МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«До захисту допущено»
В.о. завідувача кафедри
Ігор ШЕЛЕХОВ
(підпис)
червня 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавр

зі спеціальності 122 – Комп'ютерних наук,

освітньо-професійної програми «Інформатика»

на тему: «Медична інформаційна система стоматологічної клініки»

здобувача групи ІН-93 Прокопенка Владислава Олександровича

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

Владислав ПРОКОПЕНКО
(підпис)

Керівник,
асистентка кафедри комп'ютерних наук,
кандидат фізико-математичних наук Ольга ШУТИЛЄВА
(підпис)

Суми – 2023

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«Затверджую»
В.о. завідувача кафедри
_____ Ігор ШЕЛЕХОВ
(підпис)

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня бакалавра

зі спеціальності 122 – Комп'ютерних наук, освітньо-професійної програми «Інформатика»
здобувача групи ІН-93 Прокопенко Владислав Олександрович

1. Тема роботи: «Медична інформаційна система стоматологічної клініки.»
затверджую наказом по СумДУ від «01» червня 2023 р. № 0475-VI
2. Термін здачі здобувачем кваліфікаційної роботи *до 09 червня 2023 року*
3. Вхідні дані до кваліфікаційної роботи _____
4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)
1) Аналіз проблеми предметної області, постановка й формування завдань дослідження. 2) Огляд технологій, що використовуються для розробки. 3) Розробка Медичної інформаційної системи стоматологічної клініки. 4) Аналіз результатів.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)_____
6. Консультанти до проєкту (роботи), із значенням розділів проєкту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «____» _____ 2023 р.

Завдання прийняв до виконання _____ Керівник _____
(підпис) (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	Аналіз проблеми предметної області, постановка й формування завдань дослідження.		
2	Огляд технологій, що використовуються для розробки.		
3	Розробка Медичної інформаційної системи стоматологічної клініки.		
4	Аналіз результатів.		
5	Оформлення пояснювальної записки до кваліфікаційної роботи		

Здобувач вищої освіти _____ Керівник _____

(підпис) (підпис)

АНОТАЦІЯ

Записка: 59 стр., 18 рис., 1 додаток, 15 літературних джерел

Обґрунтування актуальності теми роботи – тема кваліфікаційної роботи є актуальною, оскільки присвячена розв'язанню практичної задачі розробки веб-ресурсу для стоматологічної клініки. Такий веб-ресурс дозволить автоматизувати багато рутинних і робочих процесів, таких як запис пацієнтів, забезпечення взаємодії між лікарями та адміністративним персоналом, тощо. Це дозволяє зменшити час на адміністративні завдання, і забезпечує більше часу для надання якісної стоматологічної допомоги.

Об'єкт дослідження – процес розробки веб-ресурсу для стоматологічної клініки "Dent".

Мета роботи – розробити і програмно реалізувати веб-ресурс для стоматологічної клініки "Dent", який дозволяє клієнтам здійснювати запис на прийоми та переглядати інформацію про послуги, а працівникам клініки – керувати записами і надавати необхідні послуги.

Методи досліджень – системно-інформаційний аналіз, інформаційне моделювання, комп'ютерний експеримент.

Результати – розроблений веб-ресурс за допомогою HTML/CSS, Python, фреймворку Django. Веб-ресурс реалізований як онлайн-платформа для стоматологічної клініки "Dent", де клієнти можуть здійснювати запис на прийоми, переглядати інформацію про, а працівники клініки можуть керувати записами і надавати необхідні послуги.

ВЕБ-РЕСУРС, PYTHON, DJANGO, СТОМАТОЛОГІЧНА КЛІНІКА, МЕДИЧНІ ПОСЛУГИ

ЗМІСТ

ВСТУП 5
1. ІНФОРМАЦІЙНИЙ ОГЛЯД 7
1.1 Аналіз предметної області 7
1.2 Аналіз ринку і конкурентів 8
1.3 Аналіз аналогів 10
1.4 Аналіз принципів побудови інтернет-сайту 13
1.5 Постановка задачі 16
2 ВИБІР МЕТОДІВ РІШЕННЯ ЗАДАЧІ 18
2.1 Огляд використаних інструментів 18
2.2 Моделювання бази даних 19
3 ПРОГРАМНА РЕАЛІЗАЦІЯ 23
ВИСНОВКИ 35
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 36
ДОДАТОК А 38

ВСТУП

Розробка медичної інформаційної системи (МІС) для стоматологічної клініки **є актуальною темою** з кількох причин. Розширення спектру соціальних груп, які користуються Інтернетом і шукають інформацію в мережі, створює необхідність у розвитку інформаційних систем для задоволення їхніх потреб. Ці групи не обов'язково є експертами в галузі інформаційних технологій, але вони використовують Інтернет для своєї професійної діяльності або особистих захоплень. Лікарі, будівельники, історики, юристи, спортсмени та багато інших професіоналів відчули необхідність у доступі до різних джерел інформації, які надає Інтернет. Застосування веб-технологій змінило парадигму роботи з інформацією і комп'ютерами.

У світі, де веб-технології поширені і загально доступні, комп'ютери стають не просто інструментами, а необхідними складовими частинами повсякденного життя. Вони змінюють спосіб спілкування, те як користувачі отримують інформацію, працюють та розважаються. Із кожним днем веб-технології стають все потужнішими, спрощуються їхній інтерфейси та підвищується доступність, що відкриває нові горизонти для нашого розвитку як суспільства.

У цьому контексті, розробка медичної інформаційної системи стоматологічної клініки стає важливим завданням. Інформація, доступна користувачам Інтернету, розташовується на комп'ютерах, які виконують функції веб-серверів та мають встановлене спеціальне програмне забезпечення. Велика частина цієї інформації організована у вигляді веб-сайтів, кожен з яких має власне ім'я або адресу в Інтернеті.

Метою роботи є створення медичної інформаційної системи для стоматологічної клініки, яка допоможе в просуванні їхніх послуг. **Гіпотеза роботи** полягає у поліпшенні взаємодії з пацієнтами, ефективному зберіганні

медичної інформації, плануванні роботи лікарів та забезпеченні зручного доступу до необхідної інформації. Використання Медичної інформаційної системи дозволить оптимізувати процеси обслуговування пацієнтів та покращити якість медичної допомоги, що в свою чергу сприятиме позитивному розвитку стоматологічної клініки.

Об’єкт дослідження для кваліфікаційної роботи – це процес просування послуг стоматологічної клініки через розробку інформаційної системи.

Предмет дослідження – це розробка медичної інформаційної системи для стоматологічної клініки з метою покращення взаємодії з пацієнтами та ефективного просування послуг.

Структура роботи. Дана робота складається зі вступу, аналітичного огляду, постановки задачі, вибору методу розв’язання поставленої задачі, опису програмного забезпечення інформаційної системи, висновків, списку використаних джерел та додатків.

1. ІНФОРМАЦІЙНИЙ ОГЛЯД

1.1 Аналіз предметної області

На сьогоднішній день розробка веб-сайтів стала надзвичайно актуальною темою. Багато підприємств, компаній, магазинів, закладів харчування, освітніх установ та інших суб'єктів господарювання мають власні веб-сайти, які використовуються для взаємодії з клієнтами. Володіння власним веб-сайтом в інтернеті є важливим і необхідним кроком, який дозволяє легко розширити бізнес і залучити нову аудиторію. Розробка веб-сайтів передбачає створення маркетингового інструменту, спрямованого на збільшення попиту на продукцію та послуги, або інформаційного ресурсу, який має на меті надання цільовій аудиторії необхідної інформації, або певного сервісу, який надає можливість користувачам отримувати певні послуги, які їх цікавлять. Варто зазначити, що процес створення функціональних веб-сайтів є складним і вимагає високого рівня професіоналізму. Однак також можна створювати прості динамічні або статичні сайти, які мають менше функціональних можливостей, але виконують свої завдання. Зазвичай для створення таких сайтів розробникам достатньо володіння мовою HTML (гіпертекстове розмітка), CSS (каскадні таблиці стилів) та розуміння систем управління контентом. Під час дослідження актуальності проблеми було проаналізовано велику кількість веб-сайтів закладів харчування. З урахуванням отриманої інформації можна зробити висновок, що завдяки своєму веб-сайту заклади харчування отримують багато замовлень та приваблюють нових клієнтів. Сайт сприяє популярності закладу, що збільшує прибуток і надає можливість розширення діяльності та надання більш широкого спектру послуг.

Напрямом діяльності досліджуваного об'єкту є надання своїх ресурсів клієнтам для задоволення їх потреб, а саме потреб у догляді за ротовою

порожниною. Відсутність сучасної стоматологічної клініки призвела до відкриття центру “tootheal”. Щоб підтримувати себе, своє здоров’я в хорошому стані люди відвідують спеціально обладнані, з відповідним персоналом центри. Це дозволяє людям слідкувати за станом свого здоров’я. Для того, щоб отримати інформацію, яка цікавить користувача, він повинен прийти до одного з центрів в приймальню адміністратора і дізнатися інформацію, яка його цікавить у адміністратора. Для автоматизації процесу необхідно розробити веб-систему. Розроблювана система покликана надати наступну інформацію клієнту про центр:

– його розташування;
– робочий графік;
– послуги, які він надає;
– можливість записатися на послугу;
– написати листа адміністратору, а адміністратору:
– додавати послуги;
– видаляти послуги, редагувати послуги.

Дана система дозволить користувачу отримати всю необхідну інформацію не виходячи з дому. Це дозволить суттєво зекономити час клієнта.

1.2 Аналіз ринку і конкурентів

На жаль, у сучасних реаліях просування стоматології означає зустріч з безліччю конкурентів. Великі міста надають можливість замовити послугу стоматолога практично на кожному кроці. Існує різноманітність варіантів, які можуть бути дешевими, дорогими, елітними, якісними або поганими, і таке розмаїття можливостей може викликати тривогу. Однак, правильний підхід лише зміцнить просування сайту стоматологічної компанії.

Найбільш поширеним способом, яким пацієнти знаходять собі лікаря, є через рекомендації знайомих або за допомогою веб-сайтів та соціальних

мереж. У сучасному світі сарафанне радіо вже не є достатньо надійним джерелом інформації для клієнтів. Однак належно оформлений і доглянутий сайт створить приємне перше враження про роботу стоматолога і може стати важливим чинником у виборі лікаря пацієнтом. Завдяки сайту, пацієнти можуть ознайомитися з послугами, робочим графіком, кваліфікацією та досвідом лікаря, а також переглянути відгуки і рекомендації інших клієнтів. Такий професійно розроблений веб-сайт сприяє побудові довіри та надає гарантію якості роботи стоматолога, що дозволяє пацієнтам прийняти відповідне рішення щодо вибору лікаря..

Конкурентів багато, однак, більшість з них не надають ринку будь-яких унікальних способів залучення клієнтів стоматологічних послуг. Отже, навіть звичайний сайт стоматології забезпечить вам надійний прихід пацієнтів.

Варто відзначити, що сайт не тільки допомагає залучити більше клієнтів, а й прибирає безліч побоювань, пов’язаних з походом до стоматолога завдяки наступним поліпшенням:

1. Забезпечення можливості перегляду відгуків. Відгуки клієнтів є найкращим способом оцінити роботу лікаря, вони надають цінну інформацію про якість надання послуг стоматологічної клініки. Це дозволяє зняти вагу невизначеності і допомагає прийняти рішення щодо вибору клініки.

2. Швидка відповідь та можливість обрати зручний час візиту. Клієнтам більше не потрібно дзвонити, щоб записатися на прийом. Вони можуть швидко та зручно обрати бажаний час через спеціальну форму на веб-сайті клініки.

3. Надання орієнтовної вартості обраної стоматологічної послуги. Клієнти цінують можливість отримати приблизну оцінку вартості необхідного лікування або процедури безпосередньо на сайті клініки.

4. Чітка комунікація з клінікою на всіх етапах спілкування. Веб-сайт забезпечує зручний канал зв'язку між клієнтами та клінікою, де можна

отримати відповіді на запитання, домовитися про прийом або отримати додаткову інформацію.

5. Збільшення середнього чека. Інформаційна система на веб-сайті може пропонувати додаткові послуги або рекомендації, що сприяють збільшенню середнього чека під час відвідування клініки.

6. Можливість оплати як готівкою, так і кредитними картками. Веб-сайт надає зручні способи оплати, що задовольняють різні потреби клієнтів і сприяють зручності процесу оплати за надані послуги.

1.3 Аналіз аналогів

Завданням кваліфікаційної роботи є створення медичної інформаційної системи стоматологічної клініки. Беручи це до уваги, було проведено моніторинг різних сайтів аналогічного типу.

Наприклад, сайт cherevkodentalclinic.com (рис.1.1), сайт ukrstom-centr.com.ua (рис.1.2), сайт www.stomatclinica.com.ua (рис. 1.3).

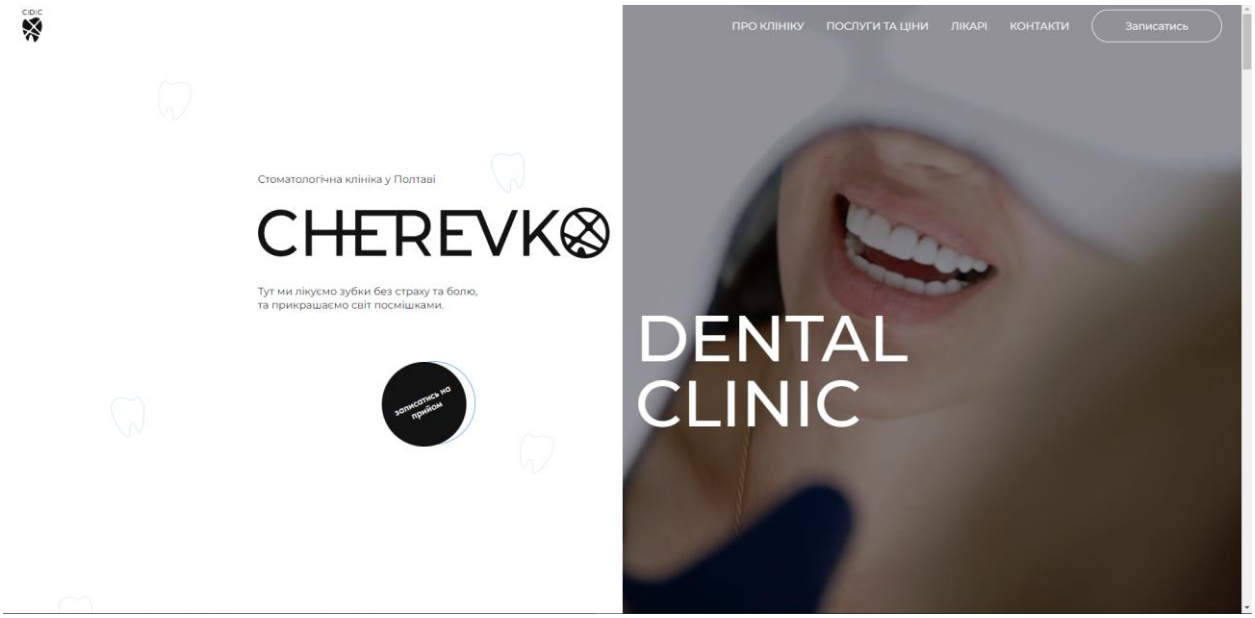

Рисунок 1.1 – Інтерфейс сайту cherevkodentalclinic.com

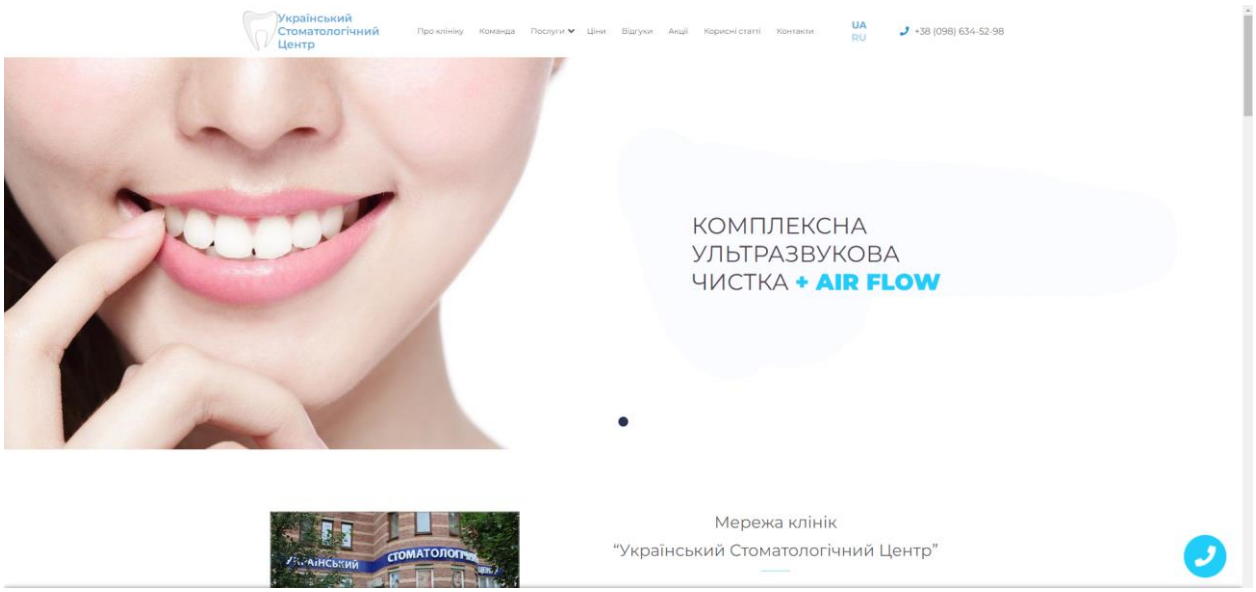

Рисунок 1.2 – Інтерфейс сайту ukrstom-centr.com.ua

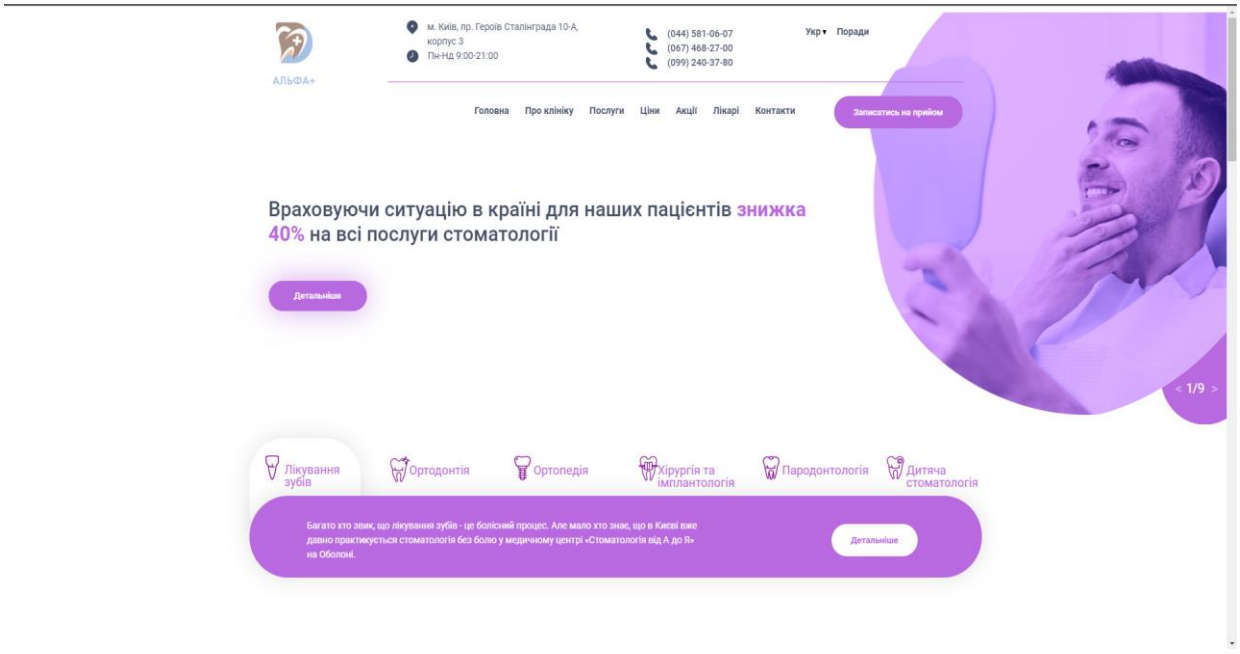

Рисунок 1.3 – Інтерфейс сайту stomatclinica.com.ua

Таблиця 1.1 – Порівняльний аналіз аналогів стоматологічних клінік (оцінки по 5-ти бальній шкалі)

Критерії	Сайт		
	cherevkodentalclinic.com	ukrstom-centr.com.ua	stomatclinica.com.ua
Зручність використання сайту	2	4	4
Дизайн сайту	1	3	3
Інформація про найближчі стоматологічні клініки	4	3	3
Коментарі інших людей	5	2	3
Фільтр областей, міст, районів	5	5	3
Наявність особистого кабінету	3	3	3
Пошук процедуру	3	4	1
Детальний опис процедур	4	4	2
Наявність онлайн-консультанта	2	5	2
Наявність історії компанії	2	4	3

Згідно з результатами проведеного аналізу аналогів можна виокремити деякі переваги та недоліки, посилаючись на які, буде створено сайт.

Переваги веб-сайтів аналогів:

- Детальна інформація про компанію та послуги
- Зовнішній вигляд та зручний функціонал сайту
- Можливість перейти за посилання на потрібну інформацію
- Наявність розділу з найпопулярнішими питаннями та відповідями
- Можливість пошуку інформації
- Авторизація через особистий кабінет

Недоліки веб-сайтів аналогів:

- Не виявлено

1.4 Аналіз принципів побудови інтернет-сайту

Дизайн головної сторінки є ключовим, оскільки споживачі в першу чергу бачать саме її. Тому дуже важливо організувати сторінки стоматологічної клініки в потрібному стилі зі збереженням усіх зручностей і простоти [4].

Дизайн інтернет-сайту відіграє ключову роль у створенні першого враження і залученні користувачів. На початковому етапі важливо визначити цільову аудиторію та врахувати її потреби і очікування при розробці дизайну. Колірна палітра, шрифти, графічні елементи та композиція повинні бути відповідними концепції бренду або тематиці сайту. Крім того, дизайн повинен бути естетичним, легким для сприйняття і візуально привабливим, з урахуванням зручності навігації та інтуїтивно зрозумілих елементів інтерфейсу. Оптимізація сайту для різних пристроїв і роздільних здатностей екранів також є важливою складовою доброго дизайну, щоб забезпечити зручність використання для користувачів на будь-яких пристроях.

Для досягення успіху треба впроваджувати більше унікального контенту і корисних деталей, які напряму пов’язані зі стоматологічної галуззю. Аналізуючи конкурентів, приходимо до висновку, що готовий сайт стоматології може бути добре оформлений, простий у використанні, але через відсутність такої корисної сторінки «Відгуки» він буде значно програвати сайтам-суперникам.

Розробка сторінки зі статистикою найактуальніших хвороб зубів або причин приходу пацієнтів до стоматолога додасть інтересу і жвавості на ваш сайт стоматології. Додаткова також корисна можливість ведення особистого медичного блогу, де лікар буде ділитися особистим досвідом роботи стоматолога з відвідувачами сайту або давати їм різноманітні поради.

Для користувача необхідно забезпечити, щоб шлях на сайті був максимально легкий і простий. Тому треба подбати про те, щоб пацієнт не

шукав годинами контакти або цінову політику, бо це тільки відштовхує і підвищує ймовірність втрати потенційного споживача.

Підвищення конкурентоспроможності перед конкурентами забезпечується мобільною адаптивністю сайту. Без заздалегідь передбаченої версії для мобільних пристроїв, якісна робота сайту неможлива, незважаючи на його високу якість розробки. Важливо завжди враховувати, що понад 60% дорослих активно користуються мобільними пристроями.

Білий – ідеальний сайт для абсолютно будь-якого проєкту. Він підкреслює медичну чистоту і ніколи не псує дизайн.

Синій – заспокійливий колір. Його використовують, коли хочуть викликати довіру до послуги. Підкреслює зв’язок з природою.

Наступний етап – організація структури. Тільки правильно підібравши її, можна забезпечити комфорт у використанні, простоту і хороше враження від ясності вкладок і сторінок сайту.

Важливі розділи

1. Сторінка "Про нас" у медичній інформаційній системі стоматологічної клініки надає корисну інформацію про клініку і команду фахівців. На цій сторінці можна знайти опис мети і цінностей клініки, а також детальну інформацію про досвід роботи стоматологів. Крім того, тут можуть бути вказані кваліфікації та освіта лікарів, що дозволяє клієнтам оцінити компетентність та довіритися клініці зі своїм здоров'ям.

2. Сторінка "Лікарі" в МІС стоматологічної клініки надає важливу інформацію про медичний персонал клініки. Ця сторінка пропонує докладні профілі лікарів, які включають їх освіту, спеціалізацію, досвід роботи та професійні досягнення. Клієнти мають змогу ознайомитися з біографіями лікарів, що дозволяє їм зробити інформований вибір при виборі свого стоматолога. Крім того, сторінка може містити фотографії лікарів, що додатково сприяє створенню довіри та зручності для клієнтів.

3. Сторінка «Послуги». Необхідно надати повний список послуг , що надаються вами на сайті, в доступній формі пояснивши що і навіщо потрібно в клініці.

4. Сторінка «Вартість послуг». Обов’язковий пункт, адже клієнти заздалегідь повинні знати, чи потягне їх бюджет такі витрати на стоматолога.

5. Сторінка «Контакти». Зручна вкладка з робочим контактним номер телефону, поштою та іншою корисною інформацією стоматологічної клініки.

Важливий функціонал:

1. Онлайн-запис клієнтів. Без цього нікуди. Дуже корисно, адже допоможе уникнути черг пацієнтів і неприємних ситуацій під дверима лікарів якісної стоматології.

2. Онлайн-чат. Клієнти зможуть дізнатися будь-яку важливу інформацію або проконсультуватися.

3. Зворотний дзвінок. Допоможе просуванню сайту стоматології, адже це значно підвищує довіру, особливо коли питання, яке хвилює клієнта, вирішено.

На ринку існує велика кількість рішень для створення веб-сайтів стоматологічних клінік, але, на жаль, не завжди вони є досконалими. Розробнику може знадобитися значний час для вивчення та освоєння таких рішень, і навіть після цього не гарантується, що він зможе створити якісний сайт з першої спроби.

Tilda Publishing є простим і легким у використанні. Завдання полягає лише в наповненні доступних шаблонів контентом, налаштуванні колірної гами та оформлення, а також внесенні редагувань відповідно до власної структури. Це дозволяє значно зекономити час і гроші [5].

WordPress є ще одним варіантом для створення сайту стоматології. Завдяки цій платформі, можна придбати готовий шаблон і налаштувати його під потрібну структуру. Вибір шаблонів може бути платним або безкоштовним.

Після встановлення шаблону, можна додавати різноманітні плагіни і розпочинати просування сайту [6].

Проте, варто враховувати, що якщо створювати сайт з нуля, без використання готових шаблонів, це буде більш витратним за часом і коштами процесом.

Незалежно від вибору платформи, проведення верстки є обов'язковою складовою створення сайту. Вона відповідає за кінцевий вигляд веб-сторінки у відповідності з готовим макетом. Верстка включає ряд особливостей і гарантує правильне та привабливе відображення вмісту сайту в усіх популярних браузерах.

Після створення сайту стоматології важливо провести тестування, щоб переконатися, що всі функції працюють належним чином. Ця процедура допоможе виявити можливі проблеми і внести необхідні виправлення, а також переконатися, що сайт виконує всі свої завдання.

Фінальним етапом роботи треба розглянути канали привертання трафіку на сайт і перетворення його в активних клієнтів. Це є головним завданням для успішного функціонування проєкту. Проте, важливо пам’ятати, що правильне виконання всіх попередніх етапів допоможе спростити процес просування сайту стоматології. Якісний та готовий сайт завжди знаходить свою аудиторію та залучає клієнтів.

1.5 Постановка задачі

Головною метою проєкту є розробка веб-сайту, який надасть клієнтам зручний доступ до інформації про компанію, ціни та послуги, а також детальний опис кожної послуги. Для забезпечення актуальності інформації на сайті, важливо врахувати можливості адміністратора.

Адміністратор повинен мати доступ до зручного інтерфейсу, що дозволить легко редагувати, видаляти та додавати нові послуги. Він повинен

мати можливість оновлювати ціни та інші деталі про послуги в режимі реального часу, забезпечуючи клієнтів актуальною інформацією.

Додатково, для ефективного управління веб-сайтом, адміністратор повинен мати доступ до журналу змін, де будуть фіксуватися дата, час та характер кожної внесеної зміни. Це дозволить відстежувати та контролювати всі дії, виконані адміністратором.

Забезпечення адміністратора зручним інтерфейсом та можливостями оновлення і контролю допоможе зберегти актуальність і якість інформації на сайті, забезпечуючи задоволення клієнтів.

2 ВИБІР МЕТОДІВ РІШЕННЯ ЗАДАЧІ

2.1 Огляд використаних інструментів

Вибір методів рішення задачі є важливим етапом у розробці проєкту, особливо коли використовуються такі технології, як Django, Bootstrap, HTML, CSS і JS. Кожен з цих інструментів має свої особливості і може бути застосований для вирішення поставлених завдань.

Django є потужним фреймворком, який забезпечує швидку розробку веб-додатків та забезпечити зручне управління базами даних. Він надає ряд вбудованих функцій і зручний синтаксис, що врази спрощує розробку [1]. Bootstrap надає набір готових компонентів та шаблонів, що дозволяє швидко розробляти естетичний і респонсивний веб-дизайн. Використання Bootstrap спрощує процес створення привабливого та користувачам зручного інтерфейсу для проєкту [3].

HTML, CSS і JavaScript є основними мовами розробки веб-сторінок [8][9]. HTML використовується для створення структури веб-сторінок, CSS – для надання контенту стилів та вигляду, а JavaScript – для додавання інтерактивності та функціональності [8]. Враховуючи всі ці інструменти, при виборі методів рішення задачі варто зосередитись на їхніх особливостях та можливостях. Потрібно врахувати потреби проєкту, його функціональність та зручність використання для користувачів.

PyCharm є інтегрованим середовищем розробки (IDE) для мови програмування Python, яке може бути використане для ефективної розробки веб-додатків, включаючи проєкти, побудовані на Django [7]. Один з ключових аспектів PyCharm – це його потужний набір функцій, які допомагають розробникам писати якісний код. IDE надає автодоповнення, розумне відступлення, підсвічування синтаксису та багато інших корисних функцій, що полегшують написання коду та зменшують кількість можливих помилок.

PyCharm також забезпечує розширену підтримку Django і має можливість автоматичного визначення шляхів до шаблонів, моделей та контролерів Django, що дозволяє зручно переміщатися по проєкту. Крім того, середовище підтримує інструменти для міграції бази даних, керування пакетами та інші корисні функції, специфічні для Django.

PyCharm також має вбудовану систему контролю версій, що дозволяє розробникам зручно працювати з репозиторіями, такими як Git, і використовувати інші розповсюджені інструменти для розробки програмного забезпечення [10]. Загалом, PyCharm є потужним інструментом для розробки проєктів на основі Python, зокрема веб-додатків на Django. Його функціональність, зручний інтерфейс та підтримка Django допомагають розробникам писати якісний код та забезпечувати ефективну розробку програмного забезпечення.

2.2 Моделювання бази даних

ERD (Entity-Relationship Diagram) є графічним інструментом для моделювання структури бази даних, включаючи зв'язки між сутностями та їх атрибутами. Використовуючи ERD, можна візуалізувати та описати взаємозв'язки між таблицями бази даних [2]. (рис.2.1).

ERD діаграма для нашого проєкту буде включати такі сутності:

1. User: Абстрактна базова модель Django, яка представляє користувача. Має атрибути, такі як email, username, is_active тощо.
2. Dentist: Підклас моделі User, що представляє дантиста. Має додатковий атрибут image.
3. Employee: Підклас моделі User, що представляє працівника.
4. Customer: Підклас моделі User, що представляє клієнта. Має додаткові атрибути first_name, last_name, phone та message.

5. Appointment: Модель, що представляє запис на прийом. Має атрибути date, time, dentist та customer, які є зв'язками з моделями Dentist та Customer відповідно.

6. Procedure: Модель, що представляє процедуру. Має атрибути name, price, brief_description, description, duration, recovery_time та procedure_fill_text.

7. AppointmentDetail: Модель, що представляє деталі запису на прийом, включаючи зв'язки з багатьма процедурами.

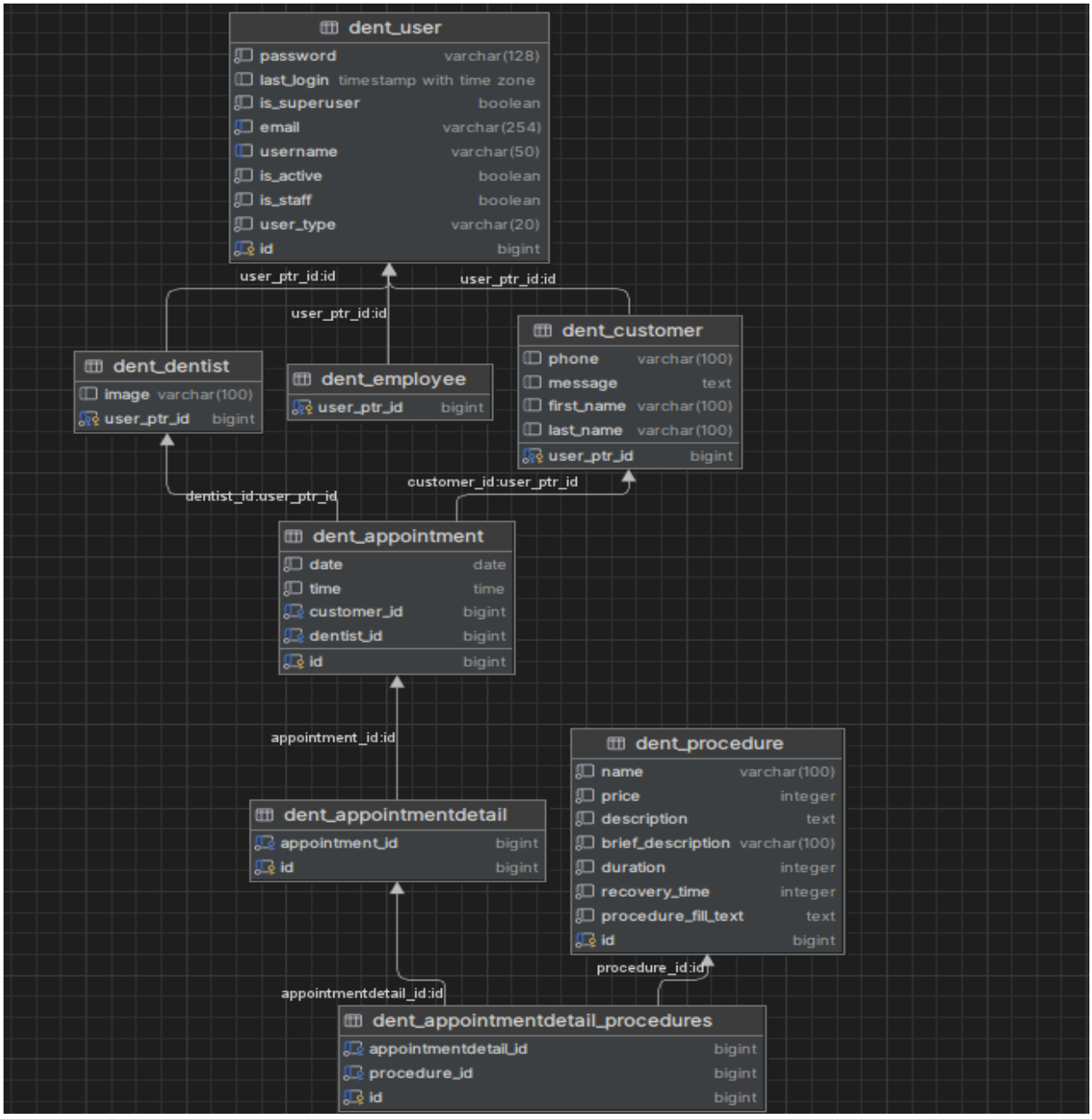

Рисунок 2.1 – ERD діаграма

ERD діаграма відображає зв'язки між цими сутностями, такі як зв'язок батьківської моделі (Dentist, Employee, Customer) з базовою моделлю User, зв’язок Appointment з моделями Dentist та Customer, зв'язок AppointmentDetail з Appointment та Procedure тощо.

ERD діаграма важлива, оскільки:

1. Вона дозволяє візуалізувати структуру бази даних і зв'язки між таблицями, що полегшує розуміння та розвиток програмного продукту.
2. Допомагає виявити проблеми та неузгодженості в моделі даних, такі як некоректні зв'язки або атрибути.
3. Сприяє ефективному проєктуванню та розробці програмного продукту шляхом визначення потреб у таблицях, зв'язках та атрибутах.
4. Становить документацію для розробників та команди проєкту, що допомагає зберігати та передавати знання про структуру бази даних.

Деякі більш конкретні моменти ERD діаграми на основі даного коду:

- User є базовою моделлю, а Dentist, Employee та Customer є підкласами моделі User, що відображає їх специфічні атрибути та функціональність.
- Dentist та Employee не мають додаткових атрибутів порівняно з базовою моделлю User.
- Customer має додаткові атрибути first_name, last_name, phone та message.
- Appointment має зв’язки з моделями Dentist та Customer, що відображають, який дентист та клієнт призначені для запису на прийом.
- AppointmentDetail має зв’язки з Appointment та Procedure, що дозволяє відстежувати, які процедури призначені для конкретного запису на прийом.

ERD діаграма надає повну картину структури бази даних та зв'язків між таблицями, допомагаючи краще розуміти взаємозв'язки між моделями та їх

атрибутами. Вона є важливим інструментом для розробників, архітекторів та команди проєкту при розробці та підтримці програмного продукту.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

Технічне завдання для розробки MIC стоматологічної клініки:

1. Реєстрація користувача

- Користувач може зареєструватися на сайті, вказавши свою електронну пошту та пароль.
- Подяка реєстрації буде відправлена на вказану електронну пошту.
- Користувач може скинути пароль, якщо він забув його.

2. Послуги клініки

- Користувач може переглянути всі доступні послуги клініки.
- Кожна послуга повинна мати детальний опис та вартість.

3. Запис на процедури

- Робітники клініки можуть створювати записи на процедури для клієнтів через сайт.
- Користувач отримує підтвердження про запис на електронну пошту.
- Користувач може скасувати запис на процедуру через сайт.

На рисунку 3.1 представлено прототип головної сторінки.

Рисунок 3.1 – Прототип головної сторінки

Головна сторінка (home) використовує бібліотеку Bootstrap для тимчасової заміни стилів і демонстрації роботи додатку. Bootstrap забезпечує

готові CSS-стилі, які допомагають швидко створювати привабливий та респонсивний вигляд веб-сторінок. На головній сторінці також виводяться дані моделі "appointments" (записи) за допомогою function based view. Це означає, що уявлення (view) home приймає запити HTTP GET із сторони клієнта та відображає сторінку зі списком записів (appointments). Для отримання даних використовується метод all() моделі "Appointment", який повертає всі записи. В контексті (context) шаблону передається змінна "procedures", яка містить всі процедури. Це дозволяє використовувати їх на сторінці для відображення додаткової інформації або взаємодії з користувачем.

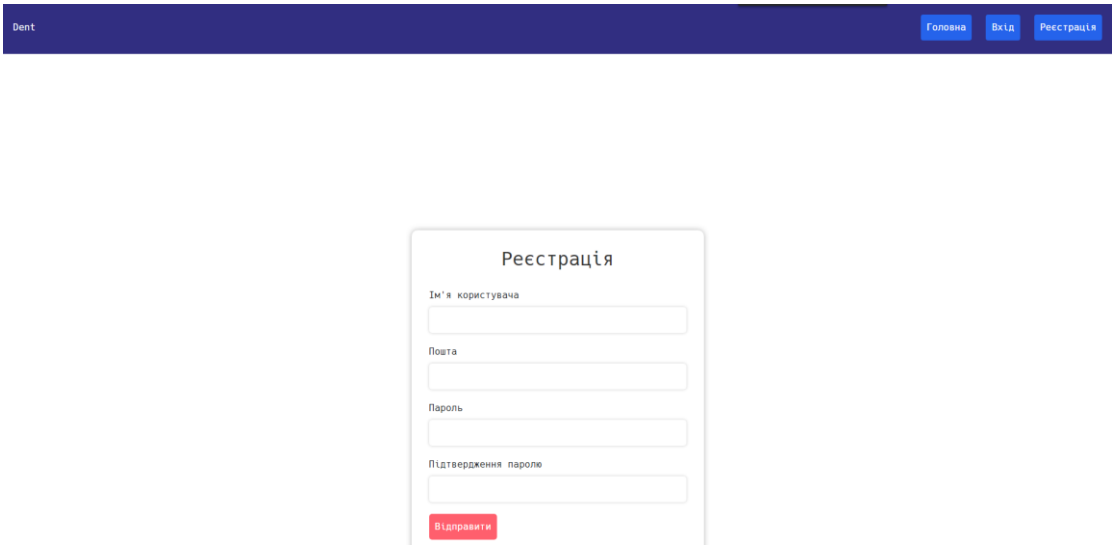

Рисунок 3.2 – Прототип сторінки реєстрації сайта

Реєстрація у Django може бути реалізована за допомогою вбудованого класу UserCreationForm. Ця форма забезпечує швидку та просту можливість реєстрації нових користувачів у системі.

UserCreationForm надає поля для введення обов'язкових даних, таких як ім'я користувача (username) та пароль (password), а також може містити інші необов'язкові поля, які можуть бути налаштовані у вашому власному класі форми.При використанні UserCreationForm в Django, вам не потрібно створювати форму з нуля або обробляти дані реєстрації вручну. Django

автоматично валідує дані форми та створює нового користувача за допомогою методу save(), який повертає створеного користувача. Наприклад, для використання UserCreationForm у вашому коді реєстрації, ви можете мати наступний фрагмент:

```python
def registration_view(request):
    if request.method == 'POST':
        form = CustomerCreationForm(request.POST)
        if form.is_valid():
            user = form.save()
            messages.success(request, 'Аккаунт було успішно створено.')

            # Generate email thanking user for registration
            subject = 'Thank you for registering!'
            message = render_to_string('verification_email.html', {'username': user.username})
            recipient_list = [user.email]

            # Write email to file instead of sending it
            with open('registration_email.txt', 'w') as f:
                f.write(message)

            return redirect('home')
        else:
            messages.error(request, 'Будь ласка, виправте помилки в формі.')
    else:
        form = CustomerCreationForm()
    return render(request, 'registration.html', {'form': form})
```

Рисунок 3.3 – Код сторінки реєстрації сайта

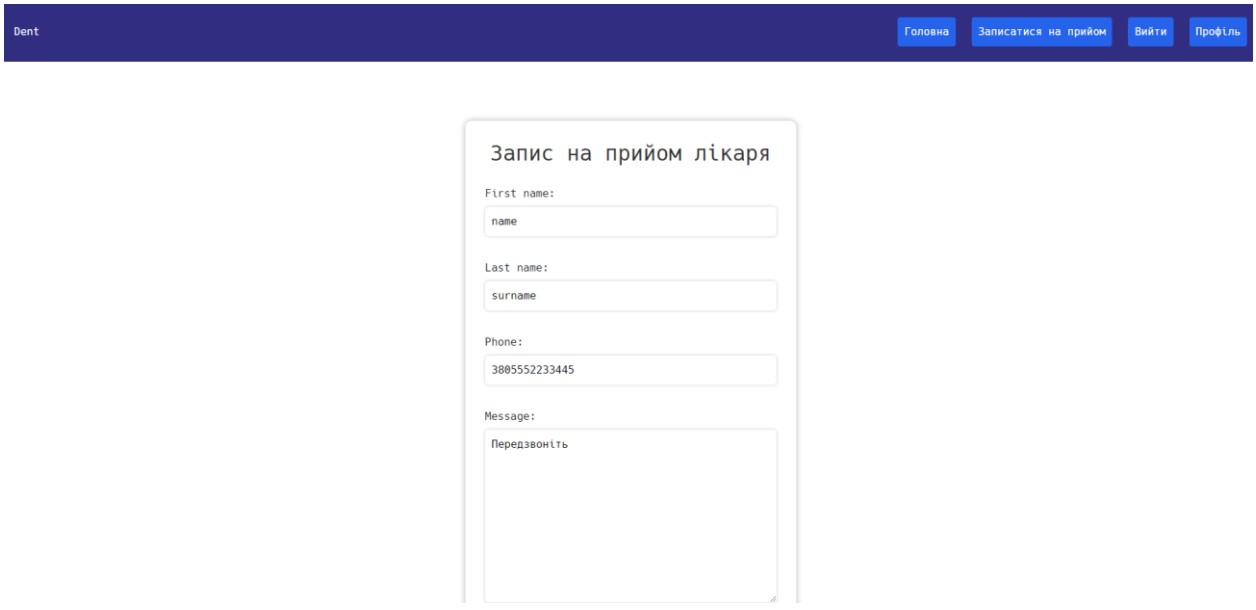

Рисунок 3.4. – Прототип сторінки запису

Після реєстрації в системі, клієнт має можливість подати запит щодо прийому до лікаря. На головній сторінці співробітників клініки відображаються всі клієнти, у яких поле "message" не є пустим. Це дозволяє співробітникам бачити всіх клієнтів, які подали запити, та обробляти їх.

Для цього використовується функція customer_request, яка перевіряє, чи користувач аутентифікований та є клієнтом. Якщо так, то отримую екземпляр клієнта з моделі Customer за його ідентифікатором (id). Далі, якщо метод запиту є POST, створюється форма CustomerRequestForm з переданими даними запиту та екземпляром клієнта. Якщо форма є валідною, дані зберігаються, а повідомлення "Ваш запит було успішно подано." відображається за допомогою messages.success(). Після цього перенаправляємо користувача на головну сторінку ("home"). Якщо метод запиту не є POST, відображається сторінка customer_request.html з формою CustomerRequestForm, в якій передається екземпляр клієнта для відображення попередньо введених даних. Якщо користувач не є аутентифікованим або не є клієнтом, виводиться повідомлення про помилку та перенаправлення на сторінку входу в систему ("login"). Таким чином, цей функціонал дозволяє клієнтам подавати запити щодо прийому до лікаря, а співробітникам клініки бачити та обробляти ці запити, які містять не пусте поле "message".

```
def customer_request(request):
    user: User = request.user
    if user.is_authenticated and user.is_customer():
        customer = Customer.objects.get(id=request.user.id)
        user: Customer
        if request.method == 'POST':
            form = CustomerRequestForm(request.POST, instance=customer)
            if form.is_valid():
                form.save()
                messages.success(request, 'Ваш запит було успішно подано.')
                return redirect('home')
            return redirect('home')
        else:
            return render(request, 'customer_request.html', {'form': CustomerRequestForm(instance=customer)})
    else:
        messages.error(request, 'Ви повинні увійти в систему як клієнт, щоб подати запит.')
        return redirect('login')
```

Рисунок 3.5 – Код сторінки запису

Також були розставлені посилання між сторінками на відповідні елементи макетів сторінок.

По закінченню розробки проєкту, було протестовано в браузері отриманий макет:

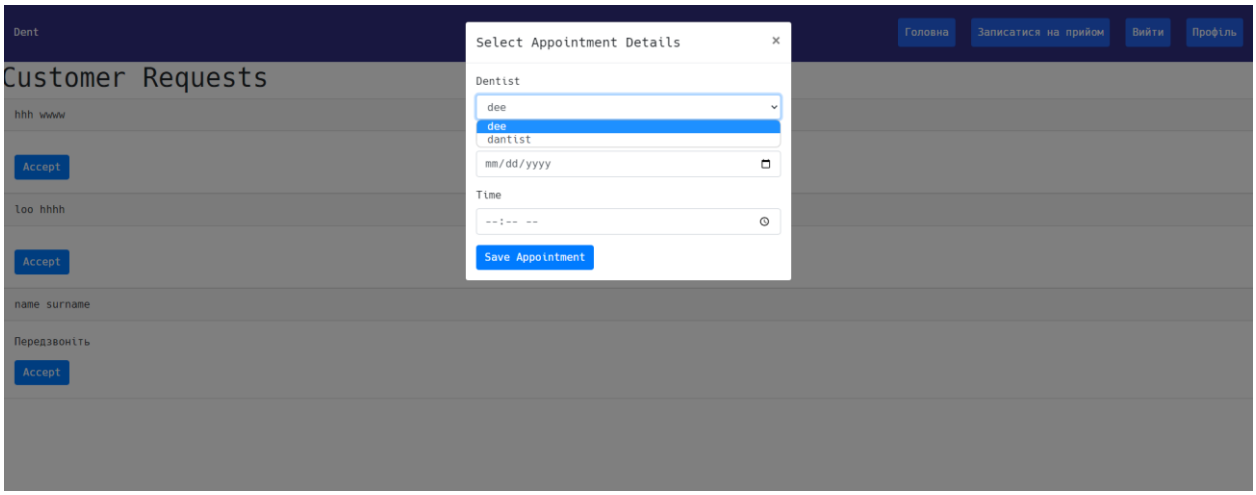

Рисунок 3.6 – Унікальний профіль робітника стоматології

```python
def profile_view(request):
    user: User = request.user
    if user.is_employee():
        customers = Customer.objects.filter(message__isnull=False)
        dentists = Dentist.objects.all()
        if request.method == 'POST':...
        return render(request, 'staff_profile.html', {'customers': customers, 'dentists': dentists})
    elif user.is_dentist():...
    else:
        appointments = Appointment.objects.filter(customer=request.user, date__gte=datetime.date.today()).order_by(
            'date', 'time')
        if request.method == 'POST':...
        return render(request, 'customer_profile.html', {'appointments': appointments})
```

Рисунок 3.7 – Унікальний профіль робітника стоматології

У даному коді реалізована функція profile_view, яка в залежності від типу користувача (клієнт, робітник, дантист) виконує певні дії.

1. Клієнт:

- Відображення списку записів на прийоми, які мають бути виконані у майбутньому та створені користувачем.

- У випадку, якщо метод запиту є POST, користувач може видалити запис на прийом.
- Після успішного видалення запису, відображається повідомлення "Запис було успішно видалено".

2. Робітник клініки:

- Відображення списку клієнтів, у яких поле "message" не є пустим. Це означає, що вони подали запит на прийом до лікаря.
- Відображення списку лікарів (дантистів).
- У випадку, якщо метод запиту є POST, робітник може створити новий запис на прийом для обраного клієнта та лікаря.
- Після успішного створення запису, відображається повідомлення "Запис було успішно створено".

3. Дантист:

- Відображення списку записів на прийоми, які належать дантисту та ще не мають деталей або процедур.
- Відображення списку доступних процедур.
- Відображення списку деталей прийому, які належать дантисту.
- У випадку, якщо метод запиту є POST, дантист може додати процедури до обраного запису на прийом.
- Після успішного додавання процедур, відображається повідомлення "Процедури було успішно додано".

Для кожного типу користувача виконується відповідний блок коду, який відповідає за відображення та обробку даних залежно від їх ролі в системі.

Деякі елементи інтерфейсу були змінені для поліпшення вигляду та користувацького досвіду. Ось опис внесених змін:

1. Головна сторінка:

- Було видалено використання фреймворку Bootstrap для стилізації елементів. Замість цього були використані власні CSS-стилі для оформлення.

- Додана бібліотека FontAwesome, яка надає велику колекцію іконок. Це дозволяє використовувати привабливі та візуально змінні іконки в різних частинах інтерфейсу.

2. Блок новин:

- Був доданий блок новин, який може містити актуальну інформацію, повідомлення або оголошення.
- Цей блок може бути розташований на головній сторінці або на інших сторінках, де він є потрібним.

3. Анімації для процедур:

- Для поліпшення візуального ефекту та привабливості, були додані анімації для процедур. Це може включати плавні переходи, зміну кольорів, появу або зникнення елементів тощо.
- Ці анімації можуть додати динаміки та привернути увагу користувача до важливих елементів.

4. Змінений детальний перегляд процедури:

- Детальний перегляд процедури був змінений за допомогою стилів, що дозволяють краще виділити цю інформацію та забезпечити зручне представлення.
- Ці зміни можуть включати використання власних CSS-стилів, розміщення елементів у зручному форматі, візуальне виділення ключової інформації тощо.

Усі ці зміни мають на меті поліпшити вигляд інтерфейсу, забезпечити більшу функціональність та полегшити взаємодію користувачів з додатком.

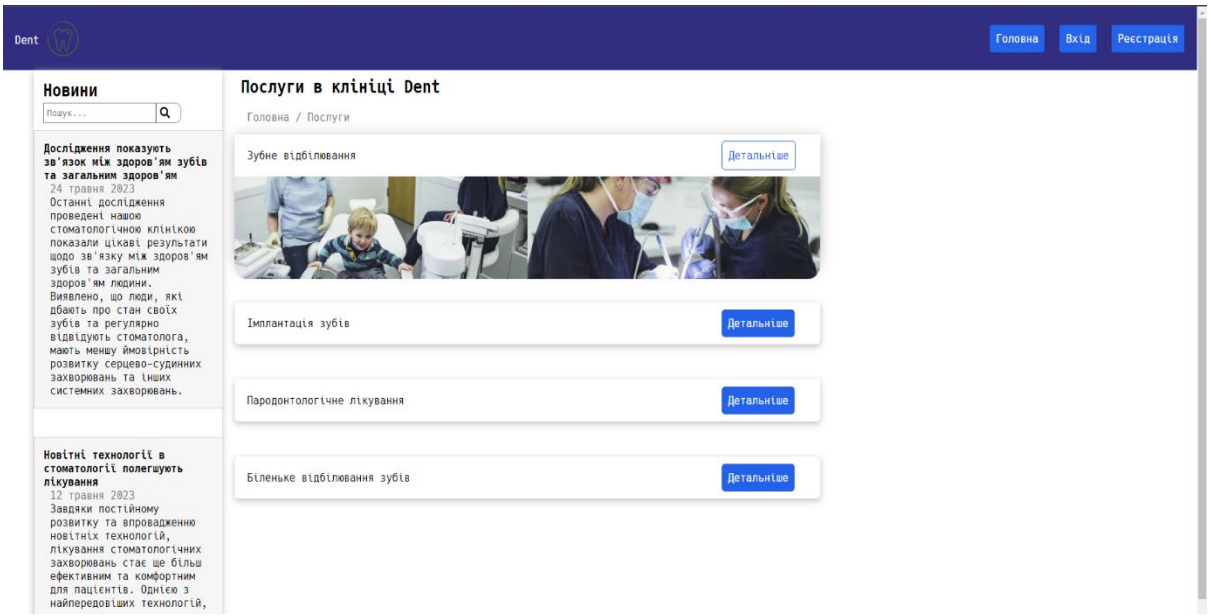

Рисунок 3.8 – Змінений вигляд головної сторінки

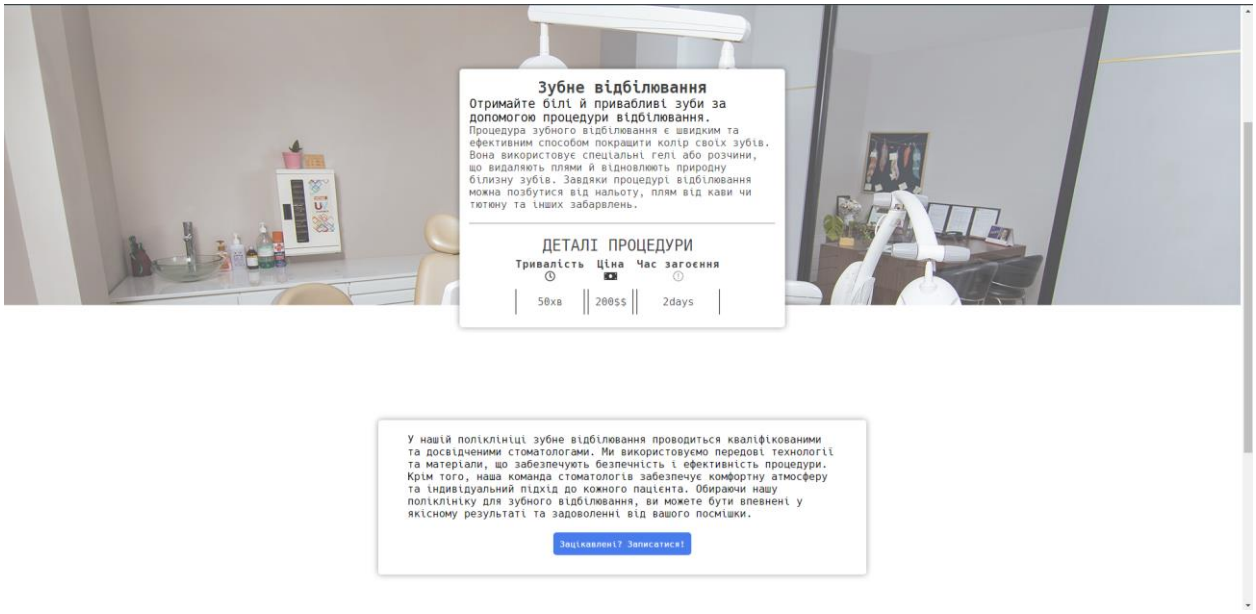

Рисунок 3.9 – Змінена сторінка детального огляду процедури

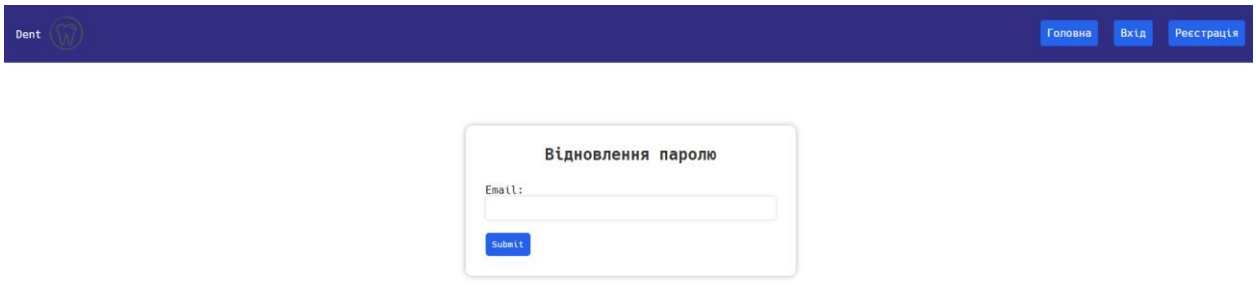

Рисунок 3.10 – Відновлення паролю

Додавання можливості відновлення пароля є важливою функціональністю для забезпечення безпеки та зручності користувачів. Процес відновлення пароля може бути реалізований наступним чином:

1. Користувач натискає на посилання "Забули пароль?" або еквівалентну кнопку на сторінці входу в систему.
2. Користувач вводить свою зареєстровану електронну пошту в поле форми.
3. Сервер перевіряє, чи існує користувач з такою електронною поштою в системі.
4. Якщо користувач існує, генерується унікальний токен (наприклад, за допомогою бібліотеки secrets у Python).
5. Токен пов’язується зі зареєстрованою електронною поштою та зберігається в базі даних або в кеші сервера.
6. Сервер надсилає електронний лист на вказану електронну пошту з унікальним посиланням, яке містить токен.
7. Користувач отримує електронний лист і натискає на посилання.
8. Клієнтська сторінка перенаправляється на сторінку зміни пароля та передається унікальний токен.
9. Сервер перевіряє, чи токен є дійсним і чи не минув його термін дії.
10. Якщо токен є дійсним, користувач може ввести новий пароль.
11. Сервер зберігає новий пароль та зберігає зміни в базі даних.
12. Користувач успішно змінює пароль і може увійти в систему з новим паролем.

```
def password_recovery(request):
    if request.method == 'POST':
        form = PasswordResetForm(request.POST)
        if form.is_valid():
            email = form.cleaned_data['email']
            user = User.objects.filter(email=email).first()
            if user is not None:
                token = default_token_generator.make_token(user)
                uid = urlsafe_base64_encode(force_bytes(user.pk))
                current_site = get_current_site(request)
                email_contents = render_to_string('password_recovery_email.txt', {
                    'user': user,
                    'domain': current_site.domain,
                    'uid': uid,
                    'token': token,
                })
                # Save email contents to a file
                save_email_to_file(email_contents)
            return HttpResponse('Password recovery email sent!')
    else:
        form = PasswordResetForm()
    return render(request, 'password_recovery.html', {'form': form})


1 usage
def save_email_to_file(email_contents):
    file_path = os.path.join(os.getcwd(), 'email.txt')
    with open(file_path, 'w') as file:
        file.write(email_contents)
```

Рисунок 3.11 – Код відновлення паролю

Важливо врахувати безпеку цього процесу, використовуючи належні методи шифрування, обмеження дійсності токена, обмеження кількості спроб зміни пароля та використання захисних механізмів проти зловживання (наприклад, CAPTCHA або обмеження швидкості надсилання запитів). Ця функціональність сприяє забезпеченню безпеки та зручності для користувачів, дозволяючи їм легко відновити свій пароль у разі його втрати або забуття.

У розробці проєкту було використано пакет django-environ, який є додатковим пакетом для фреймворку Django, що дозволяє зручно керувати конфігурацією проєкту через змінні середовища.

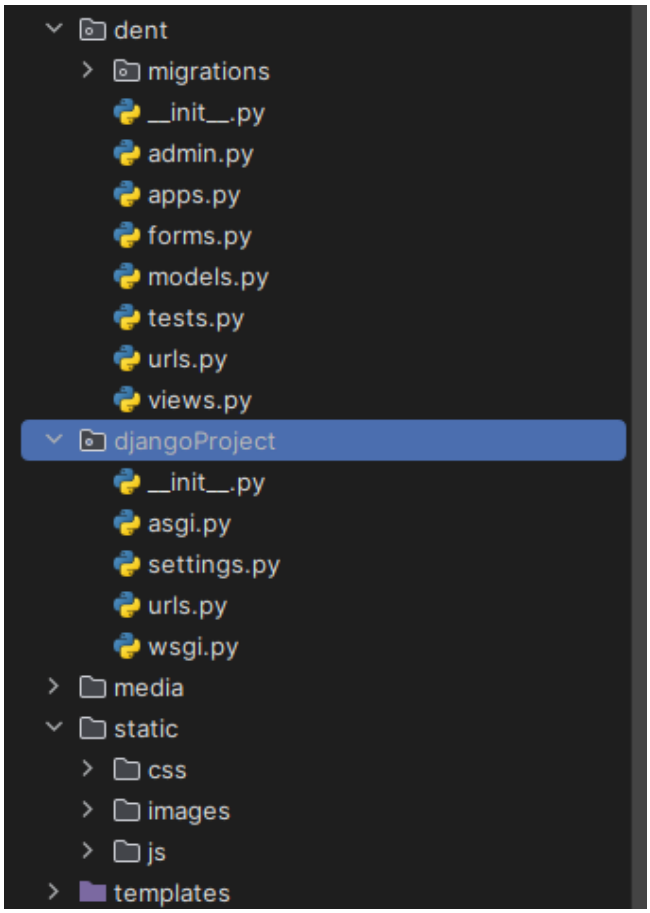

Рисунок 3.12 – Дотримання шаблона структура проєкту згідно з документацією Django

За допомогою django-environ, можна зберігати конфігураційні параметри, такі як налаштування бази даних, секретні ключі, URL-адреси сторонніх сервісів та інші, у змінних середовища. Це дозволяє підтримувати чутливі дані окремо від коду і дозволяє змінювати конфігурацію в залежності від оточення, такого як локальний режим розробки або продакшн-середовище.

Django-environ також надає зручні методи для отримання значень конфігураційних параметрів у коді Django. Це дозволило легко доступитись до налаштувань та використовувати їх у різних частинах мого проєкту.

За допомогою django-environ, було забезпечено більшу гнучкість та безпеку у керуванні конфігурацією проєкту, що є важливим аспектом розробки веб-додатків на Django.

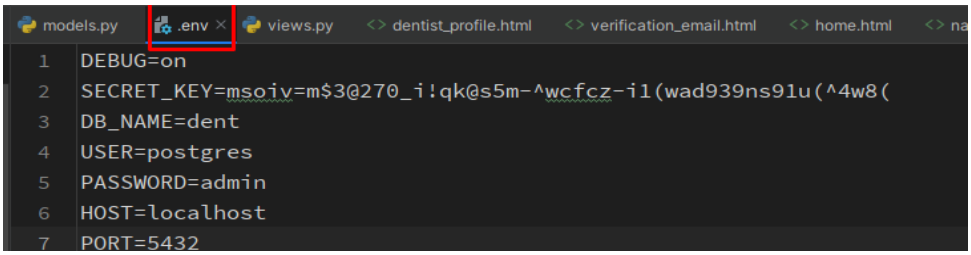

Рисунок 3.13 – Демонстрація файлу основних змінних середовища

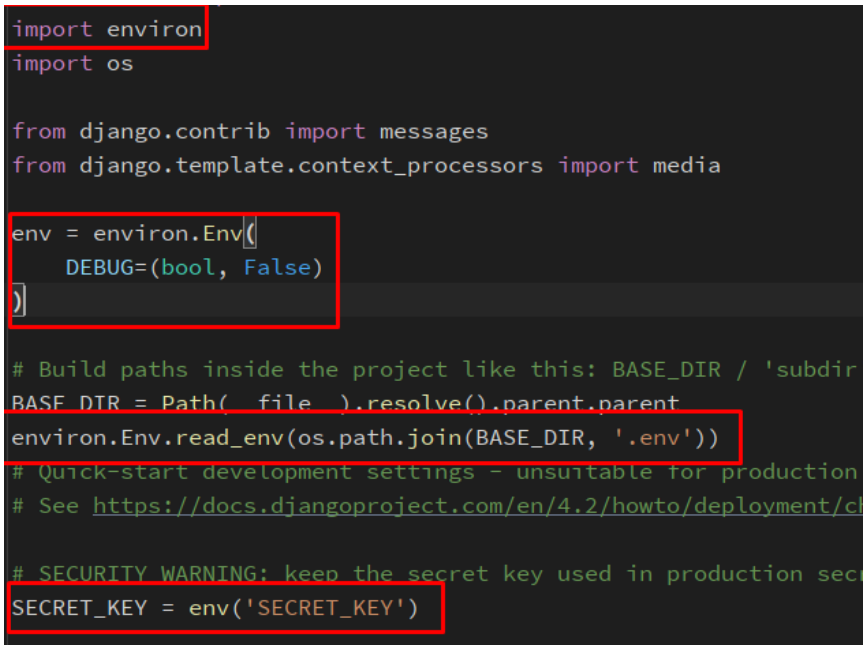

Рисунок 3.14 – Демонстрація процесу використання змінних середовища за допомогою environ пакету

На цьому можна вважати закінченим процес розробки веб-ресурсу. Повний лістинг коду знаходиться в Додатку А.

ВИСНОВКИ

У результаті роботи над проєктом стоматологічної клініки "Dent", можна зробити такі висновки.

1. Проєкт «Dent» було успішно реалізовано, використовуючи різні інструменти і технології.

2. Для розробки веб-ресурсу було використано мову програмування Python, фреймворк Django, а також мови розмітки HTML та CSS. Django дозволив швидко створити потужний та функціональний веб-додаток, який відповідає потребам стоматологічної клініки.

3. За допомогою фреймворку Django реалізовано можливість клієнтам здійснювати запис на прийоми, переглядати інформацію про послуги та отримувати додаткові послуги, а робітникам клініки – керувати записами і надавати необхідні послуги.

4. Інтерфейс веб-ресурсу був розроблений з використанням HTML та CSS, що дозволило створити привабливий та зручний для користувачів дизайн.

5. Результатом роботи є повноцінний веб-ресурс стоматологічної клініки «Dent», який відповідає поставленим завданням і задовольняє потреби клієнтів та робітників клініки. Проєкт підтримує реєстрацію користувачів, управління записами на прийоми, надання інформації про послуги, а також має зручні інтерфейси для різних типів користувачів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Джерело: "The Django Project - Web framework for perfectionists with deadlines", https://www.djangoproject.com/
2. Джерело: Elmasri, R., & Navathe, S. B. (2010). Fundamentals of database systems. Pearson Education.
3. Джерело: Bootstrap. (2021). Офіційна документація Bootstrap. Отримано з https://getbootstrap.com/
4. Джерело: Mantri, A., & Ramesh, A. (2021). Web Design Principles and Guidelines for User Experience. International Journal of Engineering Research & Technology, 10(8), 2770-2775.
5. Джерело: "Tilda Publishing – Easy website builder for businesses and creatives", https://tilda.cc/
6. Джерело: "WordPress.org - Open source software for creating websites", https://wordpress.org/
7. Джерело: "PyCharm: Python IDE for Professional Developers", https://www.jetbrains.com/pycharm/
8. Джерело: Duckett, J. (2011). HTML & CSS: Design and Build Websites. John Wiley & Sons.
9. Джерело: Flanagan, D. (2006). JavaScript: The Definitive Guide. O'Reilly Media..
10. Джерело: "Version Control with PyCharm", https://www.jetbrains.com/pycharm/features/version_control.html
11. Alchin, M., 2013. Pro Django. Apress.
12. Vincent, W.S., 2022. Django for Beginners: Build websites with Python and Django. WelcomeToCode.
13. Bennett, J., 2009. Practical Django Projects. Apress.
14. Dauzon, S., Bendoraitis, A. and Ravindran, A., 2016. Django: web development with Python. Packt Publishing Ltd.

15. Ghimire, D., 2020. Comparative study on Python web frameworks: Flask and Django.

ДОДАТОК А

Лістинг коду

views.py

```
import datetime
from django.contrib import messages
from django.contrib.auth import login, authenticate, logout

from django.contrib.auth.forms import PasswordResetForm
from django.contrib.auth.tokens import default_token_generator
from django.contrib.sites.shortcuts import get_current_site
from django.http import HttpResponse
from django.utils.http import urlsafe_base64_encode
from django.utils.encoding import force_bytes
from django.contrib.auth.models import User
import os
from django.shortcuts import render, redirect
from .forms import CustomerCreationForm, CustomerRequestForm
from dent.models import Dentist, User, Customer, Appointment, Procedure,
AppointmentDetail
from django.template.loader import render_to_string
from django.db.models import Q


def registration_view(request):
  if request.method == 'POST':
      form = CustomerCreationForm(request.POST)
      if form.is_valid():
          user = form.save()
          messages.success(request, 'Аккаунт було успішно створено.')
```

```
                # Generate email thanking user for registration
                subject = 'Thank you for registering!'
                message = render_to_string('verification_email.html', {'username': user.username})
                recipient_list = [user.email]

                # Write email to file instead of sending it
                with open('registration_email.txt', 'w') as f:
                    f.write(message)

                return redirect('home')
            else:
                messages.error(request, 'Будь ласка, виправте помилки в формі.')
        else:
            form = CustomerCreationForm()
        return render(request, 'registration.html', {'form': form})


    def profile_view(request):
        user: User = request.user
        if user.is_employee():
            customers = Customer.objects.filter(message__isnull=False)
            dentists = Dentist.objects.all()
            if request.method == 'POST':
                appointment = Appointment.objects.create(
                    customer=Customer.objects.get(id=request.POST['customer_id']),
                    dentist=Dentist.objects.get(id=request.POST['dentist']),
                    date=request.POST['appointment-date'],
```

```
                    time=request.POST['appointment-time'],
                )
                appointment.customer.message = None
                appointment.customer.save()
                messages.success(request, 'Запис було успішно створено.')
                message = render_to_string('appointment_verification_email.html',
{'username': user.username})
                with open('registration_email.txt', 'w') as f:
                    f.write(message)
                return redirect('home')
            return render(request, 'staff_profile.html', {'customers': customers,
'dentists': dentists})
        elif user.is_dentist():
            appointments = Appointment.objects.filter(
                Q(appointmentdetail__isnull=True) |
                Q(appointmentdetail__procedures=None),
                dentist=request.user,
                date__gte=datetime.date.today()
            ).order_by('date', 'time')
            procedures = Procedure.objects.all()
            appointments_detail =
AppointmentDetail.objects.filter(appointment__dentist=request.user)
            if request.method == 'POST':
                appointment_id = request.POST['appointment_id']
                appointment = Appointment.objects.get(id=appointment_id)
                selected_procedures = request.POST.getlist('procedures')
                detail = AppointmentDetail.objects.create(appointment=appointment)
                detail.procedures.set(selected_procedures)
                messages.success(request, 'Процедури було успішно додано.')
```

```
                return redirect('profile')
            return render(request, 'dentist_profile.html', {'appointments':
appointments, 'procedures': procedures,
                                                'appointments_detail':
appointments_detail})
        else:
            appointments = Appointment.objects.filter(customer=request.user,
date__gte=datetime.date.today()).order_by(
                'date', 'time')
            if request.method == 'POST':
                appointment_id = request.POST['appointment_id']
                appointment = Appointment.objects.get(id=appointment_id)
                appointment.delete()
                messages.success(request, 'Запис було успішно видалено')
                return redirect('profile')
            return render(request, 'customer_profile.html', {'appointments':
appointments})


    def customer_request(request):
        user: User = request.user
        if user.is_authenticated and user.is_customer():
            customer = Customer.objects.get(id=request.user.id)
            user: Customer
            if request.method == 'POST':
                form = CustomerRequestForm(request.POST, instance=customer)
                if form.is_valid():
                    form.save()
                    messages.success(request, 'Ваш запит було успішно подано.')
```

```
                    return redirect('home')
                return redirect('home')
            else:
                return render(request, 'customer_request.html', {'form': CustomerRequestForm(instance=customer)})
        else:
            messages.error(request, 'Ви повинні увійти в систему як клієнт, щоб подати запит.')
            return redirect('login')


    def home(request):
        procedures = Procedure.objects.all()
        return render(request, 'home.html', {'procedures': procedures})


    def procedure_detail(request, procedure_id):
        procedure = Procedure.objects.get(id=procedure_id)
        return render(request, 'procedure_detail.html', {'procedure': procedure})


    def signup_view(request):
        if request.method == 'POST':
            form = CustomerCreationForm(request.POST)
            if form.is_valid():
                user = form.save()
                login(request, user)
                return redirect('home')
        else:
```

```
            form = CustomerCreationForm()
        return render(request, 'signup.html', {'form': form})


    def login_view(request):
        if request.method == 'POST':
            email = request.POST.get('email')
            password = request.POST.get('password')
            user = authenticate(request, email=email, password=password)
            if user is not None:
                login(request, user)
                messages.success(request, 'Ви успішно ввійшли в систему.')
                return redirect('home')
            else:
                messages.error(request, 'Невірний логін або пароль.')
        return render(request, 'login.html')


    def logout_view(request):
        logout(request)
        messages.success(request, 'Ви успішно вийшли з системи.')
        return redirect('home')


    def password_recovery(request):
        if request.method == 'POST':
            form = PasswordResetForm(request.POST)
            if form.is_valid():
                email = form.cleaned_data['email']
```

```
            user = User.objects.filter(email=email).first()
            if user is not None:
                token = default_token_generator.make_token(user)
                uid = urlsafe_base64_encode(force_bytes(user.pk))
                current_site = get_current_site(request)
                email_contents = render_to_string('password_recovery_email.txt', {
                    'user': user,
                    'domain': current_site.domain,
                    'uid': uid,
                    'token': token,
                })
                # Save email contents to a file
                save_email_to_file(email_contents)
            return HttpResponse('Password recovery email sent!')
    else:
        form = PasswordResetForm()
    return render(request, 'password_recovery.html', {'form': form})


def save_email_to_file(email_contents):
    file_path = os.path.join(os.getcwd(), 'email.txt')
    with open(file_path, 'w') as file:
        file.write(email_contents)
```

urls.py

```
from django.urls import path, include
from . import views

urlpatterns = [
```

```
        path('', views.home, name='home'),
        path('signup/', views.signup_view, name='signup'),
        path('login/', views.login_view, name='login'),
        path('logout/', views.logout_view, name='logout'),
        path('customer_request/', views.customer_request, name='customer_request'),
        path('registration/', views.registration_view, name='registration'),
        path('profile/', views.profile_view, name='profile'),
        path('procedure/<int:procedure_id>/', views.procedure_detail, name='procedure_detail'),
        path('password-recovery/', views.password_recovery, name='password_recovery'),
        path('', include('django.contrib.auth.urls')),
    ]
```

models.py

```
from django.contrib.auth.hashers import make_password
from django.contrib.auth.models import AbstractBaseUser, BaseUserManager, PermissionsMixin
from django.db import models


class UserManager(BaseUserManager):
    def create_user(self, email, password=None, **extra_fields):
        if not email:
            raise ValueError('The Email field must be set')
        email = self.normalize_email(email)
        user = self.model(email=email, **extra_fields)
        user.set_password(password)
```

```
            user.save(using=self._db)
            return user

        def create_superuser(self, email, password, **extra_fields):
            extra_fields.setdefault('is_staff', True)
            extra_fields.setdefault('is_superuser', True)
            return self.create_user(email, password, **extra_fields)


    class User(AbstractBaseUser, PermissionsMixin):
        USER_TYPE_CHOICES = (
            ('cu', 'customer'),
            ('em', 'employee'),
            ('de', 'dentist'),

        )
        email = models.EmailField(unique=True)
        username = models.CharField(max_length=50, unique=True, blank=True,
null=True)
        is_active = models.BooleanField(default=True)
        is_staff = models.BooleanField(default=False)
        user_type = models.CharField(max_length=20,
choices=USER_TYPE_CHOICES, default='customer')

        USERNAME_FIELD = 'email'
        EMAIL_FIELD = 'email'
        REQUIRED_FIELDS = ['username']

        objects = UserManager()
```

```
        def is_employee(self):
            return self.user_type == 'employee' or self.user_type == 'em'

        def is_dentist(self):
            return self.user_type == 'dentist' or self.user_type == 'de'

        def is_customer(self):
            return self.user_type == 'customer' or self.user_type == 'cu'

        def __str__(self):
            return self.username


    class Dentist(User):
        image = models.ImageField(upload_to='dentist_images', blank=True,
null=True)


    class Employee(User):
        pass


    class Customer(User):
        first_name = models.CharField(max_length=100, blank=True, null=True)
        last_name = models.CharField(max_length=100, blank=True, null=True)
        phone = models.CharField(max_length=100, blank=True, null=True)
        message = models.TextField(blank=True, null=True)
```

```
class Appointment(models.Model):
    date = models.DateField()
    time = models.TimeField()
    dentist = models.ForeignKey(Dentist, on_delete=models.CASCADE)
    customer = models.ForeignKey(Customer, on_delete=models.CASCADE)


class Procedure(models.Model):
    name = models.CharField(max_length=100)
    price = models.IntegerField()
    brief_description = models.CharField(max_length=100)
    description = models.TextField()
    duration = models.IntegerField()
    recovery_time = models.IntegerField()
    procedure_fill_text = models.TextField()

    def __str__(self):
        return self.name


class AppointmentDetail(models.Model):
    appointment = models.ForeignKey(Appointment, on_delete=models.CASCADE)
    procedures = models.ManyToManyField(Procedure)

    def __str__(self):
        return self.appointment.customer.first_name + ' ' + self.appointment.customer.last_name
```

forms.py

```
from django import forms
from django.contrib.auth.forms import UserCreationForm
from django import forms
from django.contrib.auth.forms import PasswordResetForm as DjangoPasswordResetForm
from .models import Customer


class CustomerCreationForm(UserCreationForm):
  class Meta:
    model = Customer
    fields = ('username', 'email', 'password1', 'password2')


class CustomerRequestForm(forms.ModelForm):
  class Meta:
    model = Customer
    fields = ('first_name', 'last_name', 'phone', 'message', 'email')

  def __init__(self, *args, **kwargs):
    super().__init__(*args, **kwargs)
    self.fields['email'].disabled = True


class PasswordResetForm(DjangoPasswordResetForm):
  email = forms.EmailField(widget=forms.EmailInput(attrs={'class': 'form-control', 'placeholder': 'Email'}))
```

admin.py

```
from django.contrib import admin
from .models import User, Customer, Employee, Dentist, Appointment, AppointmentDetail, Procedure
from django.contrib.auth.admin import UserAdmin

admin.site.register(User)
admin.site.register(Customer)


class DentistAdmin(UserAdmin):
  model = Dentist
  list_display = ['username', 'user_type', 'email']
  fieldsets = (
    ('Personal Information', {'fields': ('email', 'username', 'password', 'image')}),
    ('Permissions', {'fields': ('is_active', 'is_staff', 'is_superuser', 'user_type')}),
  )
  add_fieldsets = (
    (None, {
      'classes': ('wide',),
      'fields': ('email', 'username', 'password1', 'password2', 'image'),
    }),
  )


admin.site.register(Dentist, DentistAdmin)
```

```
class EmployeeAdmin(UserAdmin):
    model = Employee
    list_display = ['username', 'user_type', 'email']
    fieldsets = (
        ('Personal Information', {'fields': ('email', 'username', 'password',)}),
        ('Permissions', {'fields': ('is_active', 'is_staff', 'is_superuser',
'user_type')}),
    )
    add_fieldsets = (
        (None, {
            'classes': ('wide',),
            'fields': ('email', 'username', 'password1', 'password2'),
        }),
    )


admin.site.register(Employee, EmployeeAdmin)
admin.site.register(Appointment)
admin.site.register(AppointmentDetail)
admin.site.register(Procedure)
```

django/settings.py

```
"""
Django settings for djangoProject project.

Generated by 'django-admin startproject' using Django 4.2.1.
```

```
For more information on this file, see
https://docs.djangoproject.com/en/4.2/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/4.2/ref/settings/
"""

from pathlib import Path
import environ
import os

from django.contrib import messages
from django.template.context_processors import media

env = environ.Env(
  DEBUG=(bool, False)
)

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent
environ.Env.read_env(os.path.join(BASE_DIR, '.env'))
# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/4.2/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = env('SECRET_KEY')

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = env('DEBUG')
```

```
ALLOWED_HOSTS = []

# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'dent.apps.DentConfig',

]

MESSAGE_TAGS = {
    messages.DEBUG: 'secondary',
    messages.INFO: 'info',
    messages.SUCCESS: 'success',
    messages.WARNING: 'warning',
    messages.ERROR: 'danger',
}

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',
    ]

    ROOT_URLCONF = 'djangoProject.urls'

    TEMPLATES = [
        {
            'BACKEND': 'django.template.backends.django.DjangoTemplates',
            'DIRS': [BASE_DIR / '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',
                ],
            },
        },
    ]
    AUTHENTICATION_BACKENDS                                    =
['django.contrib.auth.backends.ModelBackend',
                           'django.contrib.auth.backends.RemoteUserBackend',

'django.contrib.auth.backends.AllowAllUsersRemoteUserBackend',
```

```
'django.contrib.auth.backends.AllowAllUsersModelBackend',
                            ]

        WSGI_APPLICATION = 'djangoProject.wsgi.application'
        AUTH_USER_MODEL = 'dent.User'
        # Database
        # https://docs.djangoproject.com/en/4.2/ref/settings/#databases

        DATABASES = {
          'default': {
              'ENGINE': 'django.db.backends.postgresql',
              'NAME': env('DB_NAME'),
              'USER': env('USER'),
              'PASSWORD': env('PASSWORD'),
              'HOST': env('HOST'),
              'PORT': env('PORT'),
          }
        }

        # Password validation
        #             https://docs.djangoproject.com/en/4.2/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',
        },
    ]

    # Internationalization
    # https://docs.djangoproject.com/en/4.2/topics/i18n/

    LANGUAGE_CODE = 'en-us'

    TIME_ZONE = 'UTC'

    USE_I18N = True

    USE_TZ = True

    # Static files (CSS, JavaScript, Images)
    # https://docs.djangoproject.com/en/4.2/howto/static-files/

    STATIC_URL = 'static/'
```

```
STATICFILES_DIRS = [BASE_DIR / 'static']
# Default primary key field type
# https://docs.djangoproject.com/en/4.2/ref/settings/#default-auto-field
MEDIA_URL = '/media/'
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
```

django/urls.py

```
from django.conf import settings
from django.conf.urls.static import static
from django.contrib import admin
from django.urls import path, include


urlpatterns = [
  path('admin/', admin.site.urls),
  path('', include('dent.urls')),
]
if settings.DEBUG:
  urlpatterns += static(settings.STATIC_URL, document_root=settings.STATIC_ROOT)
  urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```