFeedback = document.getElementById('password1-feedback');

        if (password.length < 6 || password.length > 32){
            passwordError.classList.add('invalid');
            passwordFeedback.innerText = 'Пароль повинен бути від 6 до 32
символів';
            return false;
        } else if (password != password_repeat){
            passwordRepFeedback.innerText = 'Паролі не співпадають';
            passwordRepError.classList.add('invalid');
            return false;
        }

        passwordError.classList.remove('invalid');
        passwordRepError.classList.remove('invalid');
        passwordFeedback.innerText = '';
        passwordRepFeedback.innerText = '';
        return true;
    }

    function initials_validation(){
        var last_name = document.getElementById('last-name').value;
        var first_name = document.getElementById('first-name').value;
        var patronymic = document.getElementById('patronymic').value;
        var lastnameError = document.getElementById('last-name');
        var firstnameError = document.getElementById('first-name');
        var patronymicError = document.getElementById('patronymic');
        var lastnameFeedback = document.getElementById('lastname-feedback');
        var firstnameFeedback = document.getElementById('firstname-feedback');
        var patronymicFeedback = document.getElementById('patronymic-feedback');

        if (last_name === ''){
            lastnameError.classList.add('invalid');
            lastnameFeedback.innerText = 'Заповніть поле';
            return false;
        } else if (!checkForDigits(last_name)){
            lastnameError.classList.add('invalid');
            lastnameFeedback.innerText = 'У прізвищі не може бути цифр';
            return false;
        }

        if (first_name === ''){
            firstnameError.classList.add('invalid');
            firstnameFeedback.innerText = 'Заповніть поле';
            return false;
        } else if (!checkForDigits(first_name)){
            firstnameError.classList.add('invalid');
            firstnameFeedback.innerText = 'У імені не може бути цифр';
            return false;
        }
```

```
        if (patronymic === ''){
            patronymicError.classList.add('invalid');
            patronymicFeedbackFeedback.innerText = 'Заповніть поле';
            return false;
        } else if (!checkForDigits(patronymic)){
            patronymicError.classList.add('invalid');
            patronymicFeedback.innerText = 'У прізвищі не може бути цифр';
            return false;
        }

        firstnameError.classList.remove('invalid');
        lastnameError.classList.remove('invalid');
        patronymicError.classList.remove('invalid');
        firstnameFeedback.innerText = '';
        lastnameFeedback.innerText = '';
        patronymicFeedback.innerText = '';
        return true;
    }

    function phone_validator(){
        var phone_number = document.getElementById('phone_number').value;
        var phoneNumError = document.getElementById('phone_number');
        var phoneFeedback = document.getElementById('phone_number-feedback');
        var re = /^\d+$/;

        if (phone_number.length > 10){
            phoneNumError.classList.add('invalid');
            phoneFeedback.innerText = 'Занадто довгий номер телефону (пишіть без
+38)';
            return false;
        } else if (!re.test(phone_number)){
            phoneNumError.classList.add('invalid');
            phoneFeedback.innerText = 'Номер телефону може містити лише цифри';
            return false;
        }
        phoneNumError.classList.remove('invalid');
        phoneFeedback.innerText = '';
        return true;
    }

    function height_weight_validator(){
        var height = document.getElementById('height').value;
        var weight = document.getElementById('weight').value;
        var heightError = document.getElementById('height');
        var weightError = document.getElementById('weight');
        var heightFeedback = document.getElementById('height-feedback');
        var weightFeedback = document.getElementById('weight-feedback');

        if (height.length > 5){
```

```
            heightError.classList.add('invalid');
            heightFeedback.innerText = 'Некоректний зріст';
            return false;
        } else if (height[1] != '.'){
            var validHeight = height.substring(0,1) + '.' + height.substring(1);
            heightError.value = validHeight;
        }

        if (weight.length > 4){
            weightError.classList.add('invalid');
            weightFeedback.innerText = 'Некоректна вага';
            return false;
        }

        heightError.classList.remove('invalid');
        weightError.classList.remove('invalid');
        heightFeedback.innerText = '';
        weightFeedback.innerText = '';
        return true;
    }

    function checkForDigits(word) {
        var regex = /\d/;
        return !regex.test(word);
    }

    function sex_validation() {
        var sex = document.getElementById('sex').value;
        if (sex !== 'Male' && sex !== 'Female') {
            alert('Виберіть вашу стать');
            return false;
        }
        return true;
    }

    function date_of_birth_validation() {
        var dob = document.getElementById('date_of_birth').value;
        if (dob === '') {
            alert('Заповніть поле дати народження');
            return false;
        }
        return true;
    }

    function diabet_type_validation() {
        var diabetType = document.getElementById('diabet_type').value;
        if (diabetType === '') {
            alert('Виберіть тип діабету');
            return false;
        }
```

```
        return true;
    }

    function country_validation() {
        var country = document.getElementById('country').value;
        var countryFeedback = document.querySelector('.country-feedback');
        if (!checkForDigits(country)) {
            countryFeedback.innerText = 'Назва країни не може містити цифри';
            return false;
        }
        countryFeedback.innerText = '';
        return true;
    }

    function city_validation() {
        var city = document.getElementById('city').value;
        var cityFeedback = document.querySelector('.city-feedback');
        if (!checkForDigits(city)) {
            cityFeedback.innerText = 'Назва міста не може містити цифри';
            return false;
        }
        cityFeedback.innerText = '';
        return true;
    }

    function street_validation() {
        var street = document.getElementById('street').value;
        var streetFeedback = document.querySelector('.street-feedback');
        if (!checkForDigits(street)) {
            streetFeedback.innerText = 'Назва вулиці не може містити цифри';
            return false;
        }
        streetFeedback.innerText = '';
        return true;
    }

    function house_number_validation() {
        var houseNumber = document.getElementById('house_number').value;
        var houseFeedback = document.querySelector('.house-feedback');
        if (isNaN(houseNumber) || houseNumber === '') {
            houseFeedback.innerText = 'Некоректний номер будинку';
            return false;
        }
        houseFeedback.innerText = '';
        return true;
    }

    function postal_code_validation() {
        var postalCode = document.getElementById('postal_code').value;
        var postalFeedback = document.querySelector('.postal-feedback');
```

```
        if (isNaN(postalCode) || postalCode === '') {
            postalFeedback.innerText = 'Некоректний поштовий індекс';
            return false;
        }
        postalFeedback.innerText = '';
        return true;
    }
});
```

Додаток profiles

urls.py

```
from django.urls import path
from .views import DoctorProfile, PatientProfile, DoctorsPatientList
from .views import edit_patient_profile, edit_doctor_profile, unlink_doctor

urlpatterns = [
    path('doctor/<int:pk>', DoctorProfile.as_view(), name='doctor_profile'),
    path('patient/<int:pk>', PatientProfile.as_view(), name='patient_profile'),
    path('doctor/patients', DoctorsPatientList.as_view(), name='patient_list'),
    path('edit_patient/<int:patient_id>', edit_patient_profile,
name='edit_patient'),
    path('edit_doctor/<int:doctor_id>', edit_doctor_profile, name='edit_doctor'),
    path('patient/unlink-doctor', unlink_doctor, name='unlink_doctor'),
]
```

models.py

```
from django.db import models
from django.contrib.auth.models import User
from login.models import Doctor, Patient

class DoctorsProfileFeedback(models.Model):
    doctor_id = models.ForeignKey(Doctor, on_delete=models.CASCADE, db_index=True,
null=True, blank=True)
    patient_id = models.ForeignKey(Patient, on_delete=models.PROTECT,
db_index=True, null=True, blank=True)
    feedback_text = models.TextField(max_length=1000, null=True, blank=True)
    date_of_created = models.DateTimeField(auto_now_add=True)

    class Meta:
        unique_together = ['doctor_id', 'patient_id']
```

```
    def __str__(self):
        return f'''
        Відгук від {self.patient_id}, до лікаря {self.doctor_id}
        створений о {self.date_of_created} та містить текст {self.feedback_text}
        '''


class TechSupportTicket(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE, null=False,
blank=False)
    text = models.TextField(max_length=2000, blank=False, null=False)
    email = models.EmailField()
    img = models.ImageField(upload_to='tech-support/tickets/', blank=True,
null=True)
```

forms.py

```
from django import forms
from django.utils.translation import gettext_lazy as _
from login.models import Doctor, Patient, Adress, Organization
from .models import DoctorsProfileFeedback, TechSupportTicket

class FeedbackForm(forms.ModelForm):
    class Meta:
        model = DoctorsProfileFeedback
        fields = ['feedback_text']
        widgets = {
            'feedback_text': forms.Textarea(attrs={'maxlength': 1000, 'minlength':
20, 'required': True}),
        }


class DoctorLinkForm(forms.Form):
    unique_connect_token = forms.CharField(max_length=255, required=True)


class AdressEditForm(forms.ModelForm):
    class Meta:
        model = Adress
        fields = "__all__"
        labels = {
            'country': _('Країна'),
            'city': _('Місто'),
            'street': _('Вулиця'),
            'house_number': _('Номер будинку'),
            'postal_code': _('Поштовий індекс'),
        }
```

```
    def __init__(self, *args, **kwargs):
        super(AdressEditForm, self).__init__(*args, **kwargs)
        for field_name, field in self.fields.items():
            field.widget.attrs.update({'class': 'form-control'})


class PatientEditProfileForm(forms.ModelForm):
    class Meta:
        model = Patient
        fields = "__all__"
        exclude = ["doctor_id",
                   "adress_id",
                   "connect_to_doctor_date",
                   "body_mass_index",
                   "username",
                   "password",
                   "is_staff",
                   "is_active",
                   "is_superuser",
                   "last_login",
                   "groups",
                   "user_permissions",
                   "date_joined",
                   "id",
                   'age',]
        labels = {
            'phone_number': _('Номер телефону'),
            'date_of_birth': _('Дата народження'),
            'sex': _('Стать'),
            'age': _('Вік'),
            'patronymic': _('По-батькові'),
            'avatar': _('Аватарка'),
            'height': _('Зріст'),
            'weight': _('Вага'),
            'diabet_type': _('Тип діабету'),
            'is_oninsuline': _('Інсулінозалежність'),
        }

    def __init__(self, *args, **kwargs):
        super(PatientEditProfileForm, self).__init__(*args, **kwargs)
        for field_name, field in self.fields.items():
            field.widget.attrs.update({'class': 'form-control'})


class OrganizationForm(forms.ModelForm):
    class Meta:
        model = Organization
        fields = "__all__"
        exclude = ["adress_id",]
        labels = {
            'organization_name': _('Назва організації'),
```

```
            'organization_logo': _('Лого організації'),
            'organization_phone': _('Телефон організації'),
            'organization_email': _('Пошта організації'),
            'organization_description': _('Про організацію'),
            'organization_website_url': _('Сайт організації'),
        }

    def __init__(self, *args, **kwargs):
        super(OrganizationForm, self).__init__(*args, **kwargs)
        for field_name, field in self.fields.items():
            field.widget.attrs.update({'class': 'form-control'})


class DoctorEditProfileForm(forms.ModelForm):
    class Meta:
        model = Doctor
        fields = "__all__"
        exclude = ["organization_id",
                   "unique_connect_token",
                   "body_mass_index",
                   "username",
                   "password",
                   "is_staff",
                   "is_active",
                   "is_superuser",
                   "last_login",
                   "groups",
                   "user_permissions",
                   "date_joined",
                   "id",
                   'age',]
        labels = {
            'phone_number': _('Номер телефону'),
            'date_of_birth': _('Дата народження'),
            'sex': _('Стать'),
            'age': _('Вік'),
            'patronymic': _('По-батькові'),
            'avatar': _('Аватарка'),
            'work_experience': _('Стаж роботи'),
            'specialization': _('Спеціалізація'),
            'category': _('Категорія'),
            'about': _('Про себе'),
            'certificate_or_diploma': _('Сертифікат чи диплом'),
        }

    def __init__(self, *args, **kwargs):
        super(DoctorEditProfileForm, self).__init__(*args, **kwargs)
        for field_name, field in self.fields.items():
            field.widget.attrs.update({'class': 'form-control'})
```

```
class TechTicketForm(forms.ModelForm):
    class Meta:
        model = TechSupportTicket
        fields = "__all__"
        exclude = ["user"]
        labels = {
            "text": _('Опишіть вашу проблему'),
            "email": _('Вкажіть електронну пошту'),
            "img": _('Скріншот проблеми(за наявності)'),
        }

    def __init__(self, *args, **kwargs):
        super(TechTicketForm, self).__init__(*args, **kwargs)
        for field_name, field in self.fields.items():
            field.widget.attrs.update({'class': 'form-control'})
```

mixins.py

```
class UserRoleMixin:
    def get_user_roles(self, request):
        user = request.user
        is_patient = user.groups.filter(name='Patients').exists()
        is_doctor = user.groups.filter(name='Doctors').exists()
        return is_patient, is_doctor
```

views.py

```
from typing import Any
from django.core.paginator import Paginator
from django.http import HttpResponseForbidden
from django.db.models.query import QuerySet
from django.contrib.auth import update_session_auth_hash
from django.shortcuts import render, redirect, get_object_or_404
from django.views import generic
from login.models import Doctor, Patient, Adress, Organization
from login.forms import PatientForm, AdressForm
from .models import DoctorsProfileFeedback
from .forms import FeedbackForm, DoctorLinkForm, AdressEditForm,
PatientEditProfileForm, OrganizationForm, DoctorEditProfileForm
from .forms import TechTicketForm
from .mixins import UserRoleMixin


class DoctorProfile(generic.DetailView, UserRoleMixin):
    model = Doctor
    template_name = 'doctor_profile.html'
```

```
    context_object_name = 'doctor'
    paginate_by = 5

    def get_context_data(self, **kwargs):
        self.object = self.get_object()
        context = super().get_context_data(**kwargs)
        is_patient, is_doctor = self.get_user_roles(self.request)
        context['is_patient'] = is_patient
        context['is_doctor'] = is_doctor
        context['feedback_form'] = FeedbackForm()

        doctor_id = self.object.pk
        feedbacks = DoctorsProfileFeedback.objects.filter(doctor_id=doctor_id)
        paginator = Paginator(feedbacks, self.paginate_by)
        page_number = self.request.GET.get('page')
        page_object = paginator.get_page(page_number)
        context['feedback_list'] = page_object
        return context

    def get_object(self, queryset=None):
        if not hasattr(self, 'object'):
            self.object = super().get_object(queryset)
        return self.object


    def post(self, request, *args, **kwargs):
        self.object = self.get_object()
        feedback_form = FeedbackForm(request.POST)
        if feedback_form.is_valid():
            feedback = feedback_form.save(commit=False)
            feedback.doctor_id = self.object
            patient = Patient.objects.get(pk = request.user.id)
            feedback.patient_id = patient
            feedback.save()
            return redirect('doctor_profile', pk=self.object.pk)
        return
self.render_to_response(self.get_context_data(feedback_form=feedback_form))



class PatientProfile(generic.DetailView, UserRoleMixin):
    model = Patient
    template_name = 'patient_profile.html'
    context_object_name = 'patient'

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        is_patient, is_doctor = self.get_user_roles(self.request)
        context['is_patient'] = is_patient
        context['is_doctor'] = is_doctor
```

```
        context['link_form'] = DoctorLinkForm()
        return context

    def post(self, request, *args, **kwargs):
        link_form = DoctorLinkForm(request.POST)
        if link_form.is_valid():
            token = link_form.cleaned_data['unique_connect_token']
            doctor = get_object_or_404(Doctor, unique_connect_token=token)
            patient = self.get_object()
            patient.doctor_id = doctor
            patient.save(update_fields=['doctor_id', 'connect_to_doctor_date'])
            return redirect('patient_profile', pk=patient.id)
        return render(request, 'patient_profile.html', {'link_form': link_form})

    def get_object(self, queryset=None):
        if not hasattr(self, 'object'):
            self.object = super().get_object(queryset)
        return self.object


class DoctorsPatientList(generic.ListView, UserRoleMixin):
    model = Patient
    paginate_by = 3
    template_name = 'doctor_patients_list.html'
    context_object_name = 'patient_list'

    def get_queryset(self):
        doctor = self.request.user
        queryset = Patient.objects.filter(doctor_id = doctor)
        return queryset

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        is_patient, is_doctor = self.get_user_roles(self.request)
        context['is_patient'] = is_patient
        context['is_doctor'] = is_doctor
        return context


def edit_patient_profile(request, patient_id):

    if (patient_id == request.user.id):
        patient = get_object_or_404(Patient, id=patient_id)
        address = patient.adress_id
    else:
        return HttpResponseForbidden('У вас немає відповідного доступу')

    if request.method == 'POST':
        patient_form = PatientEditProfileForm(request.POST, request.FILES,
instance=patient)
```

```
        address_form = AdressEditForm(request.POST, instance=address)

        if patient_form.is_valid() and address_form.is_valid():
            patient = patient_form.save(commit=False)
            address = address_form.save()
            patient.adress_id = address
            patient.save()
            return redirect('patient_profile', pk=patient.id)
    else:
        patient_form = PatientEditProfileForm(instance=patient)
        address_form = AdressEditForm(instance=address)

    return render(request, 'profiles/edit_patient_profile.html', {
        'patient_form': patient_form,
        'address_form': address_form,
        'patient': patient
    })


def edit_doctor_profile(request, doctor_id):

    doctor = get_object_or_404(Doctor, id=doctor_id)

    if doctor_id != request.user.id:
        return HttpResponseForbidden('У вас немає відповідного доступу')
    organization = doctor.organization_id
    if organization is None:
        organization = Organization()
        address = Adress()
    else:
        address = organization.adress_id

    if request.method == 'POST':
        doctor_form = DoctorEditProfileForm(request.POST, request.FILES,
instance=doctor)
        org_form = OrganizationForm(request.POST, request.FILES,
instance=organization)
        address_form = AdressEditForm(request.POST, instance=address)

        if doctor_form.is_valid() and address_form.is_valid() and
org_form.is_valid():
            doctor = doctor_form.save(commit=False)
            org = org_form.save(commit=False)
            address = address_form.save()
            org.adress_id = address
            org.save()
            doctor.organization_id = org
            doctor.save()
            return redirect('doctor_profile', pk=doctor.id)
    else:
        doctor_form = DoctorEditProfileForm(instance=doctor)
```

```
        org_form = OrganizationForm(instance=organization)
        address_form = AdressEditForm(instance=address)

    return render(request, 'profiles/edit_doctor_profile.html', {
        'doctor_form': doctor_form,
        'org_form': org_form,
        'address_form': address_form,
        'doctor': doctor
    })


def unlink_doctor(request):
    patient = get_object_or_404(Patient, id=request.user.id)
    if request.method == 'POST':
        patient.doctor_id = None
        patient.connect_to_doctor_date = None
        patient.save()
        return redirect('patient_profile', pk = patient.id)
    return redirect('patient_profile', pk = patient.id)


def send_support_ticket(request):
    user = request.user
    tech_form = TechTicketForm()
    if request.method == "POST":
        tech_form = TechTicketForm(request.POST, request.FILES)
        if tech_form.is_valid():
            ticket = tech_form.save(commit=False)
            ticket.user = user
            ticket.save()
            return redirect('support')
    else:
        return render(request, 'tech_support.html', {'tech_form': tech_form})
```

patient_profile.html

```
{% extends 'base.html' %}
{% load static %}


{% block styles %}
<link rel="stylesheet" href="{% static 'css/profile.css' %}">
{% endblock styles %}


{% block content %}

<section id="section" class="my-3">
    <div class="container">
```

```
        <div class="row">
            <div class="col-12 border border-primary">
                <div class="row">
                    <div class="col-md-4">
                        <img src="{{ patient.avatar.url }}" class="profile-img"
alt="аватар пацієнта">
                    </div>
                    <div class="col-md-8">
                        <div class="row align-items-center my-3">
                            <div class="col-md-8">
                                <h2>{{ patient.last_name }} {{ patient.first_name
}}  {{ patient.patronymic }}</h2>
                            </div>
                            {% if request.user.id == patient.id %}
                            <div class="col-md-4 text-end">
                                <a href="{% url 'edit_patient'
patient_id=patient.id %}" class="btn btn-primary">Редагувати профіль</a>
                            </div>
                            {% endif %}
                        </div>
                        <p class="fs-5"><span class="fw-semibold">Тип
діабету:</span> {{ patient.diabet_type }}</p>
                        <p class="fs-5"><span class="fw-semibold">Стать:</span> {{
patient.sex }}</p>
                        <p class="fs-5"><span class="fw-semibold">Вік:</span> {{
patient.age }} років</p>
                        <p class="fs-5"><span class="fw-semibold">Номер
телефону:</span> {{ patient.phone_number }}</p>
                        <p class="fs-5"><span class="fw-semibold">Електронна
пошта:</span> {{ patient.email }}</p>
                        <div class="row btn__row my-3">
                            <div class="col-md-6">
                                <a href="{% url 'patient_card'
patient_id=patient.id %}" class="btn btn-primary">Картка пацієнта</a>
                                {% if is_doctor and request.user.id ==
patient.doctor_id.id %}
                                <a href="{% url 'chat' sender_id=request.user.id
receiver_id=patient.id %}" class="btn btn-primary">Написати пацієнту</a>
                                {% endif %}
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
</section>
<section id="section" class="my-3 doctor__section"></section>
    <div class="container">
        <h2 class="text-center my-3">Ваш лікар-ендокринолог</h2><hr>
```

```
        <div class="row">
            {% if patient.doctor_id is None %}
            <div class="col-12">
                <div class="row">
                    <div class="col-12">
                        <h5 class="text-center">У вас ще немає прив'язаного
лікаря</h5>
                    </div>
                    <div class="row">
                        <div class="col-md-6 mx-auto">
                            <form method="post">
                                {% csrf_token %}
                                <div class="input-group mb-3">
                                    <input type="text" class="form-control"
placeholder="Введіть код лікаря" aria-label="Введіть код лікаря" aria-
describedby="button-addon2" name="unique_connect_token" required>
                                    <button class="btn btn-primary" type="submit"
id="button-addon2">Прив'язати лікаря</button>
                                </div>
                            </form>
                        </div>
                    </div>
                </div>
            </div>
            {% else %}
            <div class="col-12 mb-3">
                <div class="row">
                    <div class="col-md-4">
                        <img src="{{ patient.doctor_id.avatar.url }}"
class="profile__img" alt="аватар доктора">
                    </div>
                    <div class="col-md-8">
                        <div class="row align-items-center my-3">
                            <div class="col-md-8">
                                <h2>{{ patient.doctor_id.last_name }} {{
patient.doctor_id.first_name }}  {{ patient.doctor_id.patronymic }}</h2>
                            </div>
                            {% if is_patient and patient.id == request.user.id %}
                            <div class="col-md-4 text-end">
                                <form action="{% url 'unlink_doctor' %}"
method="post" novalidate>
                                    {% csrf_token %}
                                    <button type="submit" class="btn btn-
danger">Відв'язати лікаря</button>
                                </form>
                            </div>
                            {% endif %}
                        </div>
                        <p class="fs-5"><span class="fw-semibold">Категорія:</span>
{{ patient.doctor_id.category }}</p>
```

```
                        <p class="fs-5"><span class="fw-semibold">Стаж
лікаря:</span> {{ patient.doctor_id.work_experience }}</p>
                        <p class="fs-5"><span class="fw-semibold">Номер
телефону:</span> {{ patient.doctor_id.phone_number }}</p>
                        <p class="fs-5"><span class="fw-semibold">Електронна
пошта:</span> {{ patient.doctor_id.email }}</p>
                        <div class="row my-5">
                            <div class="col-auto">
                                <a href="{% url 'doctor_profile'
pk=patient.doctor_id.id %}" class="btn btn-primary">Профіль лікаря</a>
                                {% if is_patient %}
                                <a href="{% url 'chat' sender_id=request.user.id
receiver_id=patient.doctor_id.id %}" class="btn btn-primary">Написати лікарю</a>
                                {% endif %}
                            </div>
                        </div>
                    </div>
                </div>
            </div>
            {% endif %}
        </div>
    </div>
{% endblock content %}
```

doctor_profile.html

```
{% extends 'base.html' %}
{% load static %}


{% block styles %}
<link rel="stylesheet" href="{% static 'css/profile.css' %}">
<script src="{% static 'js/doctor_profile.js' %}"></script>
{% endblock styles %}

{% block content %}

<section id="section" class="my-3">
    <div class="container">
        <div class="row">
            {% if is_doctor and user.id == doctor.id %}
            <div class="col-md-2 border">
                <aside>
                    <ul class="list-group">
                        <li class="list-group-item active"><a href="{% url
'doctor_profile' pk=doctor.pk %}" class="profile-link">Ваш
профіль</a></li>
```

```
                        <li class="list-group-item"><a href="{% url 'patient_list'
%}" class="profile-link">Пацієнти</a></li>
                    </ul>
                </aside>
            </div>
            {% endif %}
            <div class="col-md-10 border">
                <div class="row">
                    <div class="col-md-4">
                        <img src="{{ doctor.avatar.url }}" class="profile-img"
alt="аватар доктора">
                    </div>
                    <div class="col-md-8">
                        <div class="row align-items-center my-3">
                            <div class="col-md-8">
                                <h2>{{ doctor.last_name }} {{ doctor.first_name
}}  {{ doctor.patronymic }}</h2>
                            </div>
                            <div class="col-md-4">
                                {% if is_doctor and user.id == doctor.id %}
                                <a href="{% url 'edit_doctor' doctor_id=doctor.id
%}" class="btn btn-primary">Редагувати</a>
                                {% endif %}
                                <div class="hidden-text">
                                    {{doctor.unique_connect_token}}
                                </div>
                                <button onclick="copyHiddenText()" class="btn btn-
primary">Отримати код лікаря</button>
                            </div>
                        </div>
                        <p class="fs-5"><span class="fw-semibold">Категорія:</span>
{{ doctor.category }}</p>
                        <p class="fs-5"><span class="fw-semibold">Стаж
лікаря:</span> {{ doctor.work_experience }}</p>
                        {% if not doctor.organization_id is None and not
doctor.organization_id.organization_name is None %}
                        <div class="row">
                            <h2 class="my-2 mb-3">Організація лікаря</h2>
                                <div class="row">
                                    <div class="col-md-3">
                                        {% if
doctor.organization_id.organization_logo %}
                                        <img src="{{
doctor.organization_id.organization_logo.url }}" class="org-img" alt="лого
організації">
                                        <p class="text-center">{{
doctor.organization_id.organization_name}}</p>
                                        {% else %}
                                        <img src="{% static 'img/fon.png' %}"
class="org-img" alt="лого організації">
```

```
                                    {% endif %}
                                </div>
                                <div class="col-md-9">
                                    <p><span class="fs-5 fw-semibold">Номер
організації:</span> {{ doctor.organization_id.organization_phone}}</p>
                                    <p><span class="fs-5 fw-
semibold">Електронна пошта організації:</span> {{
doctor.organization_id.organization_email}}</p>
                                    <p><span class="fs-5 fw-semibold">Сайт
організації:</span><a href="{{ doctor.organization_id.organization_website_url }}">
{{ doctor.organization_id.organization_website_url}}</a></p>
                                    <p><span class="fs-5 fw-semibold">Опис
організації:</span> {{ doctor.organization_id.organization_description}}</p>
                                    <p><span class="fs-5 fw-semibold">Адреса
організації:</span>
                                        {{
doctor.organization_id.adress_id.country}}/{{
doctor.organization_id.adress_id.city}},
                                        {{
doctor.organization_id.adress_id.street}},
                                        буд. {{
doctor.organization_id.adress_id.house_number}},
                                        {{
doctor.organization_id.adress_id.postal_code}}
                                    </p>
                                </div>
                            </div>
                        </div>
                        {% else %}
                            <div class="row">
                                <h2 class="my-2 mb-3">Організація лікаря</h2>
                                <div class="row">
                                    {% if is_patient %}
                                    <p>У лікаря ще не вказана організація</p>
                                    {% else %}
                                    <p>У вас ще не вказана організація. Вказати
можна тут: <a class="btn btn-primary" href="{% url 'edit_doctor'
doctor_id=doctor.id %}">Редагування профілю</a></p>
                                    {% endif %}
                                </div>
                            </div>
                        {% endif %}
                        <div class="row">
                            {% if is_patient %}
                                <a href="{% url 'chat' sender_id=request.user.id
receiver_id=doctor.id %}" class="btn btn-primary">Написати лікарю</a>
                            {% endif %}
                        </div>
                    </div>
                </div>
```

```
                <div class="row">
                    <div class="col-12">
                        <h2 class="my-2 mb-3 fs-1">Інформація про лікаря</h2>
                        <p class="fs-4">{{ doctor.about }}</p>
                    </div>
                </div>
            </div>
        </div>
    </div>
</section>
<section id="section" class="my-3 feedback-section">
    <div class="container">
        <h2 class="text-center">Відгуки про лікаря</h2>
        <div class="row">
            <div class="col-12">
                {% for feedback in feedback_list %}
                    <div class="card mb-3">
                        <div class="card-body">
                            <p class="fs-5"><strong>{{
feedback.patient_id.first_name }} {{ feedback.patient_id.last_name }}</strong></p>
                            <p>{{ feedback.feedback_text }}</p>
                        </div>
                    </div>
                {% empty %}
                    <p>У лікаря ще немає відгуків</p>
                {% endfor %}
                {% if feedback_list.has_other_pages %}
                    <nav aria-label="Feedback pagination">
                        <ul class="pagination justify-content-center">
                            {% if feedback_list.has_previous %}
                                <li class="page-item">
                                    <a class="page-link" href="?page=1">&laquo;
Перша</a>
                                </li>
                                <li class="page-item">
                                    <a class="page-link" href="?page={{
feedback_list.previous_page_number }}">Попередня</a>
                                </li>
                            {% endif %}
                            {% if feedback_list.has_next %}
                                <li class="page-item">
                                    <a class="page-link" href="?page={{
feedback_list.next_page_number }}">Наступна</a>
                                </li>
                                <li class="page-item">
                                    <a class="page-link" href="?page={{
feedback_list.paginator.num_pages }}">Остання &raquo;</a>
                                </li>
                            {% endif %}
                        </ul>
```

```
                    </nav>
                {% endif %}
            </div>
        </div>
    </div>
</section>
{% if user.is_authenticated and not is_doctor %}
<section id="section" class="my-3 feedback-form-section">
    <div class="container">
        <h2 class="text-center">Залиште свій власний відгук</h2>
        <div class="row">
            <div class="col-12">
                <form action="" method="post">
                    {% csrf_token %}
                    <div class="mb-3">
                        <label for="exampleFormControlTextarea1" class="form-
label">Залиште відгук про лікаря тут</label>
                        <textarea class="form-control"
id="exampleFormControlTextarea1" rows="3" name="feedback_text"></textarea>
                      </div>
                    <button type="submit" class="btn btn-primary">Відправити
відгук</button>
                </form>
            </div>
        </div>
    </div>
</section>
{% endif %}
{% endblock content %}
```

edit_patient_profile.html

```
{% extends 'base.html' %}
{% load static %}


{% block styles %}

{% endblock styles %}

{% block content %}
<section id="section" class="my-3">
    <div class="container">
        <div class="row">
            <div class="col-md-12">
                <form method="post" enctype="multipart/form-data">
                    {% csrf_token %}
                    {{ patient_form.as_p }}
```

```
                    {{ address_form.as_p }}
                    <button type="submit" class="btn btn-primary">Зберегти
зміни</button>
                </form>
            </div>
        </div>
    </div>
</section>

{% endblock content %}
```

profiles.css

```
html, body {
    height: 100%;
    margin: 0;
    padding: 0;
}

body {
    display: flex;
    flex-direction: column;
}


#section {
    flex: 1;
}


.profile-link{
    text-decoration: none;
    color: black;
}

.profile__img{
    width: 400px;
    height: 400px;
}

footer {
    margin-top: auto;
}

.org-img {
    width: 200px;
    height: auto;
}
```

```
.feedback-form-section input[type="text"],
.feedback-form-section textarea {
    width: 100%;
    padding: 10px;
    margin-bottom: 15px;
    border: 1px solid #ccc;
    border-radius: 5px;
}

.hidden-text{
    display: none;
}

.two-auth_block{
    padding: 20px;
    background-color: rgb(245, 82, 82);
    opacity: 0.9;
}

.two-auth_block p{
    font-weight: bold;
    font-size: 18px;
}
```

Додаток doctor_search

forms.py

```
from django import forms



class DoctorSearchForm(forms.Form):
    search_input = forms.CharField(
        max_length=255,
        required=False,
        label='Пошук',
        widget=forms.TextInput(attrs={'placeholder': 'Введіть ім`я, прізвище або
по-батькові лікаря'}))


class DoctorSearchModelForm(forms.Form):
    SEX_CHOICES = [
        ('Male', 'Чоловічий'),
        ('Female', 'Жіночий'),
```

```
    ]

    CATEGORY_CHOICES = [
        ('Вища категорія', 'Вища категорія'),
        ('Перша категорія', 'Перша категорія'),
        ('Друга категорія', 'Друга категорія'),
    ]

    sex = forms.ChoiceField(choices=SEX_CHOICES, required=False, label='Стать лікаря', widget=forms.RadioSelect)
    category = forms.ChoiceField(choices=CATEGORY_CHOICES, required=False, label='Категорія', widget=forms.RadioSelect)
    work_experience = forms.CharField(required=False, label='Стаж роботи (Мінімальний)', widget=forms.NumberInput)
```

views.py

```
from typing import Any
from django.db.models import Q
from django.shortcuts import render
from django.views.generic.list import ListView
from login.models import Doctor
from .forms import DoctorSearchModelForm, DoctorSearchForm

# Create your views here.
class DoctorSearchView(ListView):
    model = Doctor
    paginate_by = 4

    def get_context_data(self, **kwargs: Any):
        context = super().get_context_data(**kwargs)
        context['filter_form'] = DoctorSearchModelForm()
        context['search_form'] = DoctorSearchForm()
        return context

    def get_queryset(self):
        queryset = super().get_queryset()
        form = DoctorSearchModelForm(self.request.GET)
        search_form = DoctorSearchForm(self.request.GET)

        if search_form.is_valid():
            spivpa = search_form.cleaned_data.get('search_input')
            if spivpa:
                queryset = queryset.filter(
                Q(first_name__icontains=spivpa) |
                Q(last_name__icontains=spivpa) |
                Q(patronymic__icontains=spivpa)
            )
```

```
        if form.is_valid():
            sex = form.cleaned_data.get('sex')
            category = form.cleaned_data.get('category')
            work_experience = form.cleaned_data.get('work_experience')
            if sex:
                queryset = queryset.filter(sex = sex)
            if category:
                queryset =  queryset.filter(category = category)
            if work_experience:
                queryset = queryset.filter(work_experience__gte = work_experience)

        return queryset
```

doctor_list.html

```
{% extends 'base.html' %}
{% load static %}

{% block styles %}
<link rel="stylesheet" href="{% static 'css/search.css' %}">
{% endblock styles %}

{% block content %}
<section id="section"  class="my-3">
    <div class="container">
        <div class="row">
            <div class="col-md-3">
              <aside>
                  <form method="get" class="search__form">
                    {{ search_form.as_p }}
                  <button type="submit" class="btn btn-primary
submit_filter_btn">Пошук</button>
                  </form>
                  <h2 class="fs-4 filter__title">Розширені фільтри пошуку</h2>
                  <form method="get" class="filter__form" novalidate>
                     {{ filter_form.as_p }}
                     <button type="submit" class="btn btn-primary
submit_filter_btn">Примінити фільтри</button>
                  </form>
              </aside>
            </div>
            <div class="col-md-9">
              <div class="row" id="doctor_row">
                  {% for doctor in object_list %}
                  <div class="card mx-4 mb-3" style="max-width: 540px;">
                    <a href="{% url 'doctor_profile' pk=doctor.pk %}"
class="doctor-link"><div class="row g-0">
```

```
                    <div class="col-md-4">
                      <img src="{{ doctor.avatar.url }}" class="img-fluid"
alt="аватар доктора">
                    </div>
                    <div class="col-md-8">
                      <div class="card-body">
                        <h5 class="card-title">{{ doctor.get_full_name }}</h5>
                        <p class="card-text">{{ doctor.specialization }}</p>
                        <p class="card-text"><small class="text-body-
secondary">{{ doctor.age }}</small></p>
                      </div>
                    </div>
                  </div></a>
                </div>
                {% if forloop.counter|divisibleby:2 and not forloop.last %}
              </div><div class="row">
              {% endif %}
              {% endfor %}
            </div>
            <div class="row">
              <div class="col-12 mx-auto text-center fs-4">
                  <span class="step-links">
                      {% if page_obj.has_previous %}
                          <a href="?page=1">&laquo; first</a>
                          <a href="?page={{ page_obj.previous_page_number
}}">previous</a>
                      {% endif %}

                      <span class="current">
                          Page {{ page_obj.number }} of {{
page_obj.paginator.num_pages }}.
                      </span>

                      {% if page_obj.has_next %}
                          <a href="?page={{ page_obj.next_page_number }}">next</a>
                          <a href="?page={{ page_obj.paginator.num_pages }}">last
&raquo;</a>
                      {% endif %}
                  </span>
              </div>
          </div>
    </div>
    </div>
</section>

{% endblock content %}
```

search.css

```
html, body {
    height: 100%;
    margin: 0;
    padding: 0;
}

body {
    display: flex;
    flex-direction: column;
}

#section {
    flex: 1;
}

footer {
    margin-top: auto;
}

input,
select {
    display: block;
    width: 100%;
    padding: 8px;
    height: 40px;
    margin: 6px 0;
    border: 1px solid #ced4da;
    border-radius: 4px;
    box-sizing: border-box;
}

input[type="radio"] {
    display: inline-block;
    appearance: none;
    -webkit-appearance: none;
    -moz-appearance: none;
    width: 20px;
    height: 20px;
    border: 2px solid #ccc;
    border-radius: 50%;
    outline: none;
    cursor: pointer;
    vertical-align: middle;
    margin-right: 5px;
  }


  input[type="radio"]:checked {
    background-color: #007bff;
    border-color: #007bff;
```

```
  }


  .filter__form{
    margin-top: 20px;
  }

  .filter__title{
    margin-top: 10px;
  }

  .doctor-link{
    text-decoration: none;
    color: black;
  }

  .card a {
    color: inherit;
    text-decoration: none;
}

  .card a:hover {
    color: inherit;
  }
```

Додаток blog

urls.py

```
from django.urls import path
from .views import InformationPanel, ArticleDetails, delete_article, edit_article

urlpatterns = [
    path('', InformationPanel.as_view(), name='information'),
    path('article/<int:pk>', ArticleDetails.as_view(), name='article_detail'),
    path('delete-article/<int:article_id>', delete_article, name='delete_article'),
    path('edit-article/<int:article_id>/', edit_article, name='edit_article'),
]
```

models.py

```
from django.db import models
```

```
class Article(models.Model):
    article_name = models.CharField(max_length=500, blank=False, null=False,
unique=True)
    article_short_name = models.CharField(max_length=50, blank=True, null=True,
unique=True)
    article_author = models.CharField(max_length=255, blank=True, null=True)
    article_description = models.CharField(max_length=500, blank=True, null=True)
    article_img = models.ImageField(upload_to='articles-img/')
    article_text = models.TextField(max_length=6000, blank=False, null=False)
    date_of_creation = models.DateField(auto_now_add=True, db_index=True)

    def get_image_url(self):
        return self.article_img.url if self.article_img else None

    def __str__(self):
        return self.article_name

    class Meta:
        verbose_name = 'Article'
        verbose_name_plural = 'Articles'
        ordering = ['-date_of_creation']
```

views.py

```
from django.shortcuts import render, redirect, get_object_or_404
from django.views.decorators.http import require_POST
from typing import Any
from django.views import generic
from .mixins import AdminRoleMixin
from .models import Article
from .forms import ArticleCreationForm

class InformationPanel(generic.ListView, AdminRoleMixin):
    model = Article
    template_name = 'info_panel.html'
    context_object_name = 'article_list'
    paginate_by = 9

    def get_context_data(self, **kwargs: Any):
        context = super().get_context_data(**kwargs)
        context['article_creation_form'] = ArticleCreationForm()
        context['is_admin'] = self.get_admin_role(self.request)
        return context

    def post(self, request, *args, **kwargs):
        article_form = ArticleCreationForm(request.POST, request.FILES)
        if article_form.is_valid():
            article_form.save()
```

```
            return redirect('information')
        return redirect('information')


class ArticleDetails(generic.DetailView, AdminRoleMixin):
    model = Article
    template_name = 'article_detail.html'
    context_object_name = 'article'

    def split_text(self):
        article = self.get_object()
        paragraphs = article.article_text.split('\n')
        return paragraphs

    def get_context_data(self, **kwargs: Any):
        context = super().get_context_data(**kwargs)
        context['article_form'] = ArticleCreationForm(instance=self.get_object())
        context['paragraphs'] = self.split_text()
        context['is_admin'] = self.get_admin_role(self.request)
        return context


@require_POST
def delete_article(request, article_id):
    article = get_object_or_404(Article, id=article_id)
    article.delete()
    return redirect('information')


def edit_article(request, article_id):
    article = get_object_or_404(Article, id=article_id)
    if request.method == 'POST':
        article_form = ArticleCreationForm(request.POST, request.FILES,
instance=article)
        if article_form.is_valid():
            article_form.save()
            return redirect('article_detail', pk=article_id)
    else:
        article_form = ArticleCreationForm(instance=article)
```

info_panel.html

```
{% extends 'base.html' %}
{% load static %}


{% block styles %}
<link rel="stylesheet" href="{% static 'css/info.css' %}">
{% endblock styles %}
```

```
{% block content %}
<section id="section" class="my-4">
    <div class="container">
        {% if is_admin %}
        <div class="row my-3">
            <div class="col-12 text-end">
                <a href="#article_creation" class="btn btn-primary px-5 mx-5" data-
bs-toggle="modal" data-bs-target="#article_creation">Додати нову статтю</a>
            </div>
            <div class="modal fade" id="article_creation" tabindex="-1" aria-
labelledby="exampleModalLabel" aria-hidden="true">
                <div class="modal-dialog modal-dialog-centered modal-dialog-
scrollable modal-lg">
                    <div class="modal-content">
                        <div class="modal-header">
                            <h5 class="modal-title"
id="article_creation_label">Додати нову статтю</h5>
                            <button type="button" class="btn-close" data-bs-
dismiss="modal" aria-label="Close"></button>
                        </div>
                        <div class="modal-body">
                            <form method="post" enctype="multipart/form-data">
                                {% csrf_token %}
                                {{ article_creation_form.as_p }}
                                <button type="submit" class="btn btn-
primary">Зберегти</button>
                            </form>
                        </div>
                    </div>
                </div>
            </div>
        </div>
        {% endif %}
        <div class="row card_container">
            {% for article in article_list %}
            <div class="col-md-4">
                <div class="card mx-auto" style="width: 27rem;">
                    <img src="{{ article.get_image_url }}" class="card-img-top"
alt="картинка статті">
                    <div class="card-body">
                        <h5 class="fs-5">{{ article.article_short_name }}</h5>
                        <p>Автор статті: {{ article.article_author }}</p>
                        <p>{{ article.article_description }}</p>
                        <div class="row">
                            <div class="col-md-9">
                                <a href="{% url 'article_detail' pk=article.id %}"
class="btn btn-primary card_btn">Перейти до статті</a>
                            </div>
                        </div>
```

```
                    </div>
                </div>
            </div>
            {% if forloop.counter|divisibleby:2 and not forloop.last %}
        </div><div class="row">
        {% endif %}
        {% endfor %}
    </div>
    <div class="row my-5">
        <div class="col-12 mx-auto text-center fs-4">
            <span class="step-links">
                {% if page_obj.has_previous %}
                    <a href="?page=1">&laquo; first</a>
                    <a href="?page={{ page_obj.previous_page_number
}}">previous</a>
                {% endif %}

                <span class="current">
                    Page {{ page_obj.number }} of {{ page_obj.paginator.num_pages
}}.
                </span>

                {% if page_obj.has_next %}
                    <a href="?page={{ page_obj.next_page_number }}">next</a>
                    <a href="?page={{ page_obj.paginator.num_pages }}">last
&raquo;</a>
                {% endif %}
            </span>
        </div>
    </div>
</section>
{% endblock content %}
```

info.css

```
.modal-body input[type="text"],
.modal-body textarea {
    width: 100%;
    padding: 10px;
    margin-bottom: 15px;
    border: 1px solid #ccc;
    border-radius: 5px;
}

.card{
    height: 100%;
    overflow: hidden;
    transition: transform 0.3s ease;
```

```
}

.card:hover{
    transform: translateY(-15px);
    box-shadow: 0 20px 8px 10px rgba(0, 0, 0, 0.1);
}
```

Додаток patient_card

urls.py

```
from django.urls import path
from .views import GlucoseMeasurmentView, PhysicalMeasurementView,
FoodMeasurementView, InsulineDoseView, AnalysisLoadView, download_pdf

urlpatterns = [
    path('<int:patient_id>', GlucoseMeasurmentView.as_view(), name='patient_card'),
    path('physics/<int:patient_id>', PhysicalMeasurementView.as_view(),
name='patient_physics'),
    path('food/<int:patient_id>', FoodMeasurementView.as_view(),
name='patient_food'),
    path('insuline-therapy/<int:patient_id>', InsulineDoseView.as_view(),
name='patient_insuline'),
    path('analysis/<int:patient_id>', AnalysisLoadView.as_view(),
name='patient_analysis'),
    path('download-pdf/<int:patient_id>', download_pdf, name='analysis_download')
]
```

models.py

```
from typing import Iterable
from django.utils import timezone
from django.db import models
from login.models import Patient


class GlucoseMeasurement(models.Model):

    CATEGORY = (
        ('Натщесердце', 'Натщесердце'),
        ('До сніданку', 'До сніданку'),
```

```
        ('Після сніданку', 'Після сніданку'),
        ('Перед перекусом', 'Перед перекусом'),
        ('Після перекусу', 'Після перекусу'),
        ('До обіду', 'До обіду'),
        ('Після обіду', 'Після обіду'),
        ('До вечері', 'До вечері'),
        ('Після вечері', 'Після вечері'),
        ('Перед сном', 'Перед сном'),
        ('Інше', 'Інше'),
    )

    glucose = models.DecimalField(max_digits=4, decimal_places=2, blank=False,
null=False)
    glucose_measurement_category = models.CharField(choices=CATEGORY, blank=True,
null=True)
    date_of_measurement = models.DateField()
    time_of_measurement = models.TimeField(default=timezone.now)
    patient_id = models.ForeignKey(Patient, on_delete=models.CASCADE)

    def __str__(self):
        return str(self.glucose) + " Категорія: " +
self.glucose_measurement_category

    class Meta:
        verbose_name = 'Замір глюкози'
        verbose_name_plural = 'Заміри глюкози'


class TypeOfActivity(models.Model):
    name = models.CharField(max_length=100, blank=False, null=False, unique=True)

    def __str__(self):
        return self.name

    class Meta:
        verbose_name = 'Тип активності'
        verbose_name_plural = 'Типи активності'


class PhysicalActivityMeasurement(models.Model):

    number_of_approaches = models.IntegerField(blank=True, null=False)
    type_of_activity = models.ForeignKey(TypeOfActivity, on_delete=models.PROTECT)
    date_of_measurement = models.DateField(default=timezone.now)
    time_of_activity = models.TimeField(default=timezone.now)
    commentary = models.TextField(max_length=1000, blank=True, null=True)
    patient_id = models.ForeignKey(Patient, on_delete=models.CASCADE)

    def __str__(self):
        return f'{self.patient_id} виконав {self.number_of_approaches} підходів у -
{self.type_of_activity.name}'
```

```
    class Meta:
        verbose_name = 'Замір фізичної активності'
        verbose_name_plural = 'Заміри фізичної активності'


class FoodItem(models.Model):

    name = models.CharField(max_length=200, blank=False, null=False, db_index=True)
    proteins = models.DecimalField(max_digits=6, decimal_places=2)
    fats = models.DecimalField(max_digits=6, decimal_places=2)
    carbohydrates = models.DecimalField(max_digits=6, decimal_places=2)

    def __str__(self):
        return f"Їжа: {self.name}, білки - {self.proteins}, жири - {self.fats},
вуглеводи - {self.carbohydrates}"

    class Meta:
        verbose_name = 'Порція їжі'
        verbose_name_plural = 'Порції їжі'


class FoodMeasurement(models.Model):

    CATEGORY = (
        ('Сніданок', 'Сніданок'),
        ('Перекус', 'Перекус'),
        ('Обід', 'Обід'),
        ('Другий перекус', 'Другий перекус'),
        ('Вечеря', 'Вечеря'),
    )

    category = models.CharField(choices=CATEGORY, blank=False, null=False,
default='Сніданок')
    insuline_dose_before = models.DecimalField(max_digits=5, decimal_places=2,
blank=False, null=False)
    insuline_dose_after = models.DecimalField(max_digits=5, decimal_places=2,
blank=True, null=True)
    date_of_measurement = models.DateField(default=timezone.now)
    time_of_eating = models.TimeField(default=timezone.now)
    bread_unit = models.IntegerField(blank=True, null=True)
    food_items = models.ManyToManyField(FoodItem)
    patient_id = models.ForeignKey(Patient, on_delete=models.CASCADE)

    def calculate_bread_unit(self):
        """
        Calculate bread unit (10-12g carbohydrate) for calculate insuline dose
        """
        if self.food_items:
            total_carbohydrates = 0
            for item in self.food_items.all():
```

```
                total_carbohydrates += item.carbohydrates
            bread_unit = total_carbohydrates/12
            return bread_unit
        else:
            return 0

    def calculate_dose(self):
        """
        Approximate calculation of insulin dose based on bread units
        """
        if self.bread_unit:
            return self.bread_unit


    def save(self, *args, **kwargs):
        """
        Overriding save method for init bread_unit after save measurement
        """
        self.insuline_dose_after = self.calculate_dose()
        super().save(*args, **kwargs)


    def __str__(self):
        return f'''
            Дата прийому їжі - {self.date_of_measurement}, час прийому -
{self.time_of_eating} |
            Кількість хлібних одиниць - {self.bread_unit}, приблизна доля інсуліну
- {self.insuline_dose_after}
        '''

    class Meta:
        verbose_name = 'Замір їжі'
        verbose_name_plural = 'Заміри їжі'


class InsulineDoseMeasurement(models.Model):

    CATEGORY = (
        ('Натщесердце', 'Натщесердце'),
        ('До сніданку', 'До сніданку'),
        ('Після сніданку', 'Після сніданку'),
        ('Перед перекусом', 'Перед перекусом'),
        ('Після перекусу', 'Після перекусу'),
        ('До обіду', 'До обіду'),
        ('Після обіду', 'Після обіду'),
        ('До вечері', 'До вечері'),
        ('Після вечері', 'Після вечері'),
        ('Перед сном', 'Перед сном'),
        ('Інше', 'Інше'),
    )
```

```
    category = models.CharField(choices=CATEGORY, blank=False, null=False)
    insuline_dose = models.DecimalField(max_digits=5, decimal_places=2,
blank=False, null=False)
    date_of_measurement = models.DateField(default=timezone.now)
    time = models.TimeField(default=timezone.now)
    patient_id = models.ForeignKey(Patient, on_delete=models.CASCADE)

    def __str__(self):
        return  f'Категорія: {self.category}, станом на {self.time} вкололи
{self.insuline_dose} ОД інсуліну'

    class Meta:
        verbose_name = 'Замір інсуліну'
        verbose_name_plural = 'Заміри інсуліну'


class AnalysType(models.Model):

    CATEGORY = (
        ('Загальний аналіз крові', 'Загальний аналіз крові'),
        ('Загальний аналіз сечі', 'Загальний аналіз сечі'),
        ('Аналіз крові на глюкозу', 'Аналіз крові на глюкозу'),
        ('Аналіз крові на глікований гемоглобин', 'Аналіз крові на глікований
гемоглобин'),
        ('Аналіз С-пептид', 'Аналіз С-пептид'),
        ('Комплексний аналіз "Ліпідограма" ', 'Комплексний аналіз "Ліпідограма"'),
        ('Інше', 'Інше'),
    )
    name = models.CharField(choices=CATEGORY, blank=False, null=False,
default='Інше')


class Analysis(models.Model):
    analysis = models.FileField(upload_to='patients-media/analysis')
    patient_id = models.ForeignKey(Patient, on_delete=models.CASCADE)
    date_of_measurement = models.DateField(default=timezone.now)
    analysis_type = models.ForeignKey(AnalysType, on_delete=models.PROTECT)

    def get_file_url(self):
        return self.analysis.url if self.analysis else None

    class Meta:
        verbose_name = 'Аналіз'
        verbose_name_plural = 'Аналізи'
```

forms.py

```
from django import forms
```

```
from django.utils.translation import gettext_lazy as _
from .models import GlucoseMeasurement, PhysicalActivityMeasurement,
TypeOfActivity, FoodMeasurement, FoodItem, InsulineDoseMeasurement
from .models import Analysis, AnalysType

class GlucoseMeasurementForm(forms.ModelForm):
    class Meta:
        model = GlucoseMeasurement
        fields = "__all__"
        exclude = ["patient_id",]
        labels = {
            'glucose': _('Кількість глюкози в крові (ммоль/л)'),
            'glucose_measurement_category': _('Категорія'),
            'date_of_measurement': _('Дата заміру'),
            'time_of_measurement': _('Час заміру'),
        }

    def __init__(self, *args, **kwargs):
        super(GlucoseMeasurementForm, self).__init__(*args, **kwargs)
        for field_name, field in self.fields.items():
            field.widget.attrs.update({'class': 'form-control'})
            if field_name == 'date_of_measurement':
                field.widget.input_type = 'date'


class PhysicalActivityForm(forms.ModelForm):

    type_of_activity = forms.ModelChoiceField(
        queryset=TypeOfActivity.objects.all(),
        label='Тип активності',
        required=True,
    )

    class Meta:
        model = PhysicalActivityMeasurement
        fields = "__all__"
        exclude = ["patient_id",]
        labels = {
            'number_of_approaches': _('Кількість підходів'),
            'commentary': _('Короткий коментар (що конкретно робили)'),
            'date_of_measurement': _('Дата заміру'),
            'time_of_activity': _('Час заміру'),
        }

    def __init__(self, *args, **kwargs):
        super(PhysicalActivityForm, self).__init__(*args, **kwargs)
        for field_name, field in self.fields.items():
            field.widget.attrs.update({'class': 'form-control'})
            if field_name == 'date_of_measurement':
                field.widget.input_type = 'date'
```

```
class FoodItemForm(forms.ModelForm):
    class Meta:
        model = FoodItem
        fields = "__all__"
        labels = {
            'name': _('Назва страви'),
            'proteins': _('Кількість білків'),
            'fats': _('Кількість жирів'),
            'carbohydrates': _('Кількість вуглеводів'),
        }


    def __init__(self, *args, **kwargs):
        super(FoodItemForm, self).__init__(*args, **kwargs)
        for field_name, field in self.fields.items():
            field.widget.attrs.update({'class': 'form-control'})


class FoodMeasurementForm(forms.ModelForm):
    class Meta:
        model = FoodMeasurement
        fields = "__all__"
        exclude = ["patient_id","food_items","bread_unit", "insuline_dose_after"]
        labels = {
            'insuline_dose_before': _('Доза інсуліну до їжі'),
            'date_of_measurement': _('Дата прийому їжі'),
            'time_of_activity': _('Час прийому їжі'),
        }

    def __init__(self, *args, **kwargs):
        super(FoodMeasurementForm, self).__init__(*args, **kwargs)
        for field_name, field in self.fields.items():
            field.widget.attrs.update({'class': 'form-control'})
            if field_name == 'date_of_measurement':
                field.widget.input_type = 'date'


class InsulineDoseForm(forms.ModelForm):
    class Meta:
        model = InsulineDoseMeasurement
        fields = "__all__"
        exclude = ["patient_id",]
        labels = {
            'category': _('Категорія'),
            'insuline_dose': _('Доза інсуліну (ОД)'),
            'date_of_measurement': _('Дата заміру'),
            'time': _('Час заміру'),
        }

    def __init__(self, *args, **kwargs):
        super(InsulineDoseForm, self).__init__(*args, **kwargs)
```

```
        for field_name, field in self.fields.items():
            field.widget.attrs.update({'class': 'form-control'})
            if field_name == 'date_of_measurement':
                field.widget.input_type = 'date'


class AnalysisTypeForm(forms.ModelForm):
    class Meta:
        model = AnalysType
        fields = ["name"]
        labels = {
            'name': _('Назва аналізу'),
        }

    def __init__(self, *args, **kwargs):
        super(AnalysisTypeForm, self).__init__(*args, **kwargs)
        for field_name, field in self.fields.items():
            field.widget.attrs.update({'class': 'form-control'})

class AnalysisForm(forms.ModelForm):
    class Meta:
        model = Analysis
        fields = ["analysis"]
```

views.py

```
from django.core.paginator import Paginator
from django.forms import modelformset_factory
from django.http import HttpResponseForbidden, FileResponse
from django.shortcuts import render, get_object_or_404, redirect
from django.views import generic, View
from django.urls import reverse, reverse_lazy
from login.models import Patient
from profiles.mixins import UserRoleMixin
from .models import GlucoseMeasurement, PhysicalActivityMeasurement,
FoodMeasurement, FoodItem, InsulineDoseMeasurement, Analysis, AnalysType
from .forms import GlucoseMeasurementForm, PhysicalActivityForm,
FoodMeasurementForm, FoodItemForm, InsulineDoseForm, AnalysisForm, AnalysisTypeForm
from .utils.utils import group_by_date


class GlucoseMeasurmentView(View, UserRoleMixin):
    model = GlucoseMeasurement
    template_name = 'patient_card/glucose.html'
    paginate_by = 24

    def get_context_data(self, patient_id, **kwargs):

        is_patient, is_doctor = self.get_user_roles(self.request)
```

```
        if is_doctor:
            glucose_measurements =
GlucoseMeasurement.objects.filter(patient_id=patient_id).order_by('-
date_of_measurement')
        elif self.request.user.id != patient_id:
            raise PermissionError('У вас немає доступу до цього ресурсу')
        else:
            glucose_measurements =
GlucoseMeasurement.objects.filter(patient_id=self.request.user.id).order_by('-
date_of_measurement')

        grouped_dates, dates_list = group_by_date(glucose_measurements)
        paginator = Paginator(dates_list, self.paginate_by)
        page_number = self.request.GET.get('page')
        page_obj = paginator.get_page(page_number)
        patient = get_object_or_404(Patient, id=patient_id)

        context = {
            'page_obj': page_obj,
            'grouped_dates': grouped_dates,
            'glucose_form': GlucoseMeasurementForm(),
            'is_doctor': is_doctor,
            'is_patient': is_patient,
            'patient_id': patient_id,
            'is_oninsuline': patient.is_oninsuline,
        }
        return context


    def get(self, request, patient_id, *args, **kwargs):
        """
        Retrieve measurments for current patient grouped by date and paginate by 4
card on page
        """
        try:
            context = self.get_context_data(patient_id)
        except PermissionError as e:
            return HttpResponseForbidden('У вас немає доступу до цього ресурсу')
        return render(request, self.template_name, context)


    def post(self, request, patient_id, *args, **kwargs):
        form = GlucoseMeasurementForm(request.POST)
        if form.is_valid():
            patient = get_object_or_404(Patient, id=request.user.id)
            glucose_measurement = form.save(commit=False)
            glucose_measurement.patient_id = patient
            glucose_measurement.save()
            redirect_url = reverse_lazy('patient_card', kwargs={'patient_id':
patient_id})
```

```
            return redirect(redirect_url)
        else:
            context = self.get_context_data(patient_id)
            return render(request, self.template_name, context)


class PhysicalMeasurementView(View, UserRoleMixin):
    model = PhysicalActivityMeasurement
    template_name = 'patient_card/physical.html'
    paginate_by = 24

    def get_context_data(self, patient_id, **kwargs):
        is_patient, is_doctor = self.get_user_roles(self.request)
        if (is_doctor):
            fit_measurements =
PhysicalActivityMeasurement.objects.filter(patient_id=patient_id).order_by('-
date_of_measurement')
        elif (self.request.user.id != patient_id):
            return HttpResponseForbidden('У вас немає доступу до цього ресурсу')
        else:
            fit_measurements =
PhysicalActivityMeasurement.objects.filter(patient_id=self.request.user.id).order_b
y('-date_of_measurement')


        grouped_dates, dates_list = group_by_date(fit_measurements)
        paginator = Paginator(dates_list, self.paginate_by)
        page_number = self.request.GET.get('page')
        page_obj = paginator.get_page(page_number)
        patient = get_object_or_404(Patient, id=patient_id)

        context = {
            'page_obj': page_obj,
            'grouped_dates': grouped_dates,
            'activity_form': PhysicalActivityForm(),
            'is_doctor': is_doctor,
            'is_patient': is_patient,
            'patient_id': patient_id,
            'is_oninsuline': patient.is_oninsuline,
        }
        return context


    def get(self, request, patient_id, *args, **kwargs):
        """
        Retrieve measurments for current patient grouped by date and paginate by 4
card on page
        """
        context = self.get_context_data(patient_id)
        return render(request, self.template_name, context)
```

```
    def post(self, request, patient_id, *args, **kwargs):
        form = PhysicalActivityForm(request.POST)
        if form.is_valid():
            patient = get_object_or_404(Patient, id=request.user.id)
            physical_activity = form.save(commit=False)
            physical_activity.patient_id = patient
            physical_activity.save()

            redirect_url = reverse_lazy('patient_physics', kwargs={'patient_id':
patient_id})
            return redirect(redirect_url)
        else:
            context = self.get_context_data(patient_id)
            return render(request, self.template_name, context)


class FoodMeasurementView(View, UserRoleMixin):
    model = FoodMeasurement
    template_name = 'patient_card/food.html'
    initial_food_item_count = 1
    paginate_by = 24

    def get_context_data(self, patient_id, **kwargs):
        is_patient, is_doctor = self.get_user_roles(self.request)

        if is_doctor:
            food_measurements =
FoodMeasurement.objects.filter(patient_id=patient_id).order_by('-
date_of_measurement')
        elif self.request.user.id != patient_id:
            return HttpResponseForbidden('У вас нет доступа к этому ресурсу')
        else:
            food_measurements =
FoodMeasurement.objects.filter(patient_id=self.request.user.id).order_by('-
date_of_measurement')

        grouped_dates, dates_list = group_by_date(food_measurements)
        paginator = Paginator(dates_list, self.paginate_by)
        page_number = self.request.GET.get('page')
        page_obj = paginator.get_page(page_number)
        patient = get_object_or_404(Patient, id=patient_id)

        context = {
            'page_obj': page_obj,
            'grouped_dates': grouped_dates,
            'food_measurement': FoodMeasurementForm(),
            'food_item_forms': [FoodItemForm(prefix=f'food_item_{i}') for i in
range(self.initial_food_item_count)],
            'is_doctor': is_doctor,
```

```
            'is_patient': is_patient,
            'patient_id': patient_id,
            'is_oninsuline': patient.is_oninsuline,
        }
        return context

    def get(self, request, patient_id, *args, **kwargs):
        context = self.get_context_data(patient_id)
        return render(request, self.template_name, context)

    def post(self, request, patient_id, *args, **kwargs):
        measurement_form = FoodMeasurementForm(request.POST)

        if measurement_form.is_valid():
            patient = get_object_or_404(Patient, id=request.user.id)
            measurement = measurement_form.save(commit=False)
            measurement.patient_id = patient
            measurement.save()

            prefixes = set(key.split('-')[0] for key in request.POST if
key.startswith('food_item_'))
            for prefix in prefixes:
                form = FoodItemForm(request.POST, prefix=prefix)
                if form.is_valid():
                    food_item = FoodItem(
                        name=form.cleaned_data['name'],
                        proteins=form.cleaned_data['proteins'],
                        fats=form.cleaned_data['fats'],
                        carbohydrates=form.cleaned_data['carbohydrates']
                    )
                    food_item.save()
                    measurement.food_items.add(food_item)

            measurement.bread_unit = measurement.calculate_bread_unit()
            measurement.save()
            redirect_url = reverse_lazy('patient_food', kwargs={'patient_id':
patient_id})
            return redirect(redirect_url)
        else:
            context = self.get_context_data(patient_id)
            return render(request, self.template_name, context)


class InsulineDoseView(View, UserRoleMixin):
    model = InsulineDoseMeasurement
    template_name = 'patient_card/insuline.html'
    paginate_by = 24

    def get_context_data(self, patient_id, **kwargs):
        is_patient, is_doctor = self.get_user_roles(self.request)
```

```
        if (is_doctor):
            insuline_measurements =
InsulineDoseMeasurement.objects.filter(patient_id=patient_id).order_by('-
date_of_measurement')
        elif (self.request.user.id != patient_id):
            return HttpResponseForbidden('У вас немає доступу до цього ресурсу')
        else:
            insuline_measurements =
InsulineDoseMeasurement.objects.filter(patient_id=self.request.user.id).order_by('-
date_of_measurement')


        grouped_dates, dates_list = group_by_date(insuline_measurements)
        paginator = Paginator(dates_list, self.paginate_by)
        page_number = self.request.GET.get('page')
        page_obj = paginator.get_page(page_number)
        patient = get_object_or_404(Patient, id=patient_id)

        context = {
            'page_obj': page_obj,
            'grouped_dates': grouped_dates,
            'insuline_form': InsulineDoseForm(),
            'is_doctor': is_doctor,
            'is_patient': is_patient,
            'patient_id': patient_id,
            'is_oninsuline': patient.is_oninsuline,
        }
        return context


    def get(self, request, patient_id, *args, **kwargs):
        """
        Retrieve measurments for current patient grouped by date and paginate by 4
card on page
        """
        context = self.get_context_data(patient_id)
        return render(request, self.template_name, context)


    def post(self, request, patient_id, *args, **kwargs):
        form = InsulineDoseForm(request.POST)
        if form.is_valid():
            patient = get_object_or_404(Patient, id=request.user.id)
            insuline = form.save(commit=False)
            insuline.patient_id = patient
            insuline.save()

            redirect_url = reverse_lazy('patient_insuline', kwargs={'patient_id':
patient_id})
            return redirect(redirect_url)
        else:
```

```
            context = self.get_context_data(patient_id)
            return render(request, self.template_name, context)


class AnalysisLoadView(View, UserRoleMixin):
    model = Analysis
    template_name = 'patient_card/analysis.html'
    paginate_by = 8

    def get_context_data(self, patient_id, **kwargs):

        is_patient, is_doctor = self.get_user_roles(self.request)

        if is_doctor:
            analysis = Analysis.objects.filter(patient_id=patient_id).order_by('-
date_of_measurement')
        elif self.request.user.id != patient_id:
            raise PermissionError('У вас немає доступу до цього ресурсу')
        else:
            analysis =
Analysis.objects.filter(patient_id=self.request.user.id).order_by('-
date_of_measurement')

        paginator = Paginator(analysis, self.paginate_by)
        page_number = self.request.GET.get('page')
        page_obj = paginator.get_page(page_number)

        context = {
            'page_obj': page_obj,
            'type_form': AnalysisTypeForm(),
            'analysis_form': AnalysisForm(),
            'is_doctor': is_doctor,
            'is_patient': is_patient,
            'patient_id': patient_id,
        }
        return context


    def get(self, request, patient_id, *args, **kwargs):
        """
        Retrieve measurments for current patient grouped by date and paginate by 4
card on page
        """
        try:
            context = self.get_context_data(patient_id)
        except PermissionError as e:
            return HttpResponseForbidden('У вас немає доступу до цього ресурсу')
        return render(request, self.template_name, context)


    def post(self, request, patient_id, *args, **kwargs):
```

```
        analysis_form = AnalysisForm(request.POST, request.FILES)
        analysis_type_form = AnalysisTypeForm(request.POST)
        if analysis_form.is_valid() and analysis_type_form.is_valid():
            patient = get_object_or_404(Patient, id=request.user.id)
            analysis = analysis_form.save(commit=False)

            _type = analysis_type_form.cleaned_data['name']
            if (AnalysType.objects.filter(name = _type).exists()):
                analysis_type = get_object_or_404(AnalysType, name=_type)
            else:
                analysis_type = analysis_type_form.save()

            analysis.analysis_type = analysis_type
            analysis.patient_id = patient
            analysis.save()
            redirect_url = reverse_lazy('patient_analysis', kwargs={'patient_id':
patient_id})
            return redirect(redirect_url)
        else:
            context = self.get_context_data(patient_id)
            return render(request, self.template_name, context)

from django.conf import settings
import os

def download_pdf(request, patient_id):
    patient = get_object_or_404(Patient, id=patient_id)
    analysis = get_object_or_404(Analysis, patient_id=patient)
    file_url = analysis.get_file_url()
    file_path = os.path.join(settings.MEDIA_ROOT, file_url.lstrip('/'))
    print(file_url)
    response = FileResponse(open(file_path, 'rb'), content_type='application/pdf')
    response['Content-Disposition'] = f'attachment;
filename="{os.path.basename(file_path)}.pdf"'
    return response
```

admin.py

```
from django.contrib import admin
from .models import GlucoseMeasurement, TypeOfActivity,
PhysicalActivityMeasurement, FoodMeasurement, FoodItem, InsulineDoseMeasurement
from .models import Analysis, AnalysType

admin.site.register(GlucoseMeasurement)
admin.site.register(TypeOfActivity)
admin.site.register(PhysicalActivityMeasurement)
admin.site.register(FoodMeasurement)
admin.site.register(FoodItem)
```

```
admin.site.register(InsulineDoseMeasurement)
admin.site.register(AnalysType)
admin.site.register(Analysis)
```

glucose.html

```
{% extends 'base.html' %}
{% load static %}

{% block styles %}
<link rel="stylesheet" href="{% static 'css/card.css' %}">
{% endblock styles %}

{% block content %}
<section id="section"  class="my-3">
    <div class="container">
        {% if is_patient %}
        <div class="row my-3">
          <div class="col-12 text-end">
            <a href="#meas_creation" class="btn btn-primary" data-bs-toggle="modal"
data-bs-target="#meas_creation">Додати новий запис</a>
          </div>
          <div class="modal fade" id="meas_creation" tabindex="-1" aria-
labelledby="exampleModalLabel" aria-hidden="true">
            <div class="modal-dialog modal-dialog-centered modal-dialog-scrollable
modal-lg">
                <div class="modal-content">
                    <div class="modal-header">
                        <h5 class="modal-title"
id="article_creation_label">Додавання нового запису</h5>
                        <button type="button" class="btn-close" data-bs-
dismiss="modal" aria-label="Close"></button>
                    </div>
                    <div class="modal-body">
                        <form method="post" novalidate>
                            {% csrf_token %}
                            {{ glucose_form.as_p }}
                            <button type="submit" class="btn btn-
primary">Зберегти</button>
                        </form>
                    </div>
                </div>
            </div>
          </div>
        </div>
        {% endif %}
        <div class="row">
            <div class="col-md-2 align-self-start text-start">
                <ul class="nav flex-column">
                    <li class="nav-item card_nav__item">
```

```
                    <a class="nav-link card__link active" aria-current="page"
href="{% url 'patient_card' patient_id %}">Заміри глюкози</a>
                  </li>
                  {% if is_oninsuline %}
                  <li class="nav-item card_nav__item">
                    <a class="nav-link card__link" aria-current="page" href="{%
url 'patient_insuline' patient_id=patient_id %}">Записи інсуліну</a>
                  </li>
                  {% endif %}
                  <li class="nav-item card_nav__item">
                    <a class="nav-link card__link" href="{% url 'patient_food'
patient_id=patient_id %}">Записи по прийомам їжі та розрахунок ХО</a>
                  </li>
                  <li class="nav-item card_nav__item">
                    <a class="nav-link card__link" href="{% url 'patient_physics'
patient_id=patient_id %}">Фізична активність</a>
                  </li>
                  <li class="nav-item card_nav__item">
                    <a class="nav-link card__link" href="{% url
'patient_analysis' patient_id=patient_id %}">Завантаження аналізів</a>
                  </li>
                </ul>
              </div>
              <div class="col-md-10 border meas__section">
                <div class="row my-3">
                  {% for date, measurements in page_obj %}
                  <div class="col-md-2">
                    <div class="dropdown">
                      <a class="btn btn-primary dropdown_btn" role="button" data-bs-
toggle="dropdown" aria-expanded="false">
                        {{ date }}
                      </a>
                      <ul class="dropdown-menu meas_glucose__list dropdown-menu-end">
                        <table class="table text-center">
                          <thead>
                            <tr>
                              <th scope="col">Категорія</th>
                              <th scope="col">Час заміру</th>
                              <th scope="col">Глюкоза (ммоль/л)</th>
                            </tr>
                          </thead>
                          <tbody>
                            {% for meas in measurements %}
                            <tr>
                              <td>
                                <div class="category-block">
                                  <span>{{ meas.glucose_measurement_category
}}</span>
                                </div>
                              </td>
```

```
                                <td>
                                  <div class="time-block">
                                    <span>{{ meas.time_of_measurement }}</span>
                                  </div>
                                </td>
                                <td>
                                  <div class="glucose-block">
                                    <span class="glucose_value">{{ meas.glucose
}}</span>
                                  </div>
                                </td>
                              </tr>
                              {% endfor %}
                            </tbody>
                          </table>
                        </ul>
                      </div>
                    </div>
                    {% if forloop.counter|divisibleby:6 and not forloop.last %}
                    </div><div class="row my-3">
                    {% endif %}
                    {% endfor %}
                  </div>
                  <div class="row align-self-end">
                      <div class="col-md-12">
                          {% if page_obj.has_other_pages %}
                              <nav aria-label="Page navigation">
                                  <ul class="pagination justify-content-center">
                                      {% if page_obj.has_previous %}
                                          <li class="page-item">
                                              <a class="page-link"
href="?page=1">&laquo;</a>
                                          </li>
                                          <li class="page-item">
                                              <a class="page-link" href="?page={{
page_obj.previous_page_number }}">{{ page_obj.previous_page_number }}</a>
                                          </li>
                                      {% endif %}
                                      <li class="page-item active" aria-current="page">
                                          <span class="page-link">{{ page_obj.number
}}</span>
                                      </li>
                                      {% if page_obj.has_next %}
                                          <li class="page-item">
                                              <a class="page-link" href="?page={{
page_obj.next_page_number }}">{{ page_obj.next_page_number }}</a>
                                          </li>
                                          <li class="page-item">
                                              <a class="page-link" href="?page={{
page_obj.paginator.num_pages }}">&raquo;</a>
```

```
                                    </li>
                                {% endif %}
                            </ul>
                        </nav>
                    {% endif %}
                </div>
            </div>
        </div>
    </div>
</section>
{% endblock content %}

{% block scripts %}
<script src="{% static 'js/dropdown_hover.js' %}"></script>
<script src="{% static 'js/glucose_card.js' %}"></script>
{% endblock scripts %}
```

Physic.html

```
{% extends 'base.html' %}
{% load static %}

{% block styles %}
<link rel="stylesheet" href="{% static 'css/card.css' %}">
{% endblock styles %}

{% block content %}
<section id="section"  class="my-3">
    <div class="container">
        {% if is_patient %}
        <div class="row my-3">
          <div class="col-12 text-end">
            <a href="#fit_creation" class="btn btn-primary" data-bs-toggle="modal"
data-bs-target="#fit_creation">Додати новий запис</a>
          </div>
          <div class="modal fade" id="fit_creation" tabindex="-1" aria-
labelledby="exampleModalLabel" aria-hidden="true">
            <div class="modal-dialog modal-dialog-centered modal-dialog-scrollable
modal-lg">
                <div class="modal-content">
                    <div class="modal-header">
                        <h5 class="modal-title"
id="article_creation_label">Додавання нового запису</h5>
                        <button type="button" class="btn-close" data-bs-
dismiss="modal" aria-label="Close"></button>
                    </div>
                    <div class="modal-body">
```

```
                        <form method="post" novalidate>
                            {% csrf_token %}
                            {{ activity_form.as_p }}
                            <button type="submit" class="btn btn-
primary">Зберегти</button>
                        </form>
                    </div>
                </div>
            </div>
          </div>
        </div>
        {% endif %}
        <div class="row">
            <div class="col-md-2 align-self-start text-start">
                <ul class="nav flex-column">
                    <li class="nav-item card_nav__item">
                      <a class="nav-link card__link" aria-current="page" href="{%
url 'patient_card' patient_id=patient_id %}">Заміри глюкози</a>
                    </li>
                    {% if is_oninsuline %}
                    <li class="nav-item card_nav__item">
                      <a class="nav-link card__link" aria-current="page" href="{%
url 'patient_insuline' patient_id=patient_id %}">Записи інсуліну</a>
                    </li>
                    {% endif %}
                    <li class="nav-item card_nav__item">
                      <a class="nav-link card__link" href="{% url 'patient_food'
patient_id=patient_id %}">Записи по прийомам їжі та розрахунок ХО</a>
                    </li>
                    <li class="nav-item card_nav__item">
                      <a class="nav-link card__link active" href="{% url
'patient_physics' patient_id=patient_id %}">Фізична активність</a>
                    </li>
                    <li class="nav-item card_nav__item">
                      <a class="nav-link card__link" href="{% url
'patient_analysis' patient_id=patient_id %}">Завантаження аналізів</a>
                    </li>
                </ul>
            </div>
            <div class="col-md-10 border meas__section">
              <div class="row my-3">
                {% for date, measurements in page_obj %}
                <div class="col-md-2">
                  <div class="dropdown">
                    <a class="btn btn-primary dropdown_btn" role="button" data-bs-
toggle="dropdown" aria-expanded="false">
                      {{ date }}
                    </a>
                    <ul class="dropdown-menu meas_fit__list dropdown-menu-end">
                        <table class="table text-center">
```

```
                    <thead>
                        <tr>
                          <th scope="col">Категорія</th>
                          <th scope="col">Час</th>
                          <th scope="col">Підходів</th>
                        </tr>
                    </thead>
                    <tbody>
                        {% for meas in measurements %}
                        <tr>
                          <td>
                            <div class="category-block">
                              <span>{{ meas.type_of_activity }}</span>
                            </div>
                          </td>
                          <td>
                            <div class="time-block">
                              <span>{{ meas.time_of_activity }}</span>
                            </div>
                          </td>
                          <td>
                            <div class="meas__block">
                              <span>{{ meas.number_of_approaches }}</span>
                            </div>
                          </td>
                        </tr>
                        <tr>
                          <td colspan="3">
                              <span>{{ meas.commentary }}</span>
                          </td>
                        </tr>
                        {% endfor %}
                    </tbody>
                </table>
            </ul>
          </div>
        </div>
        {% if forloop.counter|divisibleby:6 and not forloop.last %}
        </div><div class="row my-3">
        {% endif %}
        {% endfor %}
    </div>
    <div class="row align-self-end">
        <div class="col-md-12">
            {% if page_obj.has_other_pages %}
                <nav aria-label="Page navigation">
                    <ul class="pagination justify-content-center">
                        {% if page_obj.has_previous %}
                            <li class="page-item">
```

```
                                        <a class="page-link"
href="?page=1">&laquo;</a>
                                    </li>
                                    <li class="page-item">
                                        <a class="page-link" href="?page={{
page_obj.previous_page_number }}">{{ page_obj.previous_page_number }}</a>
                                    </li>
                                {% endif %}
                                <li class="page-item active" aria-current="page">
                                    <span class="page-link">{{ page_obj.number
}}</span>
                                </li>
                                {% if page_obj.has_next %}
                                    <li class="page-item">
                                        <a class="page-link" href="?page={{
page_obj.next_page_number }}">{{ page_obj.next_page_number }}</a>
                                    </li>
                                    <li class="page-item">
                                        <a class="page-link" href="?page={{
page_obj.paginator.num_pages }}">&raquo;</a>
                                    </li>
                                {% endif %}
                            </ul>
                        </nav>
                    {% endif %}
                </div>
            </div>
        </div>
    </div>
</section>
{% endblock content %}

{% block scripts %}
<script src="{% static 'js/dropdown_hover.js' %}"></script>
{% endblock scripts %}
```

Food.html

```
{% extends 'base.html' %}
{% load static %}

{% block styles %}
<link rel="stylesheet" href="{% static 'css/card.css' %}">
{% endblock styles %}

{% block content %}
<section id="section"  class="my-3">
```

```
    <div class="container">
        {% if is_patient %}
        <div class="row my-3">
          <div class="col-12 text-center">
              <h3>Примітка</h3>
              <p class="fs-4">
                На даній сторінці розраховується <strong
class="attention">ПРИБЛИЗНА</strong> доза інсуліну для компенсації вуглеводів
              </p>
              <p class="fs-4">
                Розрахунок йде за формулою 1 ХО (хлібна одиниця) - 12г вуглеводів.
1 ОД інсуліну = 1 ХО
              </p>
          </div>
        </div>
        <div class="row my-3">
          <div class="col-12 text-end">
            <a href="#meas_creation" class="btn btn-primary" data-bs-toggle="modal"
data-bs-target="#meas_creation">Додати новий запис</a>
          </div>
          <div class="modal fade" id="meas_creation" tabindex="-1" aria-
labelledby="exampleModalLabel" aria-hidden="true">
            <div class="modal-dialog modal-dialog-centered modal-dialog-scrollable
modal-lg">
                <div class="modal-content">
                    <div class="modal-header">
                        <h5 class="modal-title"
id="article_creation_label">Додавання нового запису</h5>
                        <button type="button" class="btn-close" data-bs-
dismiss="modal" aria-label="Close"></button>
                    </div>
                    <div class="modal-body">
                      <form method="post" id="food-measurement-form">
                        {% csrf_token %}
                        {{ food_measurement.as_p }}
                        <div id="food-item-forms">
                            {% for form in food_item_forms %}
                                <div class="food-item-form">
                                    {{ form.as_p }}
                                </div>
                            {% endfor %}
                        </div>
                        <button class="btn btn-primary" type="button" id="add-food-
item-btn">Додати страву</button>
                        <button class="btn btn-primary" type="submit">Зробити
запис</button>
                      </form>
                    </div>
                </div>
            </div>
```

```
          </div>
        </div>
        {% endif %}
        <div class="row">
            <div class="col-md-2 align-self-start text-start">
                <ul class="nav flex-column">
                    <li class="nav-item card_nav__item">
                      <a class="nav-link card__link active" aria-current="page"
href="{% url 'patient_card' patient_id=patient_id %}">Заміри глюкози</a>
                    </li>
                    {% if is_oninsuline %}
                    <li class="nav-item card_nav__item">
                      <a class="nav-link card__link" aria-current="page" href="{%
url 'patient_insuline' patient_id=patient_id %}">Записи інсуліну</a>
                    </li>
                    {% endif %}
                    <li class="nav-item card_nav__item">
                      <a class="nav-link card__link" href="{% url 'patient_food'
patient_id=patient_id %}">Записи по прийомам їжі та розрахунок ХО</a>
                    </li>
                    <li class="nav-item card_nav__item">
                      <a class="nav-link card__link" href="{% url 'patient_physics'
patient_id=patient_id %}">Фізична активність</a>
                    </li>
                    <li class="nav-item card_nav__item">
                      <a class="nav-link card__link" href="{% url
'patient_analysis' patient_id=patient_id %}">Завантаження аналізів</a>
                    </li>
                </ul>
            </div>
            <div class="col-md-10 border meas__section">
              <div class="row my-3">
                {% for date, measurements in page_obj %}
                <div class="col-md-2">
                  <div class="dropdown">
                    <a class="btn btn-primary dropdown_btn" role="button" data-bs-
toggle="dropdown" aria-expanded="false">
                      {{ date }}
                    </a>
                    <ul class="dropdown-menu meas_glucose__list dropdown-menu-end">
                        <table class="table text-center">
                          <thead>
                              <tr>
                                <th scope="col">Прийом їжі</th>
                                <th scope="col">Час прийому їжі</th>
                                <th scope="col">ХО</th>
                                <th scope="col">Доза інсуліну до їжі</th>
                                <th scope="col">Приблизна доза компенсації</th>
                              </tr>
                          </thead>
```

```
                    <tbody>
                        {% for meas in measurements %}
                        <tr>
                          <td>
                            <div class="category-block">
                              <span>{{ meas.category }}</span>
                            </div>
                          </td>
                          <td>
                            <div class="time-block">
                              <span>{{ meas.time_of_eating }}</span>
                            </div>
                          </td>
                          <td>
                            <div class="meas__block">
                              <span>{{ meas.bread_unit }}</span>
                            </div>
                          </td>
                          <td>
                            <div class="meas__block">
                              <span>{{ meas.insuline_dose_before }}</span>
                            </div>
                          </td>
                          <td>
                            <div class="meas__block">
                              <span>{{ meas.insuline_dose_after }}</span>
                            </div>
                          </td>
                        </tr>
                        <tr>
                          <td colspan="4">
                            {% for food in meas.food_items.all %}
                            <span>{{ food.name}},</span>
                            {% endfor %}
                          </td>
                        </tr>
                        {% endfor %}
                    </tbody>
                </table>
            </ul>
          </div>
        </div>
        {% if forloop.counter|divisibleby:6 and not forloop.last %}
        </div><div class="row my-3">
        {% endif %}
        {% endfor %}
    </div>
    <div class="row align-self-end">
        <div class="col-md-12">
            {% if page_obj.has_other_pages %}
```

```
                        <nav aria-label="Page navigation">
                            <ul class="pagination justify-content-center">
                                {% if page_obj.has_previous %}
                                    <li class="page-item">
                                        <a class="page-link"
href="?page=1">&laquo;</a>
                                    </li>
                                    <li class="page-item">
                                        <a class="page-link" href="?page={{
page_obj.previous_page_number }}">{{ page_obj.previous_page_number }}</a>
                                    </li>
                                {% endif %}
                                <li class="page-item active" aria-current="page">
                                    <span class="page-link">{{ page_obj.number
}}</span>
                                </li>
                                {% if page_obj.has_next %}
                                    <li class="page-item">
                                        <a class="page-link" href="?page={{
page_obj.next_page_number }}">{{ page_obj.next_page_number }}</a>
                                    </li>
                                    <li class="page-item">
                                        <a class="page-link" href="?page={{
page_obj.paginator.num_pages }}">&raquo;</a>
                                    </li>
                                {% endif %}
                            </ul>
                        </nav>
                    {% endif %}
                </div>
            </div>
        </div>
    </div>
</section>
{% endblock content %}

{% block scripts %}
<script src="{% static 'js/dropdown_hover.js' %}"></script>
<script src="{% static 'js/food.js' %}"></script>
{% endblock scripts %}
```

analysis.html

```
{% extends 'base.html' %}
{% load static %}
```

```
{% block styles %}
<link rel="stylesheet" href="{% static 'css/card.css' %}">
{% endblock styles %}

{% block content %}
<section id="section"  class="my-3">
    <div class="container">
        <div class="row">
            <div class="col-md-2 align-self-start text-start">
                <ul class="nav flex-column">
                    <li class="nav-item card_nav__item">
                      <a class="nav-link card__link active" aria-current="page"
href="{% url 'patient_card' patient_id %}">Заміри глюкози</a>
                    </li>
                    {% if is_oninsuline %}
                    <li class="nav-item card_nav__item">
                      <a class="nav-link card__link" aria-current="page" href="{%
url 'patient_insuline' patient_id=patient_id %}">Записи інсуліну</a>
                    </li>
                    {% endif %}
                    <li class="nav-item card_nav__item">
                      <a class="nav-link card__link" href="{% url 'patient_food'
patient_id=patient_id %}">Записи по прийомам їжі та розрахунок ХО</a>
                    </li>
                    <li class="nav-item card_nav__item">
                      <a class="nav-link card__link" href="{% url 'patient_physics'
patient_id=patient_id %}">Фізична активність</a>
                    </li>
                    <li class="nav-item card_nav__item">
                      <a class="nav-link card__link" href="{% url
'patient_analysis' patient_id=patient_id %}">Завантаження аналізів</a>
                    </li>
                </ul>
            </div>
            <div class="col-md-10 border meas__section">
            <div class="analyses_block">
            {% for analysis in page_obj %}
                <div class="row my-4">
                    <div class="col-md-12">
                        <div class="analysis_row">
                            <span>{{ analysis.date_of_measurement }}</span>
                            <span>{{ analysis.analysis_type.name }}</span>
                            <a href="{% url 'analysis_download'
patient_id=patient_id %}" download>
                                <button type="button">
                                    <img src="{% static 'img/svg/download.svg' %}"
alt="Кнопка завантаження аналізу">
                                </button>
                            </a>
                        </div>
```

```
                    </div>
                </div>
            {% endfor %}
        </div>
            <div class="row align-self-end">
                <div class="col-md-12">
                    {% if page_obj.has_other_pages %}
                        <nav aria-label="Page navigation">
                            <ul class="pagination justify-content-center">
                                {% if page_obj.has_previous %}
                                    <li class="page-item">
                                        <a class="page-link"
href="?page=1">&laquo;</a>
                                    </li>
                                    <li class="page-item">
                                        <a class="page-link" href="?page={{
page_obj.previous_page_number }}">{{ page_obj.previous_page_number }}</a>
                                    </li>
                                {% endif %}
                                <li class="page-item active" aria-current="page">
                                    <span class="page-link">{{ page_obj.number
}}</span>
                                </li>
                                {% if page_obj.has_next %}
                                    <li class="page-item">
                                        <a class="page-link" href="?page={{
page_obj.next_page_number }}">{{ page_obj.next_page_number }}</a>
                                    </li>
                                    <li class="page-item">
                                        <a class="page-link" href="?page={{
page_obj.paginator.num_pages }}">&raquo;</a>
                                    </li>
                                {% endif %}
                            </ul>
                        </nav>
                    {% endif %}
                </div>
            </div>
            {% if is_patient %}
            <div class="row">
                <div class="col-12">
                    <form id="analysis_drag" method="post" enctype="multipart/form-
data">
                        {% csrf_token %}
                        <div class="row align_items_center">
                            <div class="col-md-6">
                                {{ type_form.as_p }}
                            </div>
                            <div class="col-md-6">
```

```
                                <input id="fileInput" type="file" name="analysis"
style="display:none">
                                <div id="dropzone" class="dropzone">
                                    Перетягніть файл сюди або натисніть, щоб
вибрати файл
                                </div>
                                <div class="upload_error" id="upload_error">

                                </div>
                            </div>
                        </div>
                    </form>
                </div>
            </div>
            {% endif %}
        </div>
    </div>
</section>
{% endblock content %}

{% block scripts %}
<script src="{% static 'js/dropdown_hover.js' %}"></script>
<script src="{% static 'js/drag-n-drop-analysis.js' %}"></script>
{% endblock scripts %}
```

card.css

```
html, body {
    height: 100%;
    margin: 0;
    padding: 0;
}

body {
    display: flex;
    flex-direction: column;
}


.attention{
    color: red;
}

.meas__section {
    display: flex;
    flex-direction: column;
    justify-content: space-between;
```

```
    background-color: bisque;
    min-height: 600px;
}

.analyses_block{
    display: flex;
    flex-direction: column;
}

.card_nav__item {
    margin-bottom: 10px;
    text-align: center;
}

.card__link {
    display: block;
    padding: 10px;
    background-color: #4169E1;
    color: white;
    text-decoration: none;
}

.card__link:hover {
    background-color: rgb(68, 224, 6);
    color: black;
}

.form-control {
    margin-bottom: 15px;
}

.dropdown_btn{
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100px;
    width: 100%;
}

.meas_glucose__list{
    min-width: 350px;
    max-height: 300px;
    overflow: hidden;
    overflow-y: auto;
    flex-wrap: wrap;
}

.meas_fit__list{
    min-width: 600px;
    max-height: 300px;
```

```
    overflow: hidden;
    overflow-y: auto;
    flex-wrap: wrap;
}

.meas__block{
    background-color: #4169E1;
}

.meas__block, .category-block, .time-block, .glucose-block, .bad{
    display: flex;
    justify-content: center;
    align-items: center;
    height: 40px;
    width: 100%;
    border-radius: 5px;
    padding: 5px 10px;
    margin-right: 10px;
    color: #FFF;
}

.category-block {
    background-color: #4169E1;
    margin-left: 10px;
}

.glucose-block {
    background-color: #4169E1;
}

.time-block {
    background-color: #eeaf5d;
}

.bad{
    background-color: #ff6666;
}

.analysis_row{
    display: flex;
    justify-content: space-between;
    align-items: center;
    height: 50px;
    padding: 20px 20px;
    background-color: #4169E1;
    color: #FFF;
}


.upload_error{
    display: none;
```

```
    color: red;
    font-weight: 400;
    font-size: 24px;
}

.dropzone{
    padding: 20px 20px;
    border: 5px dashed black;
    border-radius: 5px;
    width: 100%;
    margin: 0;
}

@media (max-width: 768px){
    .dropdown_btn{
        margin-bottom: 10px;
        width: 100%;
    }
}
```

Dropdown_hover.js

```
document.addEventListener("DOMContentLoaded", function() {
    var dropdowns = document.querySelectorAll('.dropdown');

    dropdowns.forEach(function(dropdown) {

        dropdown.addEventListener('mouseenter', function() {
            this.querySelector('.dropdown-menu').classList.add('show');
        });
        dropdown.addEventListener('mouseleave', function() {
            this.querySelector('.dropdown-menu').classList.remove('show');
        });
    });


});
```

drag-n-drop_analysis.js

```
var dropzone = document.getElementById('dropzone');
var fileInput = document.getElementById('fileInput');
var error_block = document.getElementById('upload_error');

dropzone.onclick = function() {
    fileInput.click();
};
```

```
dropzone.ondragover = function(e) {
    e.preventDefault();
};

fileInput.onchange = function(e) {
    e.preventDefault();

    var file = fileInput.files[0];
    if (drop_validation(file)){
        return;
    }

    document.getElementById('analysis_drag').submit();
};

dropzone.ondrop = function(e) {
    e.preventDefault();
    fileInput.files = e.dataTransfer.files;

    var file = fileInput.files[0];
    if (drop_validation(file)){
        return;
    }

    document.getElementById('analysis_drag').submit();
};

function drop_validation(file){
    if (file.type != 'application/pdf'){
        error_block.style.display = 'block';
        error_block.innerText = 'Файл повинен бути в розширені .pdf';
        return;
    }
}
```

food.js

```
document.addEventListener('DOMContentLoaded', function() {
    const addButton = document.getElementById('add-food-item-btn');
    const formContainer = document.getElementById('food-item-forms');
    let formCount = 1;

    addButton.addEventListener('click', function() {
        const formPrefix = `food_item_${formCount}`;
        const formHtml = `
            <div class="food-item-form">
                <label for="${formPrefix}-name">Назва продукту:</label>
                <input type="text" name="${formPrefix}-name" id="${formPrefix}-
name" class="form-control">
                <label for="${formPrefix}-proteins">Білки:</label>
```

```
                <input type="text" name="${formPrefix}-proteins" id="${formPrefix}-
proteins" class="form-control">
                <label for="${formPrefix}-fats">Жири:</label>
                <input type="text" name="${formPrefix}-fats" id="${formPrefix}-
fats" class="form-control">
                <label for="${formPrefix}-carbohydrates">Вуглеводи:</label>
                <input type="text" name="${formPrefix}-carbohydrates"
id="${formPrefix}-carbohydrates" class="form-control">
            </div>
        `;
        formContainer.insertAdjacentHTML('beforeend', formHtml);
        formCount++;
    });

    const form = document.getElementById('food-measurement-form');
    form.addEventListener('submit', function(event) {

        const allForms = document.querySelectorAll('.food-item-form input');
        let isValid = true;
        allForms.forEach(input => {
            if (input.required && !input.value.trim()) {
                isValid = false;
            }
        });

        if (!isValid) {
            event.preventDefault();
            alert('Заполните все обязательные поля продуктов');
        }
    });
});
```

glucose_card.js

```
document.addEventListener("DOMContentLoaded", function() {
    var glucoseMeasElements = document.querySelectorAll('.glucose-block');

    glucoseMeasElements.forEach(function(glucoseMeas) {
        var glucoseValueSpan = glucoseMeas.querySelector('.glucose_value');

        if (glucoseValueSpan) {
            var glucoseValueText = glucoseValueSpan.innerText;
            var glucoseValue = parseFloat(glucoseValueText);
            console.log(glucoseValue);

            if (!isNaN(glucoseValue)) {
                if (glucoseValue <= 3 || glucoseValue > 6) {
                    glucoseMeas.classList.add('bad');
```

```
                }
            }
        }
    });
});
```

Додаток chat

urls.py

```
from django.urls import path
from .views import render_chat, send_message, load_messages

urlpatterns = [
    path('dialog/<int:sender_id>/<int:receiver_id>', render_chat, name='chat'),
    path('send-message/', send_message, name='send_message'),
    path('load-messages/<int:sender_id>/<int:receiver_id>/', load_messages,
name='load_message'),
]
```

Models.py

```
from django.db import models
from django.contrib.auth.models import User

class ChatMessage(models.Model):
    sender = models.ForeignKey(User, on_delete=models.CASCADE, blank=False,
null=False, related_name='sent_messages')
    receiver = models.ForeignKey(User, on_delete=models.CASCADE, blank=False,
null=False, related_name='receive_messages')
    timestamp = models.DateTimeField(auto_now_add=True)
    message_text = models.TextField(max_length=2000, blank=False, null=False)
```

views.py

```
from django.http import JsonResponse
from .models import ChatMessage
from django.views.decorators.http import require_POST
from django.contrib.auth.models import User
from django.shortcuts import render, get_object_or_404
import json
```

```
def render_chat(request, sender_id, receiver_id):
    sender = get_object_or_404(User, pk=sender_id)
    receiver = get_object_or_404(User, pk=receiver_id)
    context = {
        'sender': sender,
        'receiver': receiver,
        'sender_id': sender.id,
        'receiver_id': receiver.id,
    }
    return render(request, 'chat/chat.html', context)


@require_POST
def send_message(request):
    data = json.loads(request.body)
    sender_id = data.get('sender_id')
    receiver_id = data.get('receiver_id')
    message_text = data.get('message_text')

    if sender_id and receiver_id and message_text:
        sender = get_object_or_404(User, pk=sender_id)
        receiver = get_object_or_404(User, pk=receiver_id)

        ChatMessage.objects.create(
            sender=sender,
            receiver=receiver,
            message_text=message_text
        )

        return JsonResponse({'success': True})
    else:
        return JsonResponse({'success': False})


def load_messages(request, sender_id, receiver_id):
    sender_receiver_messages = ChatMessage.objects.filter(sender_id=sender_id,
receiver_id=receiver_id)
    receiver_sender_messages = ChatMessage.objects.filter(sender_id=receiver_id,
receiver_id=sender_id)

    all_messages = list(sender_receiver_messages) + list(receiver_sender_messages)

    sorted_messages = sorted(all_messages, key=lambda message: message.timestamp)
    message_list = [
        {
            'sender_id': message.sender_id,
            'sender_username': message.sender.username,
            'recipient_username': message.receiver.username,
            'timestamp': message.timestamp.strftime('%Y-%m-%d %H:%M:%S'),
            'content': message.message_text,
```

```
        }
        for message in sorted_messages
    ]
    return JsonResponse({'messages': message_list})
```

chat.html

```
{% extends 'base.html' %}
{% load static %}

{% block styles %}
<link rel="stylesheet" href="{% static 'css/chat.css' %}">
{% endblock styles %}

{% block content %}
<section id="section" class="my-3">
    <div class="container">
        <div class="row align-items-center">
            <div class="col-12">
                <h2 class="text-center">Чат з користувачем {{ receiver.username
}}</h2>
            </div>
            <div class="col-12 text-center">
                <div class="message_box" id="chat-container">

                </div>
                <form id="message-form">
                    {% csrf_token %}
                    <input type="hidden" id="sender-id" value="{{ sender_id }}">
                    <input type="hidden" id="receiver-id" value="{{ receiver_id
}}">
                    <input type="hidden" id="current-user-id" value="{{
request.user.id }}">
                    <textarea id="message-text" placeholder="Введіть
повідомлення"></textarea>
                    <button class="btn btn-primary text-end"
type="submit">Відправити повідомлення</button>
                </form>
            </div>
        </div>
    </div>
</section>
{% endblock content %}

{% block scripts %}
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
<script>
    const csrfToken = "{{ csrf_token }}";
    const senderId = "{{ sender_id }}";
```

```
    const receiverId = "{{ receiver_id }}"
</script>
<script src="{% static 'js/chat.js' %}"></script>
{% endblock scripts %}
```

chat.css

```
html, body {
    height: 100%;
    margin: 0;
    padding: 0;
}

body {
    display: flex;
    flex-direction: column;
}


#section {
    flex: 1;
}

footer{
    margin-top: auto;
}

#chat-container {
    border: 1px solid black;
    border-radius: 10px;
    min-height: 800px;
    overflow-y: auto;
    background-color: rgb(255, 255, 255);
}

.message {
    margin-top: 10px;
    border: 1px solid #3a47f7;
    border-radius: 8px;
    padding: 10px;
    margin-bottom: 10px;
    width: 30%;
}

.sent {
    background-color: #FFF;
    padding: 10px;
    border-radius: 5px;
```

```
    margin-left: auto;
    margin-right: 15px;
}

.received {
    background-color: #3b7bf3;
    padding: 10px;
    border-radius: 5px;
    margin-right: auto;
    margin-left: 15px;
}

.message strong {
    color: black;
}

.message-sender, .message-content{
    text-align: left;
    color: black;
}

.message-timestamp{
    color: black;
    text-align: right;
}

textarea{
    margin-top: 15px;
    border: 1px solid black;
    border-radius: 10px;
    width: 100%;
    height: 100px;
    padding: 20px 20px;
}
```

chat.js

```
$(document).ready(function() {
    loadMessages(senderId, receiverId);

    setInterval(function() {
        loadMessages(senderId, receiverId);
    }, 10000);

    $.ajaxSetup({
        beforeSend: function(xhr, settings) {
            if (!this.crossDomain) {
                xhr.setRequestHeader('X-CSRFToken', csrfToken);
```

```
            }
        }
    });


    function sendMessage(senderId, receiverId, messageText) {
        $.ajax({
            url: '/chat/send-message/',
            method: 'POST',
            contentType: 'application/json',
            data: JSON.stringify({
                'sender_id': senderId,
                'receiver_id': receiverId,
                'message_text': messageText,
            }),
            success: function(data) {
                if (data.success) {
                    $('#message-text').val('');
                } else {
                    alert('Ошибка при отправке сообщения.');
                }
            },
            error: function(xhr, status, error) {
                console.error('Ошибка при отправке сообщения:', error);
            }
        });
    }

    function loadMessages(senderId, receiverId) {
        $.ajax({
            url: `/chat/load-messages/${senderId}/${receiverId}/`,
            method: 'GET',
            success: function(data) {
                $('#chat-container').empty();
                var currentUserId = parseInt($('#current-user-id').val());
                data.messages.forEach(function(message) {
                    var isSentMessage = (message.sender_id == currentUserId);
                    console.log(isSentMessage);
                    var messageHtml = `
                        <div class="message ${isSentMessage ? 'sent' :
'received'}">
                            <p class="message-
sender"><strong>${message.sender_username}</strong></p><hr>
                            <p class="message-content">${message.content}</p>
                            <p class="message-timestamp">${message.timestamp}</p>
                        </div>
                    `;
                    $('#chat-container').append(messageHtml);
                });
            },
```

```
            error: function(xhr, status, error) {
                console.error('Ошибка при загрузке сообщений:', error);
            }
        });
    }

    $('#message-form').submit(function(e) {
        e.preventDefault();
        var senderId = $('#sender-id').val();
        var receiverId = $('#receiver-id').val();
        var messageText = $('#message-text').val();

        sendMessage(senderId, receiverId, messageText);
    });
});
```