МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ЦЕНТР ЗАОЧНОЇ, ДИСТАНЦІЙНОЇ ТА ВЕЧІРНЬОЇ ФОРМ НАВЧАННЯ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: «Інформаційна система для онлайн-консультацій»

за спеціальністю 122 «Комп'ютерні науки»,
освітньо-професійна програма «Інформаційні технології проектування»

Виконавець роботи: студента групи ІТ.мз-13с Пшінник Дмитро Іванович

Кваліфікаційну роботу захищено на засіданні ЕК з оцінкою _____ «___» лютого 2023 р.

Науковий керівник _____ к.т.н., доц., Федотова Н. А.
(підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Суми-2023

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

ЗАТВЕРДЖУЮ
В.о.зав. кафедри ІТ

_____ С. М. Ващенко
«___» _____ 2022 р.

ЗАВДАННЯ

на кваліфікаційну роботу магістра студентові

Пшінник Дмитро Іванович
(прізвище, ім'я, по батькові)

1 Тема проекту Інформаційна система для онлайн-консультацій

затверджена наказом по університету від «04» 11 2022 р. №1015-VI

2 Термін здачі студентом закінченого проекту «10» грудня 2022 р.

3 Вхідні дані до проекту _____

4 Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) Вступ, аналіз предметної області, постановка задачі та методи дослідження, моделювання та проектування, практична реалізація інформаційної системи, висновок, список використаних джерел, додаток А планування робіт , додаток Б код реалізації інформаційної системи.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) _____

6. Консультанти випускної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Аналіз предметної області	Федотова Н. А.		
Постановка задачі та вибір методів розробки	Федотова Н. А.		
Практична реалізація інформаційної системи	Федотова Н. А.		

Дата видачі завдання _____.

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів випускної проекту	Термін виконання етапів проекту	Примітка
1	Дослідження предметної області	07.09.22-15.09.22	
2	Аналіз конкурентів	15.09.22-21.09.22	
3	Постановка задачі	21.09.22-30.09.22	
4	Розробка дизайну	30.09.22-15.11.22	
5	Проектування бази даних	15.11.22-22.11.22	
6	Розробка каркасу та створення компонентів	22.11.22-05.12.22	
7	Розробка функціональних блоків	05.12.22-25.12.22	
8	Тестування	25.12.22-15.01.23	
9	Наповнення контентом	15.01.23-30.01.23	
10	Завантаження на сервер	30.01.23-02.02.23	

Магістрант _____ Пшінник Д.І.

Керівник роботи _____ к.т.н., доц. Федотова Н.А.

РЕФЕРАТ

Тема кваліфікаційної роботи магістра «Інформаційна система для онлайн консультацій».

Практична значення полягає у створенні інформаційної системи по створенню онлайн-консультацій та їх відвідування. Слухачі можуть планувати відвідування та мати постійний контакт з фахівцями.

В першому розділі – було проведено аналіз обраної предметної області, а також виконано огляд останніх досліджень та публікацій, порівняння з аналогами, встановлення задачі проекту.

В другому розділі – встановлено чітку мету та задачі дослідження, проведено порівняння інструментів реалізації.

В третьому розділі – створено структури інформаційної системи, проектування IDEF0 та її декомпозиції, візуалізація діаграми варіантів використання функціоналу, проектування бази даних.

В четвертому розділі – реалізовано та описано інформаційну систему як зі сторони фахівця, так і зі сторони слухача. Продемонстровано можливості адміністратора.

Результатом проведеної роботи є створена інформаційна система для онлайн-консультацій.

Кваліфікаційна робота містить 99 сторінок, 12 таблиць, 56 рисунків, список літератури 20 найменувань, 2 додатки.

Ключові слова: ІНФОРМАЦІЙНА СИСТЕМА, КОНСУЛЬТАЦІЯ, ОНЛАЙН-КОНСУЛЬТАЦІЯ, ПЛАТІЖНА СИСТЕМА, ФАХІВЕЦЬ, КОРИСТУВАЧ, СЛУХАЧ, ПЛАНУВАННЯ, API ZOOM, ВІДГУК, АДМІНІСТРАТОР, КАЛЕНДАР, КОМУНІКАЦІЯ, БАГАТОНАПРАВЛЕННІСТЬ, ПРОЕКТ.

ЗМІСТ

ВСТУП

Проблема отримання відповідей на цілком конкретні запитання або отримання пояснення від фахівця на певні галузеві аспекти почала набирати обертів. Зустрічі в реальному житті перенеслися в інформаційне поле. Перейшовши в інформаційне поле, фахівці зникли на фоні інших людей зі своїми пропозиціями подібних послуг.

Заглядаючи наперед, потреби в консультаціях в інформаційному полі формату «онлайн» будуть збільшуватися, а проблеми накопичуватися. Користувачеві важко слідкувати за фахівцем, який проводить консультації на нерегулярній основі. Враховуючи особливості поширення інформації, фахівцеві важко зібрати аудиторію через недостачі інформаційних ресурсів, які можуть просувати цю тематику. Часто користувач забуває про заплановану консультацію, або взагалі отримує інформацію про проведення занадто пізно.

Онлайн-консультації потрібно ще прив’язати до комфортного програмного забезпечення та забезпечити сповіщення для користувачів. Програмним забезпеченням для онлайн-консультацій можливо запропонувати використовувати Zoom. А сповіщення реалізувати через e-mail. Особливу увагу потрібно звернути на пряму розмову між фахівцем та користувачем, шляхом реалізації через додавання мінімалістичного чату.

Подібна інформаційна система вже має привабливі функції як для фахівця, так і для звичайного користувача. Але потрібно для реалізації такої нелегкої задачі загострювати увагу і на інших, не менш важливих аспектах, щоб головні функції були завжди під рукою, та було інтуїтивно зрозуміло для використання.

Практичне значення інформаційної системи полягає в тому, що інформаційна система, яка направлена на розв’язання проблем онлайн-консультацій матиме особливий функціонал націлений на розв’язання питань пов’язаних з тематикою онлайн-консультацій.

Тема: «Інформаційна система для онлайн-консультацій».

Мета: розробити інформаційну систему для онлайн-консультацій як частину інформаційного поля для покращення взаємодії між фахівцем та користувачем за різними напрямками.

Об’єкт дослідження: процес зв’язку з користувачами шляхом використання інформаційної системи.

Предмет дослідження: розширення можливостей взаємодії між фахівцем та користувачем задля спрощення консультаційного сервісу в режимі онлайн в межах інформаційного простору.

Потрібно вирішити наступні завдання:

– Створити можливість реєстрації та авторизації користувачів, розділених на 2 категорії – слухачі та фахівці;

– Реалізувати можливість вести листування між слухачем та фахівцем у вигляді месенджера;

– Зробити комфортний та простий інтерфейс у вигляді календаря з інтуїтивно зрозумілим управлінням;

– Розширити функціонал календаря та додати можливість додавати нові задачі, бронювати час для зустрічі та ставити оцінку та коментар консультації.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд останніх досліджень і публікацій

Не дивлячись на те, що основи інформаційних систем майже не замінюються на рівні галузевих структур, технології не стоять на місці та постійно розвиваються.

Постійне слідкування та моніторинг змін в програмному забезпеченні – запорука успіху при вирішенні тих чи інших завдань. Потрібно звертати увагу на невеликі додаткові функції в розробці інформаційних систем, бо навіть незначний функціонал може зменшити час розробки [1].

Для спрощення розробки, прийнято розділяти програмну систему на «Front end» та «Back end», де вважається, що «Front end» – клієнтська сторона, а «Back end» – серверна. При цьому художники клієнтської сторони не повинні втручатися в роботу інженерів серверної сторони, та навпаки.

Художники та дизайнери інтерфейсу інформаційної системи, постійно стикають з проблемами покращення ергономічності інтерфейсу, мати простий дизайн який не нав’язує лишнього, а заохочує користуватися інформаційною системою. Галузі поступово розвивалися і тематики інформаційних систем теж зазнавали змін. Ріс попит на темну тематику інтерфейсу та мінімалізм в представленому функціоналу. Шрифт тексту ставав більш строгим, а кольори стали використовувати не такі химерні. Анімація зачаровувала своєю легкістю та простотою [2].

Інноваційні кроки в цьому напрямку робилися здебільшого в сторону спрощення візуалізованого представлення інформаційної системи. Це, дещо, закриває шлях для експериментів, але дизайн завжди йшов своїм шляхом, дивуючи кожного разу своєю неординарністю.

Оптимізація продуктивності інформаційної системи дуже цінується дизайнерами та веб-розробниками, оскільки вона покращує взаємодію з користувачем (UX) під час відвідування веб-сайту. Він використовує кілька методів для оптимізації інформаційної системи та скорочення часу завантаження

сторінки. Оптимізація інтерфейсу фокусується на зменшенні розміру файлу та мінімізації кількості запитів, необхідних для завантаження даної сторінки [3].

Використання мережі доставлення вмісту вибирає сервери для конкретних користувачів на основі ефективної системи доставки даних, яка наближена до мережі. Зазвичай вибирається сервер із найшвидшою відповіддю.

Інколи інші методи використовуються в оптимізації інформаційної системи та широко використовуються інженерами серверної частини [4].

Коли браузер завантажує інформаційну частину, він відкриває окреме підключення протоколу керування передачею (TCP) для кожного надісланого запиту протоколу передачі гіпертексту (HTTP). Ці запити представляють загальну кількість елементів сторінки, які необхідно завантажити. Однак браузери обмежені відкриттям певної кількості одночасних з’єднань з одним хостом. Щоб уникнути «пляшкової шийки», використовується агрегація ресурсів, щоб зменшити кількість окремих елементів сторінки та згрупувати невеликі файли (наприклад, зображення) в один файл. Це зменшує кількість HTTP-запитів і "туди й назад", необхідних для завантаження веб-сторінки.

Інформаційна система складається з файлів коду, таких як JavaScript, HTML та інших. Оскільки інформаційні системи стають складнішими, файли коду стають складнішими також й завантажуються довше. Стиснення файлів може зменшити файли коду до 80% і зробити швидкодію між користувачем та сервером інформаційної системи більш чуйним.

Теорія баз даних охоплює широкий спектр тем, пов’язаних із вивченням і дослідженням баз даних і теорії систем управління базами даних [5].

З точки зору теорії управління даними, основи мови запитів, обчислювальної складності, виразу запиту, теорії скінченних моделей, теорії проектування баз даних, теорії відношень, керування паралелізмом і основ відновлення бази даних, часових поясів і просторових регіонів.

Більшість досліджень традиційно спираються на реляційні моделі. Тому що ця модель часто вважається найпростішою та базовою моделлю, що викликає

інтерес. Відповідні результати для об’єктно орієнтованих або напівструктурованих моделей часто йдуть як результат з реляційних моделей [6].

Основним напрямком теорії баз даних є розуміння складності, потужності та логіки мов запитів. Починаючи з інтуїції, що такі важливі запити, як реляційна алгебра, логіка першого порядку і логічне програмування, як реєстрація даних. Іншим фокусом були основи оптимізації запитів та інтеграції даних. Велика частина роботи виконується над запитами на об’єднання, які дозволяють оптимізувати пошук за допомогою різних алгоритмів, незважаючи на їх обмеження без можливості доповнення [7].

До баз даних зазвичай відноситься набір пов’язаних даних і способів їх організації. Доступ до цих даних зазвичай надається (обмежується) системою управління базами даних, доступ до деяких даних може бути обмежений. Система управління базами даних пропонує різноманітні функції, які дозволяють вводити, зберігати та отримувати великі обсяги даних, а також надає спосіб керування організацією цих даних [8].

Оскільки вони тісно пов’язані, термін «база даних» часто стосується як самої бази даних, так і система управління базами даних, яка використовується для її запуску [9].

За межами професійного світу інформаційних технологій, термін «база даних» часто використовується для позначення набору пов’язаних даних, таких як таблиці та картотеки. Тому що вимоги до розміру та експлуатації часто вимагають використання системи керування базами даних [10].

Фізичний сервер бази даних — це спеціальний комп’ютер, на якому працює лише система управління базами даних і відповідне програмне забезпечення, та зберігається фактична база даних. Сервери баз даних, як правило, є багатопроцесорними комп’ютерами з дисковими масивами для достатньої пам’яті та стабільного зберігання. Апаратні прискорювачі баз даних, підключені до одного або кількох серверів через високошвидкісні канали, також використовуються в середовищах обробки великих обсягів транзакцій. Система управління базами даних є ядром більшості програм баз даних [11].

Оскільки система управління базами даних займає важливе місце на ринку, постачальники обчислювальної техніки та зберігання даних часто враховують вимоги до системи управління базами даних у своїх планах розвитку [12].

База даних XML – це тип структурованої орієнтованою на документи бази даних, до якої можна надсилати запити на основі атрибутів документа XML. Бази даних XML в основному використовуються програмами, які хочуть представити дані як набір документів. Його структура може варіюватися від дуже гнучкої до дуже твердої.

Бази даних NoSQL зазвичай дуже швидкі, не вимагають фіксованих схем таблиць, уникають об’єднань, зберігаючи нетипові дані, і розроблені для горизонтального масштабування [13].

В останні роки існує великий попит на великомасштабні розподілені бази даних з високою толерантністю до розділів, але згідно з теоремою CAP, розподілені системи не можуть гарантувати узгодженість, доступність і толерантність до розділів одночасно. Розподільна система може задовольнити дві з цих напрямів одночасно, але не всі три. У результаті, багато баз даних NoSQL використовують так звану наскрізну узгодженість, щоб забезпечити як доступність, так і толерантність до розділів, знижуючи рівень узгодженості даних [14].

Бази даних використовуються для зберігання адміністративної інформації та більш спеціалізованих даних, таких як інженерні дані та економічні моделі. Наприклад, комп’ютеризовані бібліотечні системи, системи бронювання квитків, комп’ютеризовані системи інвентаризації запчастин і багато систем керування вмістом, які зберігають інформаційні системи у базах даних [15].

Отже, провівши глибокий аналіз програмного забезпечення, а також враховуючи огляд останніх досліджень стало очевидним актуальність обраної теми, а саме створення інформаційної системи для онлайн-консультацій. Інформаційна система включає в себе простоту, універсальність та великий набір функціональних можливостей.

1.2 Аналіз програмних продуктів

Перш ніж починати роботу над інформаційною системою, обов’язково проводиться робота з іншими інформаційними системами, які надають такі ж або подібні послуги.

MEDIKIT – інформаційна система для платних онлайн-консультацій з лікарями різних напрямків. Сервіс дає можливість отримати консультаційну допомогу від лікарів в режимі онлайн. Достатньо описати скарги, як через деякий час, користувача починає консультувати лікар того напрямку, який відповідає запиту [16].

Консультація може проводитися через текстовий чат, відео або аудіо зв’язок. Консультація закінчується порадами та рекомендаціями для подальшого лікування. Інформаційна система хоч і має разючий функціонал та має безліч недоліків в плані реалізації дизайнерської частини, а саме: текст інформаційного призначення погано видно, важко орієнтуватися у функціоналі. Недоліки добре видно на рисунку 1.1.

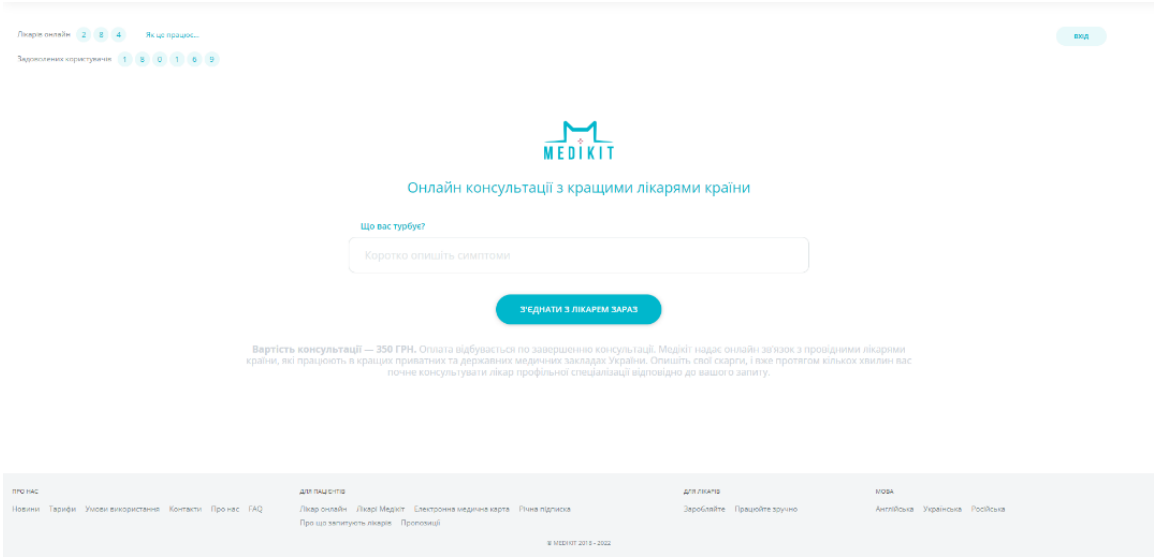


Рисунок 1.1 – Головна сторінка Medikit

Apollo 24|7 – інформаційна система яка містить онлайн-аптеки, онлайн консультації та діагностичні лабораторні тести вдома [17].

Хоч ця інформаційна система і має неймовірний набір функцій, але головна сторінка перенавантажена інформацією, яка подається лише англійською мовою та не має можливості перевести на іншу мову. Приклад демонстрації на рисунку 1.2.

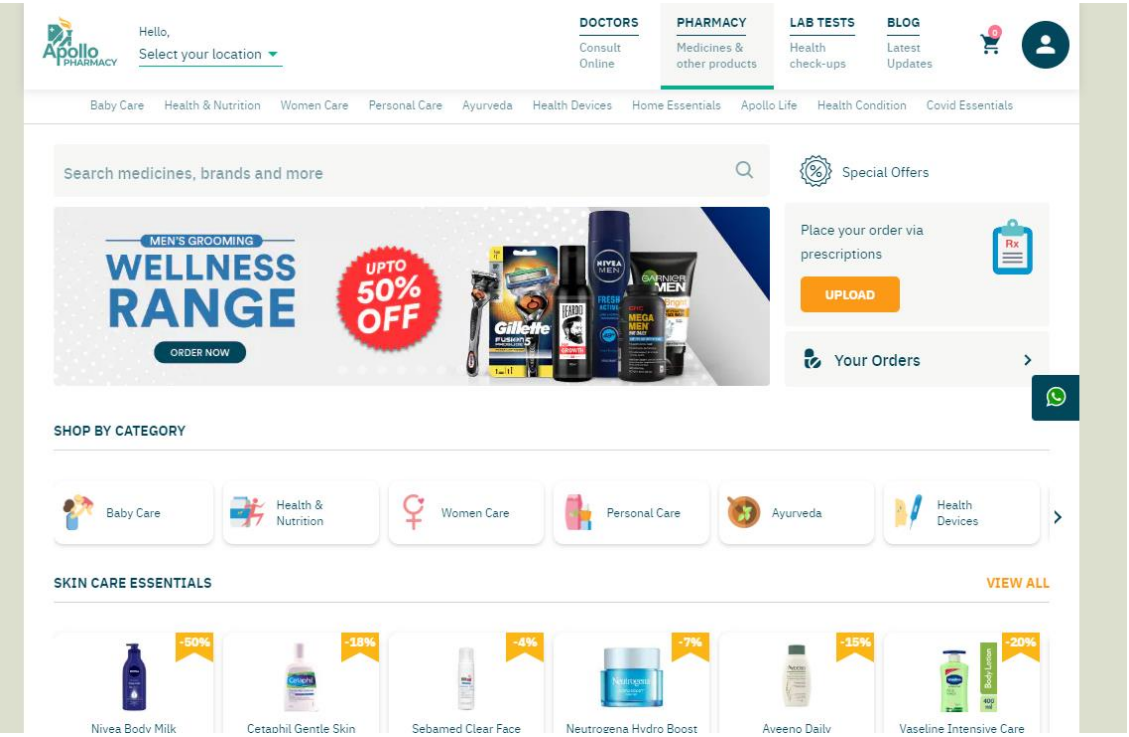


Рисунок 1.2 – Онлайн-аптека Apollo 24|7

HARLEY THERAPY – інформаційна система для психотерапії та консультування в Лондоні. Основним напрямком консультацій є консультація психотерапевта для допомоги в прояснюванні проблем психологічного стану [18].

Головною проблемою інформаційної системи є обмежена клієнтоорієнтованість, що не дає використовувати систему для вирішення інших проблем в цій галузі. Також, наявна проблема довгого завантаження сторінок сервісу. Головна сторінка сервізу зображена на рисунку 1.3.

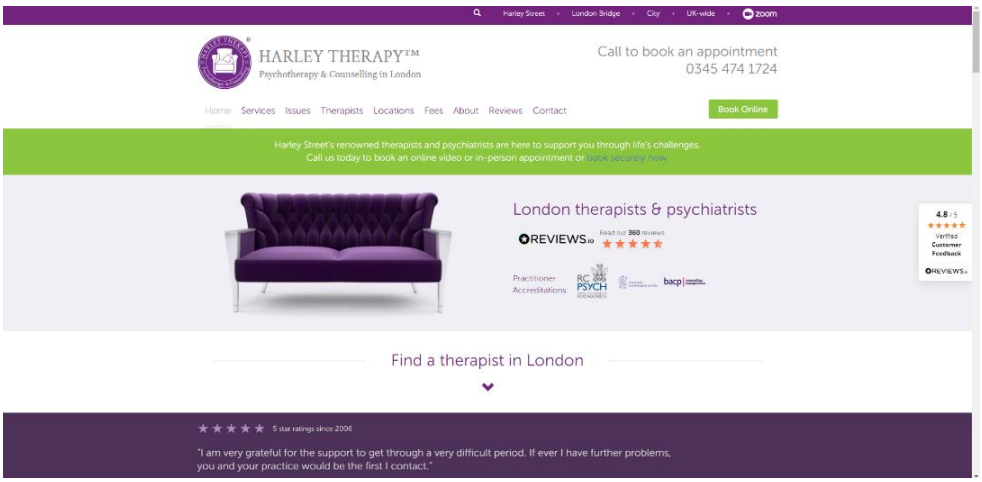


Рисунок 1.3 – Головна сторінка Harley therapy

Нижче проведено аналіз конкурентних інформаційних систем, що надають подібні послуги наведені в таблиці. 1.1 та таблиці 1.2.

Таблиця 1.1 – Порівняльна таблиця сайтів для консультацій

№	Назва	Переваги	Недоліки
1	Medikit	Оплата послуг сервісу картами, електронними грошима	
		Має мінімалістичний та приємний дизайн	Інформаційний текст погано видно
		Дає можливість консультуватися з реальними лікарями	Функціонал пов’язаний лише з медичним фахом
2	Apollo 24|7	Великий набір сервісних можливостей з відповідним функціоналом	Сторінки перевантажені інформацією
		Наявність онлайн-аптеки, що спрощує замовлення потрібних ліків	Наявна лише англійська мова без можливості переводу всієї сторінки

№	Назва	Переваги	Недоліки
3	Harley therapy	Має можливість отримати професійну допомогу	Консультація проходить лише англійською мовою
		Має мінімалістичний та приємний дизайн	
		Дає можливість консультуватися з реальними лікарями	Функціонал пов’язаний лише з терапевтами та психіатрами

Таблиця 1.2 – Порівняльна характеристика аналогів

Критерії	«Medikit»	«Apollo 24|7»	«Harley therapy»	Інформаційна система для онлайн-консультацій
Можливість запланувати консультацію	–	–	–	+
Комфортний дизайн	–	–	+	+
Охоплює різні напрямки	–	–	–	+
Наявність чату	+	+	–	+
Можливість проконсультуватися за відео зв’язком	+	–	+	+
Можливість створити особисту сторінку	+	+	–	+

Критерії	«Medikit»	«Apollo 24|7»	«Harley therapy»	Інформаційна система для онлайн-консультацій
Консультації на різних мовах	–	–	–	+

Проаналізувавши набір із конкурентних інформаційних систем, можна зробити відповідні висновки. Хоч дані інформаційні системи і мають унікальний функціонал, та все ж тісно пов’язані вони лише з медициною. Це робить із системи вузькоспеціалізований інструмент направлений на рішення проблемі лише в окремій галузі. Оцінивши комунікативні здібності цих інформаційних систем, можна підкреслити реалізацію подібних корисних особливостей [19].

Отже, інформаційна система повинна мати універсальність в галузевій орієнтованості, підтримувати гарну комунікацію між фахівцем та користувачем, а також комфортне управління записами на подібні активності.

2 ПОСТАНОВКА ЗАДАЧІ ТА МЕТОДИ ДОСЛІДЖЕННЯ

2.1. Мета та задачі дослідження

Створення інформаційної системи для онлайн-консультацій, яка направлена на задоволення потреб фахівців та користувачів. Ця система повинна упорядкувати віртуальних графік консультацій різних напрямків та полегшити орієнтацію між ними. Детальніша інформація представлена у додатку А.

Для виконання поставленої мети реалізовані такі функціональні вимоги:

1. Реєстрація та авторизація користувачів;
2. Можливість зареєструватися як звичайний користувач, та як фахівець;
3. Встановлення фахівцем час роботи на сторінці «Календар» ;
4. Формування сповіщень на електронну пошту;
5. Можливість редагувати особисту інформацію;
6. Можливість написати зареєстрованому користувачу;
7. Можливість забронювати час для консультацій з можливістю прикріплення файлів;
8. Можливість редагування часу зустрічі;
9. Додавання нових задач до календаря;
10. Оцінювання користувача після зустрічі;
11. Створення Zoom-посилання;
12. Можливість подати скаргу на користувача.

Для виконання поставленої мети реалізовані такі нефункціональні вимоги:

1. Фільтрація користувачів за напрямками роботи, ціною послуг та ім’ям;
2. Відображення оцінки від інших користувачів;
3. Перегляд інформації запланованих зустрічей в календарі.

Використання інформаційної системи має охоплювати великий спектр напрямків, охоплюючи потреби максимальної кількості користувачів. Користувачі та фахівці повинні мати можливість реєстрації облікового запису. При

використанні інформаційної системи, дизайн повинен мати простий вигляд, а функціонал має бути розміщений на відповідних посиланнях. Система має забезпечити користувачів та фахівців актуальними сповіщеннями на e-mail [20].

Потрібно, щоб користувачеві та фахівцю відповідали відповідні унікальні функції. Також дати можливість залишити користувачеві залишити якісний фідбек у вигляді оцінки. Наділити особливими можливостями функцію «календар» для комфортного управління та моніторингу консультаціями.

2.2. Розробка візуальної частини інформаційної системи

При розробці інформаційної системи для онлайн-консультацій з’явилася можливість розділити роботу на етапи. Первинним етапом стало створення візуалізованої частини сайту, тобто верстка front-end за допомогою HTML5 та CSS3. Для цього етапу використано базові інструменти верстки сайтів [21].

JavaScript використовують, щоб створити додаткову анімацію, а також для реалізації взаємодії між користувачами на веб-сайті. Ця мова програмування має дуже розгалужену структуру для розширення функціонала та прискорення написання коду. Прикладами є React.js, Vue.js та Angular.js. Аналіз фреймворків проведено в таблиці 2.1.

Тому, ретельно проаналізувавши фреймворк, можна зробити висновок, що React.js є найкращим вибором для розробки інформаційної системи онлайн-консультацій.

Цей фреймворк ідеально підходить для створення особливих функціональних додатків. Він дозволяє швидко приступити до роботи з бібліотеками та вставляти React.js в уже чинні сторінки [22].

Таблиця 2.1 – Порівняльна таблиця фреймворків мови JavaScript

№	Назва	Переваги	Недоліки
1	React.js	Стабільна та передова реалізація технологій.	React спроможне реалізувати частину View моделі MVC.
		Спрощена продуктивність.	React - передова технологія, при швидкому розвитку може викликати труднощі у розробників.
№	Назва	Переваги	Недоліки
		Комфортне налаштування та тестування.	
2	Vue.js	Постійно зростаюча аудиторія.	Має свої особливості при написанні коду.
		Підтримує майже всі сфери IDE.	Відсутність потрібної кількості бібліотек.
		Документація має структуровану схему навігації.	Складності с доступом до потрібної документації.
3	Angular.js	Подвійна прив'язка даних.	Великий ступінь входу для роботи з MVC.
		Автоматизована та продуктивна робота сервера.	Збільшення часу відповіді браузера для візуалізації веб-сайтів та програм.
		Зручність моделювання додатків.	Складності с доступом до потрібної документації.

2.3. Розробка функціональної частини інформаційної системи

Розробники back-end працюють із різними бібліотеками, API, веб-службами тощо. Жоден фреймворк не є ідеальним, тому що кожен фреймворк має свої сильні та слабкі сторони, і розробники можуть робити з сучасним фреймворком все, що завгодно. Однак фреймворк є досить низькорівневим рішенням, яке забезпечує архітектуру та деякі допоміжні бібліотеки [23].

Розглянуто основні прийоми розробки функціональних частин інформаційної системи онлайн-консультацій в таблиці 2.2.

Таблиця 2.2 – Порівняльна таблиця фреймворків для back-end розробки

№	Назва	Переваги	Недоліки
1	Laravel	Оптимізована маршрутизація.	Проблеми з точністю.
		Має власний CLI.	Наявні компанії, які не хочуть витрачати час та ресурси для побудови з допомогою Laravel.
		Велика бібліотека шаблонів.	
		Структурована документація.	
2	Node.js	Підтримка сучасних технологій.	Погана реалізація оснащення.
		Велика швидкість обробки.	Відсутність спеціалізованих фахівців.
		Підтримка аудиторії.	Непомірний поріг входу.
3	Spring Boot	Супутнє програмне забезпечення.	Краще підходить для систем невеликого масштабу.
		Розвинутий в масштабі розробки.	Непомірний поріг входу.

		Може використовувати хмарні технології.	Сильно відрізняється від інших концептів.
		Структурована документація.	

Провівши інформаційний аналіз, взято для розробки Laravel. Фреймворк має безліч переваг над конкурентами у сфері PHP, бо має функції, які будуть використані при створенні веб-систем [24].

База даних буде реалізована за допомогою MySQL, бо має переваги серед конкурентів в простоті та надійності. Гнучкість MySQL походить від підтримки багатьох типів таблиць. Користувачі можуть вибирати між таблицями MyISAM, які підтримують повнотекстовий пошук, і таблицями InnoDB, які підтримують транзакції з одним записом. Крім того, MySQL поставляється зі спеціальним типом таблиці EXAMPLE, який демонструє, як створити новий тип таблиці. Завдяки відкритій архітектурі та ліцензії GPL до бази даних MySQL постійно додаються нові типи таблиць [25].

3 МОДЕЛЮВАННЯ ТА ПРОЕКТУВАННЯ

3.1 Структура інформаційної системи

Найважливішим етапом проектування інформаційної системи є розробка структури прикладного програмного забезпечення. Проектування інформаційної системи можна описати як багатоетапний процес створення та модернізації з використанням методичних та різноманітних інструментів [26].

Виділення етапу проектування інформаційної системи як окремого етапу можна розмістити між аналізом та розробкою. На практиці, однак, планування, яке формально починається з визначення цілей проекту, часто продовжується через фази тестування та впровадження, що робить чіткий розподіл на фази складним або в принципі неможливим [27].

Основною відмінністю є інформація про сайт і доступ до певних сторінок. Перед етапом авторизації система дає доступ до деяких модулів (табл. 3.1).

Таблиця 3.1 – Сторінки до авторизації

№	Сторінка	Опис
1	Головна	Сторінка яка надає первинну інформацію про інформаційну систему.
2	Авторизація	Сторінка, щоб авторизуватися зареєстрованому користувачеві.
3	Реєстрація	Сторінка, щоб зареєструватися новому користувачеві.

Після етапу авторизації користувач отримує додаткові можливості. Вже на цьому кроці з’являється різниця – фахівець чи звичайний користувач.

Для більш детальної демонстрації різниці можливостей обох видів облікових записів, створюється порівняльна таблиця доступу до інформаційної системи онлайн-консультацій (табл. 3.2).

Таблиця 3.2 – Таблиця рівнів доступу

Сторінки	Фахівець	Користувач
Сторінка з новинами	+	+
Сторінка фахівців	+	+
Сторінка повідомлень	+	+
Сторінка планування онлайн-консультацій	+	+
Домашня сторінка користувача	+	+
Редагування даних	+	+
Особиста сторінка з відгуками	+	-
Налаштування безпеки	+	+
Перегляд онлайн-консультацій в календарному вигляді	+	+
Створення онлайн-консультацій	-	+
Перегляд набору онлайн-консультацій фахівця	+	+
Оцінка онлайн-консультацій	-	+
Отримання заявок на онлайн-консультацію	+	-
Вихід з системи	+	+

Інформаційна система для онлайн-консультацій структурована, та має логічну взаємодію між користувачами. Демонстрацією є спроектована інформаційна система. Користувач, який пройшов реєстрацію, отримує можливість переглядати інформацію про онлайн-консультації, переглядати інші сторінки користувачів і т.д.

Особливу увагу було приділено цілям дипломної роботи.

3.2 Структурно-функціональне моделювання процесу

Для інформаційної системи поставлена конкретна задача – створити інформаційну систему для онлайн-консультацій. Для побудови буде використано вхідні та вихідні дані, механізми та управління.

Вхідними даними є: інформація про онлайн-консультацію, інформація про користувача, інформація про фахівця. Вихідними даними є: закінчена онлайн-консультація, підвищення рейтингу. Механізмами є: розробник, програмне забезпечення, web-сервіс, апаратне забезпечення, API Zoom.

Контекстна діаграма представлена на рисунку 3.1.

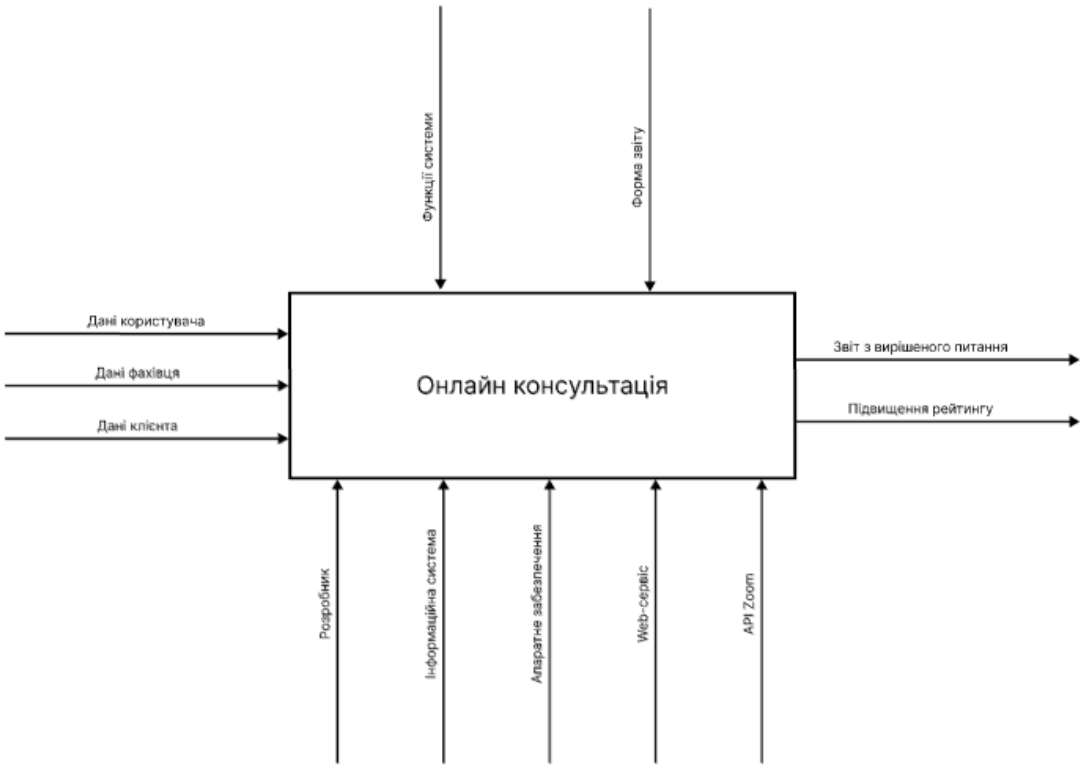


Рисунок 3.1 – Контекстна діаграма інформаційної системи для онлайн консультацій

Модель інформаційної системи для онлайн-консультацій продемонстровано у вигляді діаграми декомпозиції на рисунку 3.2.

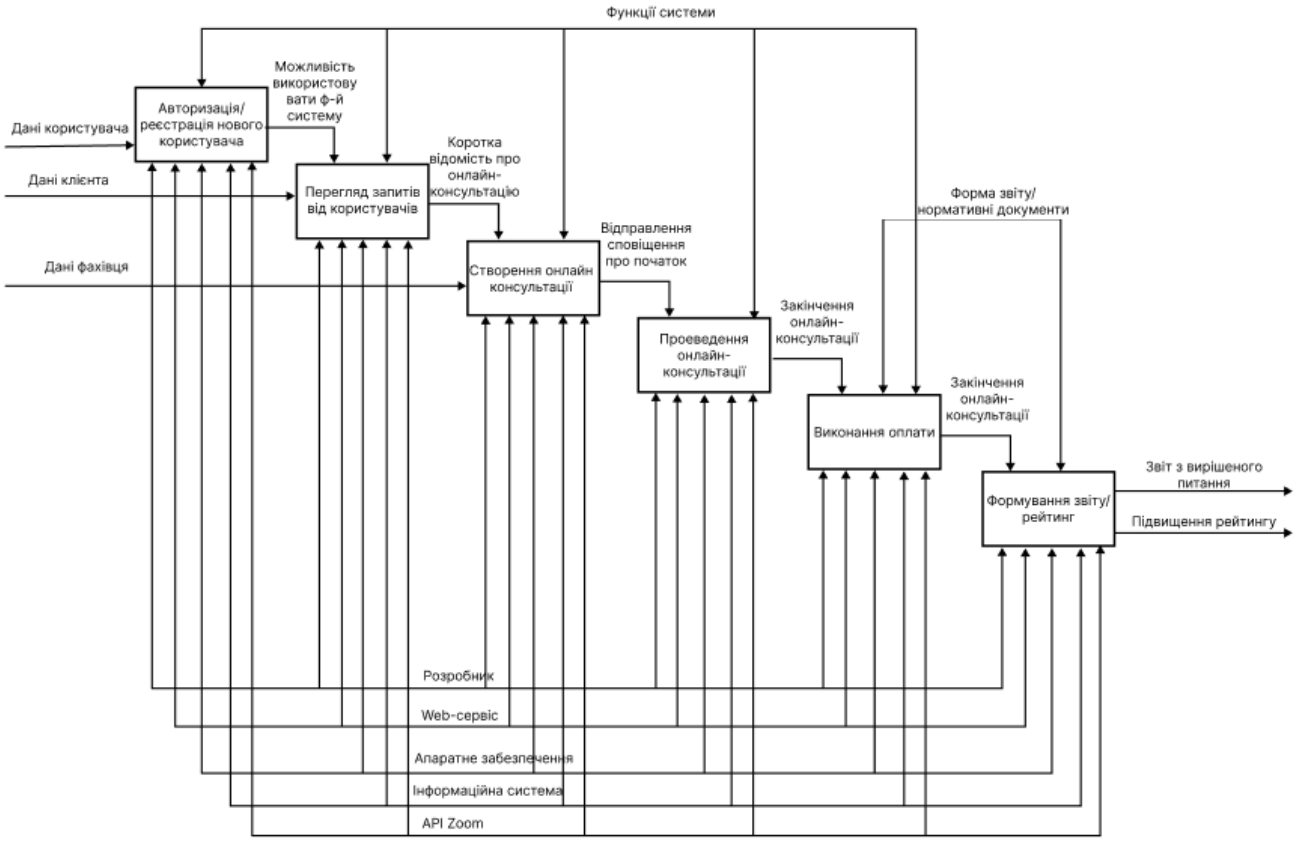


Рисунок 3.2 – Декомпозиція контекстної діаграми інформаційної системи для онлайн консультацій

3.3 Моделювання діаграми варіантів використання

Потрібно звернути увагу на розподіл функціонала за напрямком використання функціонала акторами інформаційної системи в таблиці.3.1 та таблиці 3.2.

Таблиця 3.1. – Опис акторів

№	Іменування актора	Детальний опис
1	Фахівець	Цей вид користувача має можливість створювати онлайн-консультації, формувати календарне навантаження, редагувати їх, ухвалювати або

№	Іменування актора	Детальний опис
		скасовувати заявки та вести листування з користувачем.
2	Слухач	Цей тип користувача може відправляти заявки на онлайн-консультації, формувати графік відвідування, формувати оціночний відгук на консультацію та вести переписку з фахівцем, скаржитися на фахівця або користувача.
3	Адміністратор	Цей вид користувача має найширшу базу можливостей, а саме: можливість створювати та редагувати новини, редагувати та видаляти данні користувача або фахівця, вести діалог з користувачами або фахівцями.

Таблиця 3.2. – Текстове тлумачення варіантів використання

№	Варіант використання	Детальний опис	Іменування актора
1	Авторизація	Модуль, який дає можливість зареєстрованому користувачеві авторизуватися на власну сторінку.	Всі користувачі
2	Реєстрація	Модуль, який дає можливість створити сторінку новому користувачеві.	Всі користувачі
3	Редагування інформації на сайті	Модуль, який дає можливість вносити зміни, або видаляти інформацію в межах інформаційної системи.	Адміністратор
4	Перегляд онлайн-консультацій	Модуль, який дає можливість переглядати доступний набір онлайн-консультацій. Перегляд обмежується лише авторизованими користувачами.	Всі користувачі

№	Варіант використання	Детальний опис	Іменування актора
5	Перегляд інформації про зареєстрованих фахівців	Модуль, який дає можливість переглядати авторизованим користувачам інформацію про зареєстрованих фахівців.	Всі користувачі
6	Створення нової онлайн-консультації	Модуль, який дає можливість створювати нову онлайн-консультацію за бажаним напрямком.	Фахівець
7	Редагування онлайн-консультації	Модуль, який дає можливість редагувати інформацію та учасників онлайн-консультації.	Фахівець
8	Зміна статусу користувача в онлайн-консультації	Модуль, який дає можливість змінювати статус користувача перед початком онлайн-консультації.	Фахівець
9	Запит на онлайн-консультацію	Модуль, який дає можливість відправляти запит на відвідування онлайн-консультацію.	Користувач
10	Формування Zoom-посилання	Модуль, який дає можливість створювати посилання для онлайн-консультації за допомогою сервісу Zoom.	Фахівець
11	Чат між фахівцем та користувачем	Модуль, який дає можливість вести діалог між фахівцем та користувачем в текстовому вигляді.	Фахівець, користувач
12	Пошук та фільтрації онлайн-консультацій	Модуль, який дає можливість вести пошук або проводити пошук за фільтрами для пришвидшення	Всі користувачі

№	Варіант використання	Детальний опис	Іменування актора
		звичайного пошуку онлайн консультації.	
13	Блокування користувачів	Модуль, який дає можливість блокувати користувачів або фахівців.	Адміністратор
14	Редагування особистих даних	Модуль, який дає можливість редагувати авторизованому користувачеві особисту інформацію в межах інформаційної системи.	Всі користувачі
15	Оцінка онлайн-консультації	Модуль, який дає можливість оцінювати онлайн-консультацію за спеціальною системою оцінювання.	Користувач
16	Публікація новин	Модуль, який дає можливість створювати стіну новин через панель адміністратора.	Адміністратор

Сформувавши уявлення про функціональні можливості інформаційної системи для онлайн консультацій, була сформована схема варіантів використання на рисунку 3.3.

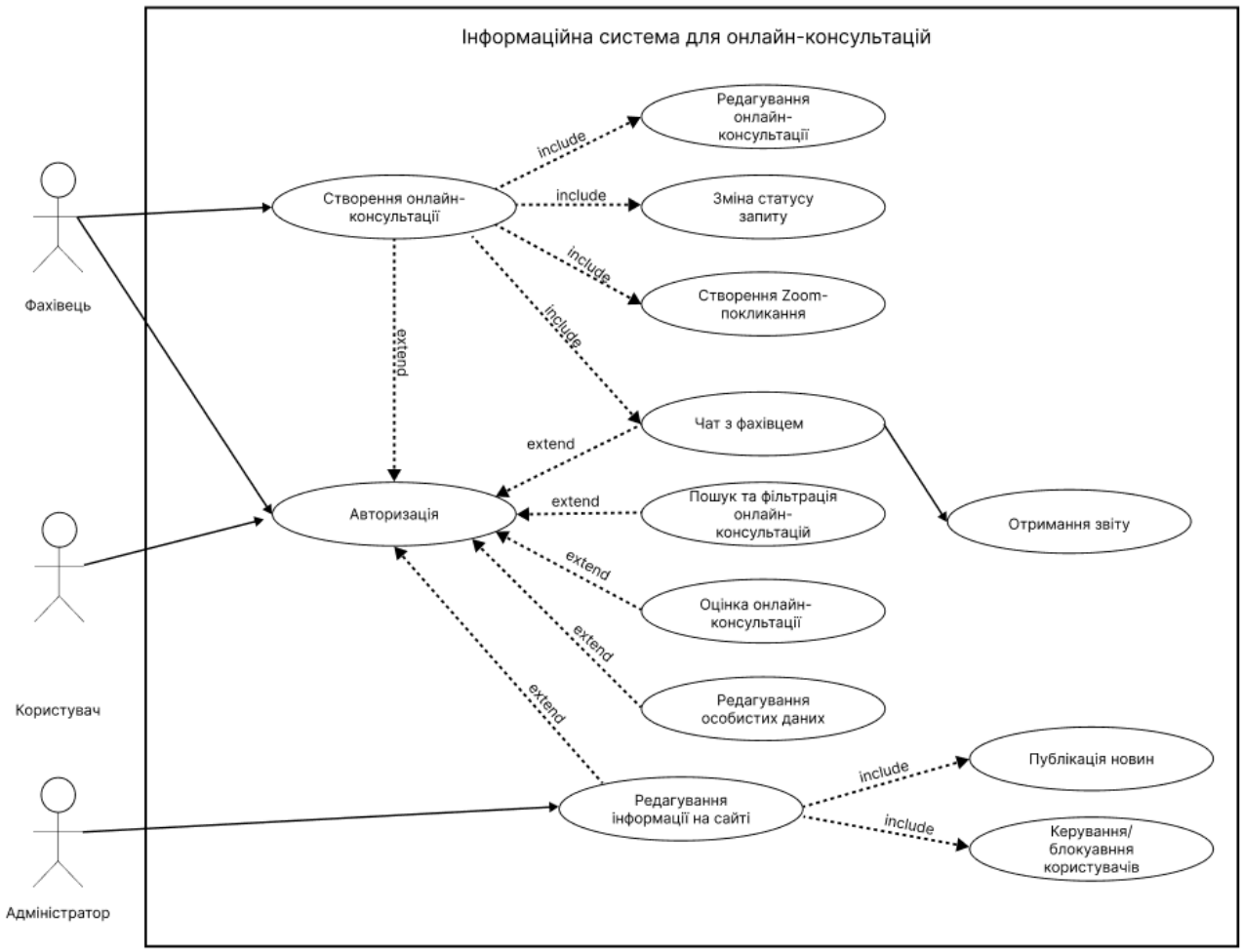


Рисунок 3.3 – Діаграма варіантів використання

3.4. Проектування бази даних

Основні користувачі системи – фахівець та користувач. За допомогою цієї системи вони можуть отримувати різні відомості про онлайн-консультації, інших фахівців, напрямки та інше.

Враховуючи цільову аудиторію користувачів, можна сформувати основні вимоги до бази даних. Користуючись цією системою, користувач може знайти людину для консультації за різними напрямками [28].

Візуалізація взаємодії показана на ER-діаграмі (рис.3.5). Провівши структурний аналіз використання інформаційної системи в моделі, можна

переходити до моделювання бази даних. Створені найменування таблиць, полів, їх зміст тип та обмеження в табл. 3.2.

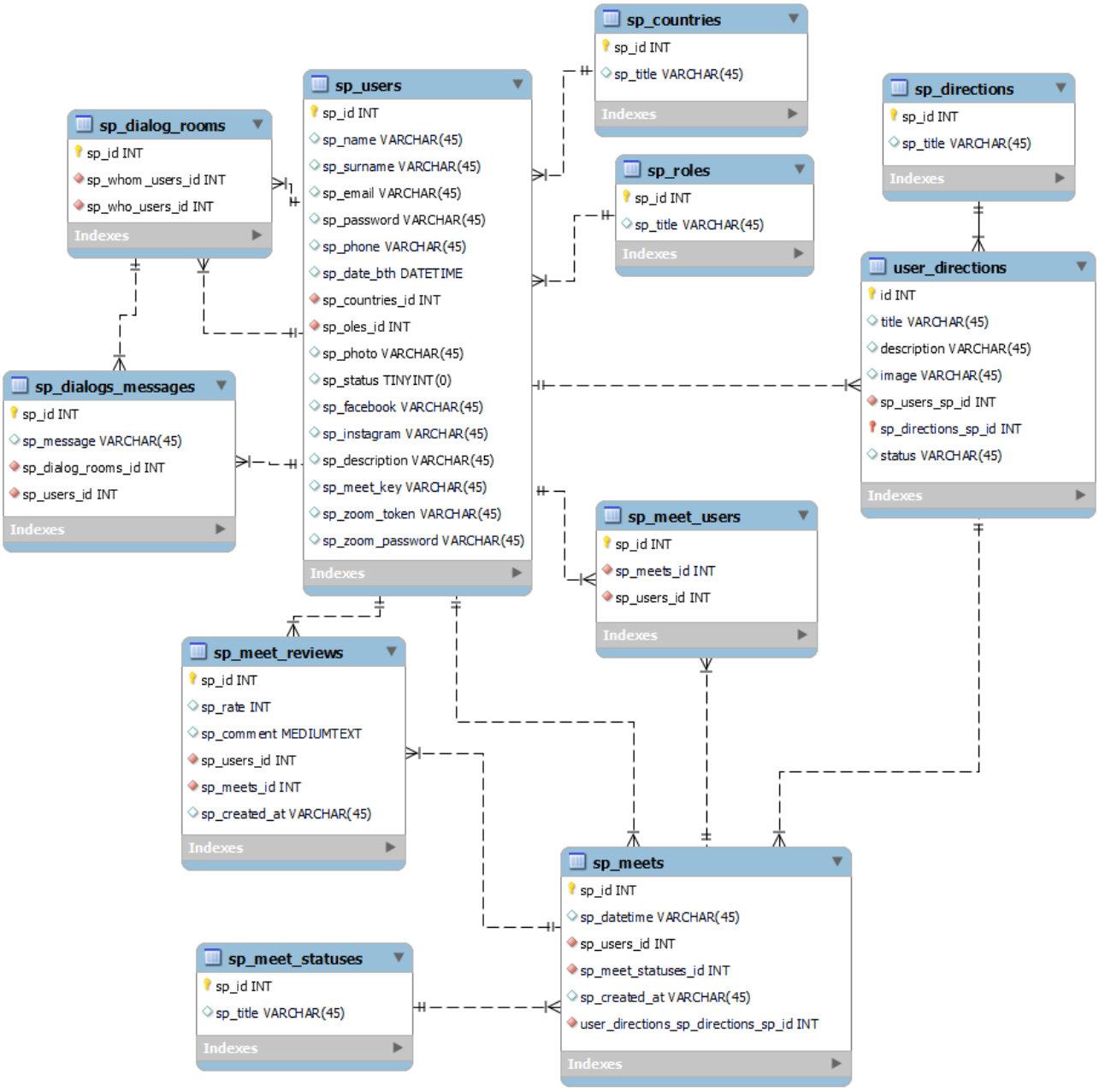


Рисунок 3.5 – ER-діаграма зі зв’язками

На ER-діаграмі (рис. 3.6) представлені таблиці без зв’язків, оскільки використовуються для заповнення інформації на сайті або інформації про адміністраторів системи та новин. В цій таблиці немає зв’язків, бо вони не взаємодіють з іншими блоками інформації.

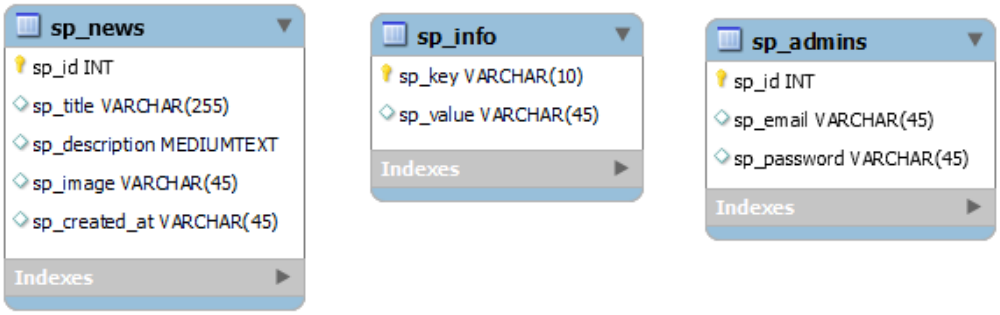


Рисунок 3.6 – ER-діаграма без зв’язків

Таблиця 3.3 – Інформація за таблицями ER-діаграми інформаційної системи онлайн-консультацій

№	Таблиця	Поле	Зміст	Тип	Ключі	Обмеження
1	sp_users	ip_id	Ідентифікатор аккаунта	INTEGER	PK	Не порожній
		sp_name	Ім’я користувача	VARCHAR(45)		Не порожній
		sp_surname	Прізвище користувача	VARCHAR(45)		Не порожній
		sp_email	Електронна пошта	VARCHAR(45)		Не порожній
		sp_password	Пароль користувача	VARCHAR(45)		Не порожній
		sp_phone	Номер телефону	VARCHAR(45)		Не порожній
		sp_date_bth	Дата народженя	DATETIME		Не порожній
		sp_countries_id	Ідентифікатор країни	INTEGER	FK	Не порожній
		sp_roles_id	Ідентифікатор ролі	INTEGER	FK	Не порожній
		sp_photo	Фото або логотип	VARCHAR(45)		Не порожній
		sp_status	Статус блокування аккаунту	TINYINT		Не порожній
		sp_facebook	Посилання на фейсбук	VARCHAR(45)		Не порожній
		sp_instagram	Посилання на інстаграм	VARCHAR(45)		Не порожній
		sp_description	Короткий опис користувача	VARCHAR(45)		Не порожній
		sp_zoom_token	Ключ API в Zoom	VARCHAR(45)		Не порожній
		sp_zoom_password	Пароль API в Zoom	VARCHAR(45)		Не порожній
2	sp_countries	sp_id	Ідентифікатор країни	INTEGER	PK	Не порожній
		sp_title	Назва країни	VARCHAR(45)		Не порожній
3	sp_roles	sp_id	Ідентифікатор ролі	INTEGER	PK	Не порожній

№	Таблиця	Поле	Зміст	Тип	Ключі	Обмеження
		sp_title	Назва ролі	VARCHAR(45)		Не порожній
4	sp_users_has_directions	sp_users_id	Ідентифікатор користувача	INTEGER	FK	Не порожній
		sp_directions_id	Ідентифікатор напрямку	INTEGER	FK	Не порожній
5	sp_directions	sp_id	Ідентифікатор напрямку	INTEGER	PK	Не порожній
		sp_title	Назва напрямку	VARCHAR(45)		Не порожній
6	sp_meets	sp_id	Ідентифікатор зустрічі	INTEGER	PK	Не порожній
		sp_title	Назва зустрічі	VARCHAR(45)		Не порожній
		sp_datetime	Час початку зустрічі	VARCHAR(45)		Не порожній
		sp_users_id	Ідентифікатор створювача зустрічі	INTEGER	FK	Не порожній
		sp_meet_statuses_id	Ідентифікатор статусу зустрічі	INTEGER	FK	Не порожній
		sp_created_at	Дата створення зустрічі	DATETIME		Не порожній
		sp_description	Описання зустрічі	TEXT		Не порожній
		sp_image	Картинка зустрічі	VARCHAR(45)		Не порожній
		sp_directions_id	Ідентифікатор напрямку	INTEGER	FK	Не порожній
7	sp_meet_users	sp_id	Ідентифікатор	INTEGER	PK	Не порожній
		sp_meets_ id	Ідентифікатор зустрічі	INTEGER	FK	Не порожній
		sp_users_id	Ідентифікатор аккаунта	INTEGER	FK	Не порожній

№	Таблиця	Поле	Зміст	Тип	Ключі	Обмеження
8	sp_meet_statuses	sp_id	Ідентифікатор статусу зустрічі	INTEGER	PK	Не порожній
		sp_title	Назва статусу зустрічі	VARCHAR(45)		Не порожній
9	sp_meet_reviews	sp_id	Ідентифікатор відгуку зустрічі	INTEGER	PK	Не порожній
		sp_rate	Оцінка зустрічі	INTEGER		Не порожній
		sp_comment	Коментар зустрічі	MEDIUMTEXT		Не порожній
		sp_users_id	Ідентифікатор аккаунта	INTEGER	FK	Не порожній
		sp_meets_id	Ідентифікатор зустрічі	INTEGER	FK	Не порожній
		sp_created_at	Дата створення відгуку	DATETIME		Не порожній
10	sp_dialogs_messages	sp_id	Ідентифікатор повідомлення	INTEGER	PK	Не порожній
		sp_message	Текст повідомлення	TEXT		Не порожній
		sp_dialog_rooms_id	Ідентифікатор кімнати чату	INTEGER	FK	Не порожній
		sp_users_id	Ідентифікатор аккаунта	INTEGER	FK	
11	sp_dialogs_rooms	sp_id	Ідентифікатор аккаунта	INTEGER	PK	Не порожній
		sp_whom_users_id	Ідентифікатор аккаунта замовника	INTEGER	FK	Не порожній
		sp_who_users_id	Ідентифікатор аккаунта виконавця	INTEGER	FK	Не порожній

№	Таблиця	Поле	Зміст	Тип	Ключі	Обмеження
13	sp_adminins	sp_id	Ідентифікатор аккаунта	INTEGER	PK	Не порожній
		sp_email	Електронна пошта	VARCHAR(45)		Не порожній
		sp_password	Пароль	VARCHAR(45)		Не порожній
14	sp_info	sp_key	Ідентифікатор запису	VARCHAR(10)	PK	Не порожній
		sp_value	Службова інформація	VARCHAR(45)		Не порожній
15	sp_news	sp_id	Ідентифікатор новини	INTEGER	PK	Не порожній
		sp_title	Заголовок новини	VARCHAR(255)		Не порожній
		sp_description	Текст новини	MEDIUMTEXT		Не порожній
		sp_image	Картинка новини	VARCHAR(45)		Не порожній
		sp_created_at	Дата створення	DATETIME		Не порожній

4 ПРАКТИЧНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

4.1 Програмна реалізація

Програмна реалізація інформаційної системи для онлайн консультацій була виконана за допомогою:

– React.js, який для створення front-end;

– фреймворк Laravel, який є частиною PHP, для розробки функціональної частини інформаційної системи.

Представлення структури на рисунку 4.1 та деталізований опис кожного пакету в таблиці 4.1

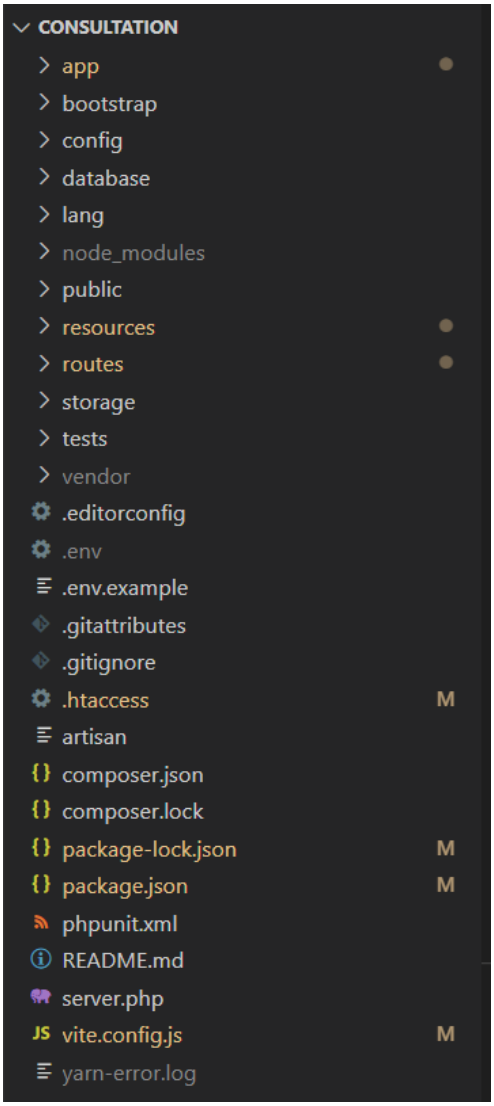


Рисунок 4.1 – Структура проекту

Таблиця 4.1 – Опис структури проекту

№	Пакет	Короткий опис
1	«app»	Містить код ядра Laravel-додатку.
2	«bootstrap»	Містить файли, ціль яких налаштувати автозавантаження фреймворку.
3	«config»	Містить конфігурації додатку.
4	«database»	Містить первинні данні для бази даних.
5	«lang»	Містить файли з варіантами переводу сайту.
6	«public»	Містить потрібні ресурси та є первинною точкою запитів.
7	«resources»	Містить первинний та ресурси LESS, SASS, JavaScript.
8	«routes»	Містить трафік маршрутів додатку.
9	«storage»	Містить створені шаблони, файли сесії, кешу і тимчасові файли фреймворку.
10	«tests»	Містить автоматизовані тести.

Після ретельного аналізу, отримана структура проекту інформаційної системи з шаблонами, файлами, ресурсами, даними та налаштуваннями.

4.2 Використання інформаційної системи зі сторони фахівця

Для початку використання інформаційної системи для онлайн-консультацій потрібно перейти на сторінку http://consultations.zzz.com.ua. На рисунку 4.2 зображена головна сторінка на яку заходить незареєстрований користувач з коротким описом головної ідеї інформаційної системи.

У верхній частині сторінки наявні переадресування на головний функціонал інформаційної системи, а також перехід на сторінку авторизації або реєстрації користувача.

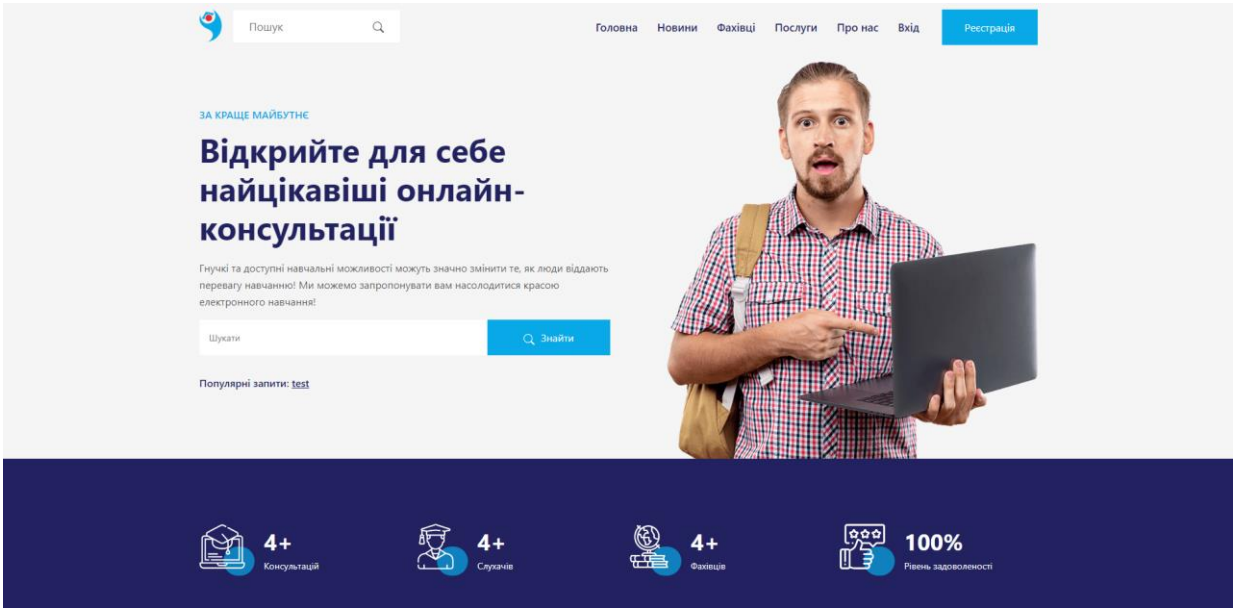


Рисунок 4.2 – Головна сторінка для відвідувачів

Для реєстрації користувачеві потрібно заповнити такі поля:

– Ім’я;
– Прізвище;
– Електронна пошта;
– Пароль;
– Мобільний телефон;
– Країну та місто проживання;
– Вибрати тип акаунту.

Поля обов’язкові для заповнення. Заповнивши всі поля, потрібно натиснути кнопку «Зареєструватися».

Наступним кроком після реєстрації є авторизація зображена на рисунку 4.4. Перехід на сторінку авторизації проходить автоматично та тут потрібно заповнити поля електронної пошти та паролю, які були вказані під час реєстрації.

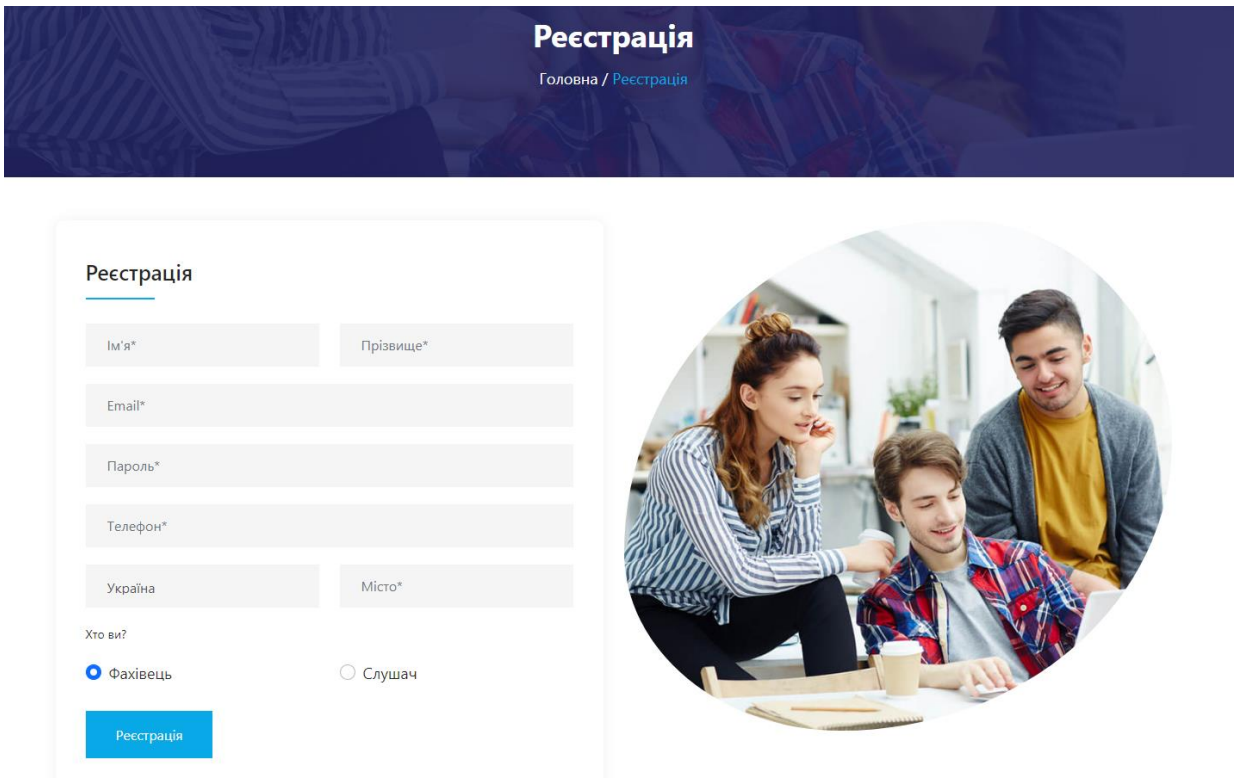


Рисунок 4.3 – Сторінка реєстрації

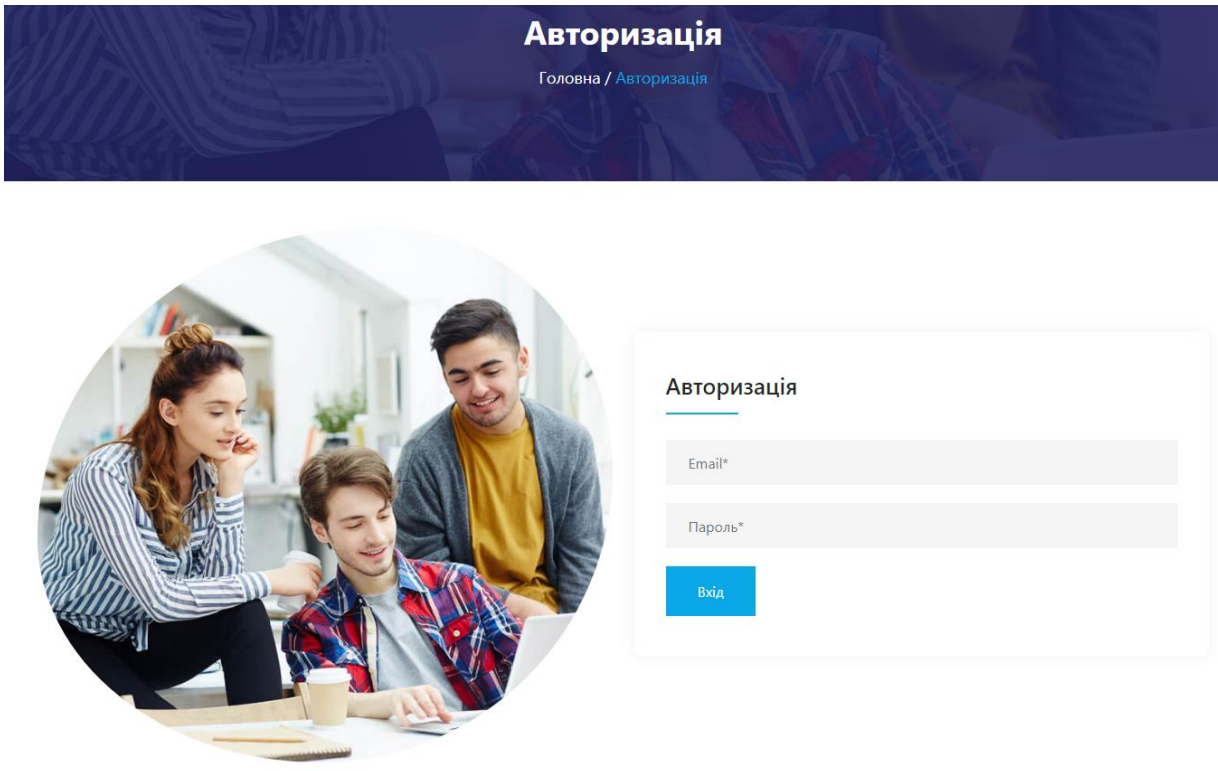


Рисунок 4.4 – Сторінка авторизації

Після авторизації користувач перенаправляється на головну сторінку свого кабінету з візуалізацією календаря, як зображено на рисунку 4.5. У налаштуваннях особистого кабінету, користувач має змогу змінити інформацію про себе, а саме:

– Фото користувача;
– Ім’я та прізвище;
– Електронну адресу;
– Номер телефону;
– Дату народження;
– Країну та місто;
– Посилання на соціальні мережі Facebook та Instagram;
– Змінити пароль з підтвердженням.

Для фахівців, є можливість додати такі дані:

– Час роботи;
– Коротка інформація про себе.

Особливу увагу було приділено комфортності навігації з кабінету користувача, щоб потрібні функції були завжди під рукою.

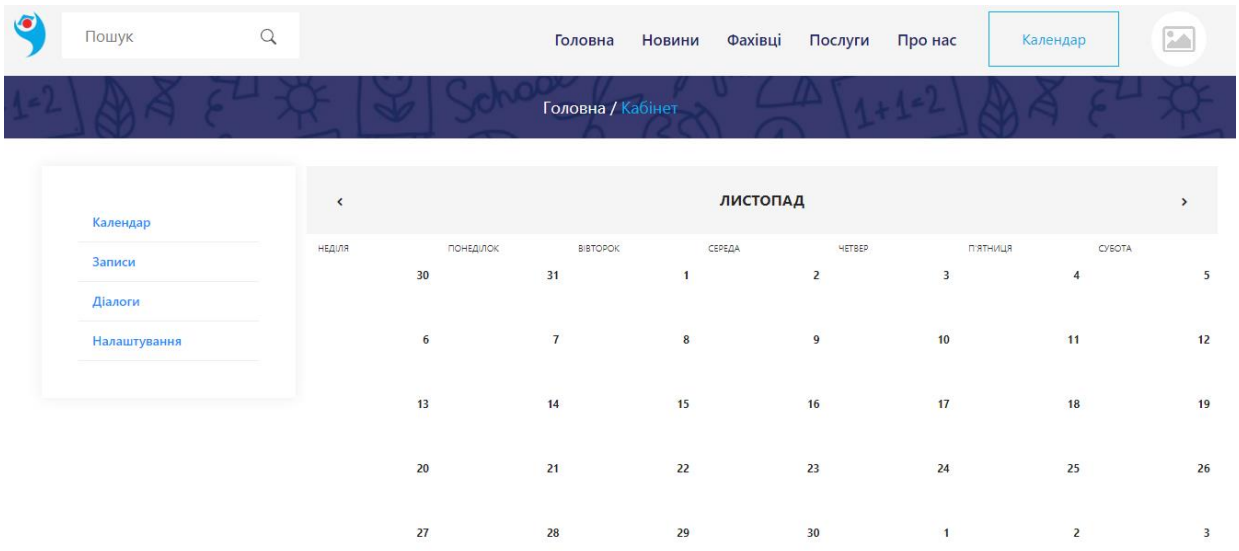


Рисунок 4.5 – Головна сторінка кабінету користувача

Сторінку з налаштуваннями особистої інформації користувача типу «фахівець» зображено на рисунку 4.6.

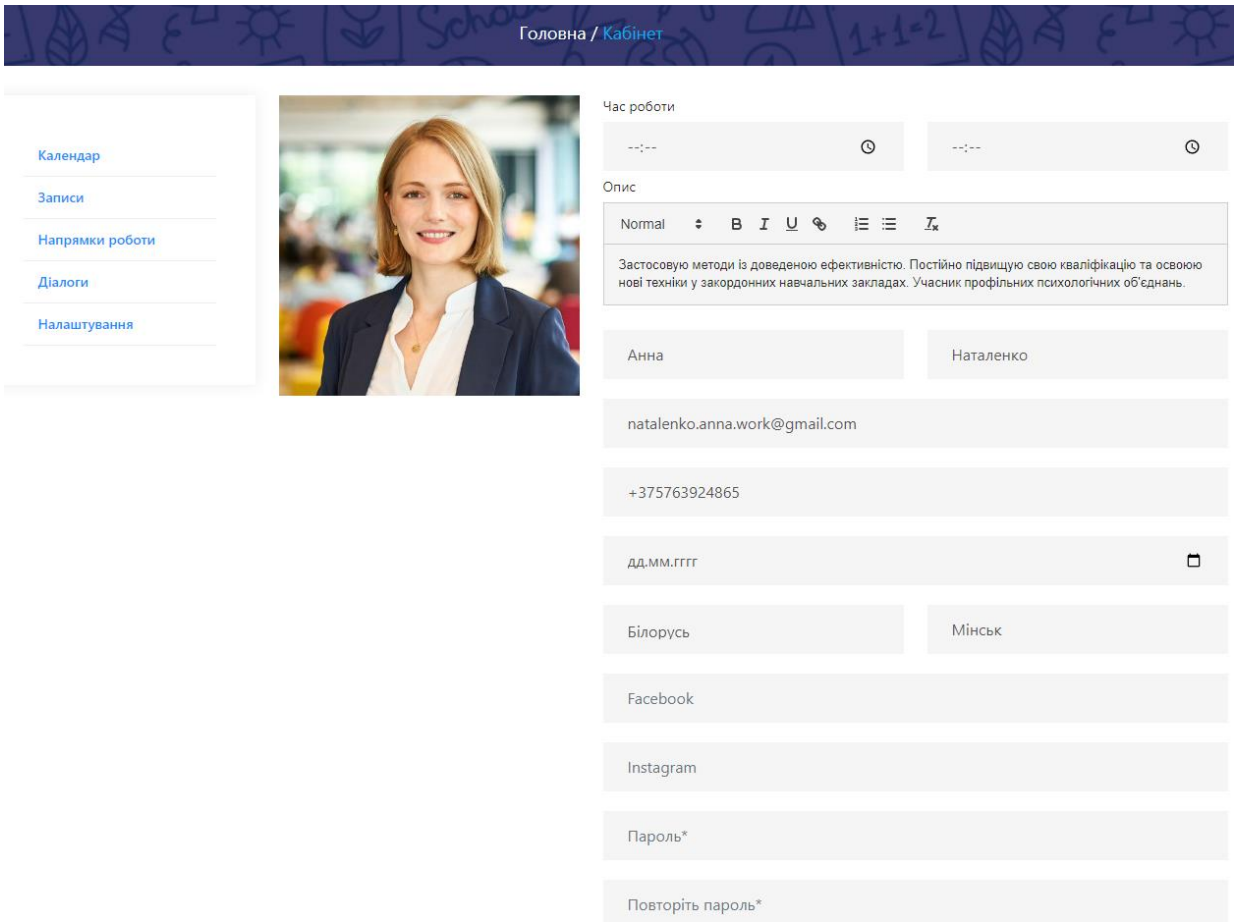


Рисунок 4.6 – Сторінка налаштування особистої інформації

Після завантаження та збереження особистого фото в налаштуваннях сторінки, фото стане відображатися в верхній частині сайту, так званому «хедер», як це зображено на рисунку 4.7.

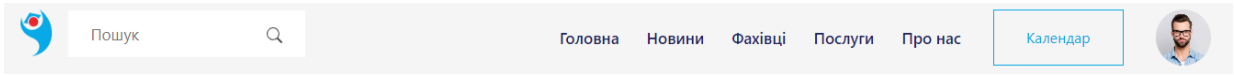


Рисунок 4.7 – Верхня частина сторінки

Фахівець має можливість змінювати свій час роботи не лише в налаштуваннях особистої сторінки, а й на головній сторінці кабінету де візуалізований календар.

Щоб запланувати онлайн-консультацію, фахівцеві потрібно перейти до «Напрямки роботи» та натиснути на кнопку «Додати напрям», як це зображено на рисунку 4.8.

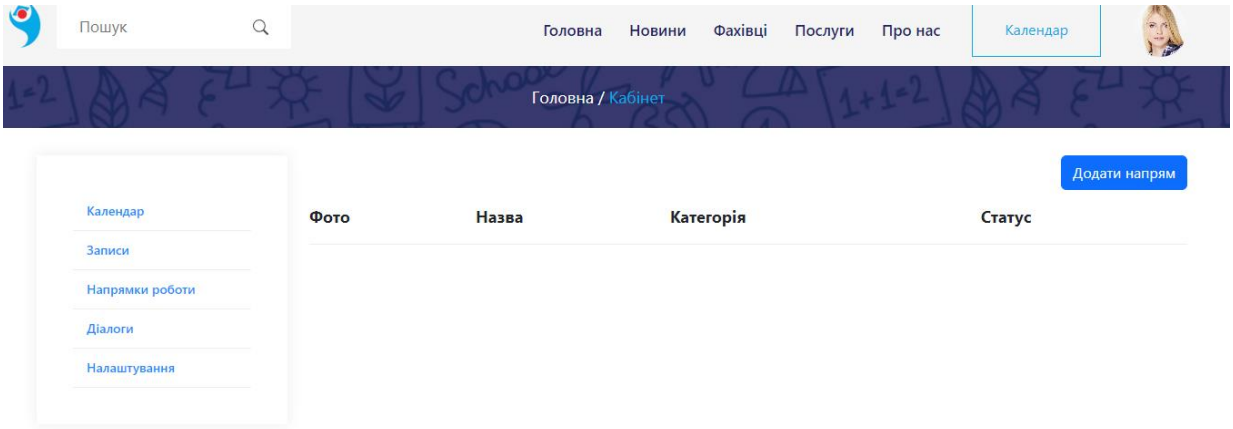


Рисунок 4.8 – Сторінка «Напрямки роботи»

Для успішного створення напряму роботи, потрібно заповнити такі поля:

– Заголовок;
– Напрям;
– Ціна;
– Короткий опис;
– Завантажити картинку;
– Вибрати статус.

Створення короткого опису має підтримку базових інструментів редагування тексту, а завантаження картинки додає покращене сприйнятися інформації про консультацію.

Сторінку зі створення напрямку роботи зображено на рисунку 4.9.

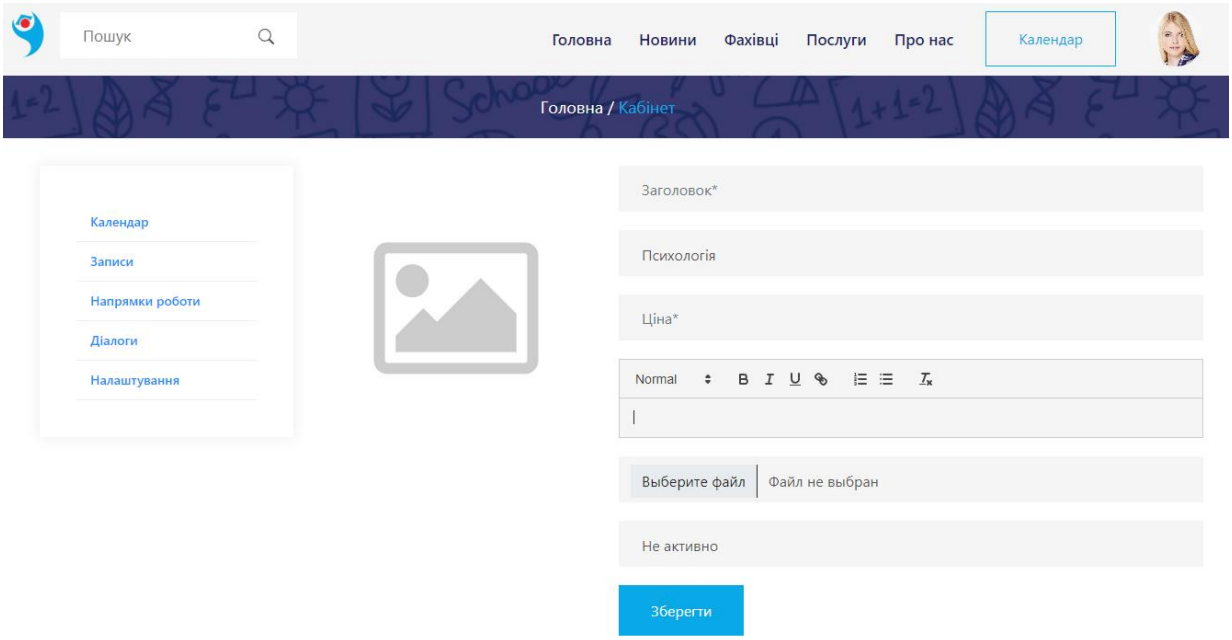


Рисунок 4.9 – Сторінка створення напрямку роботи

Приклад заповнення полів при створенні напрямку роботи зображений на рисунках 4.10-4.11. Після заповнення даними потрібно натиснути кнопку «Зберегти» де фахівець отримає повідомлення про успішне збереження даних як на рисунку 4.12.

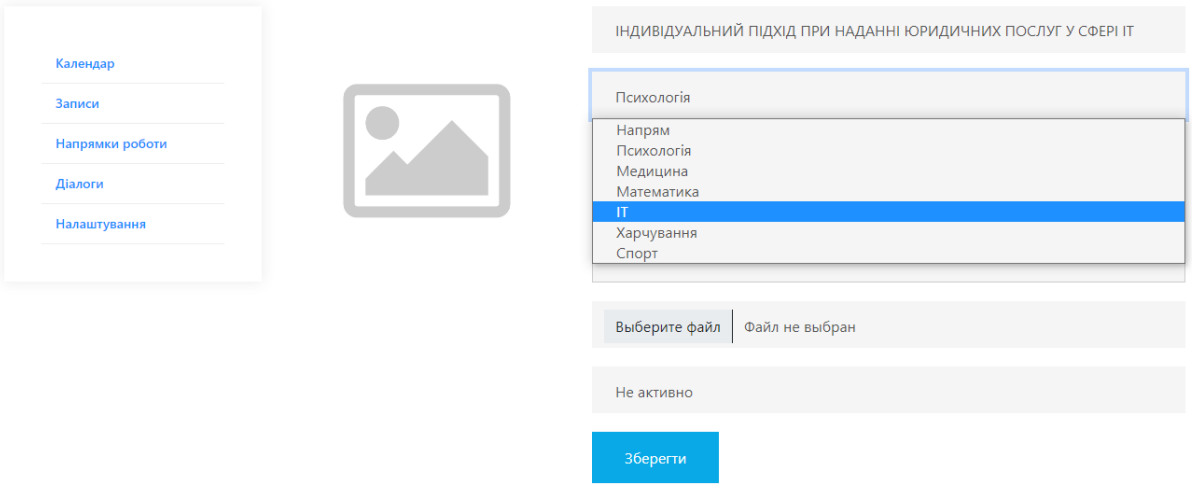


Рисунок 4.10 – Приклад вибору напрямку при створенні

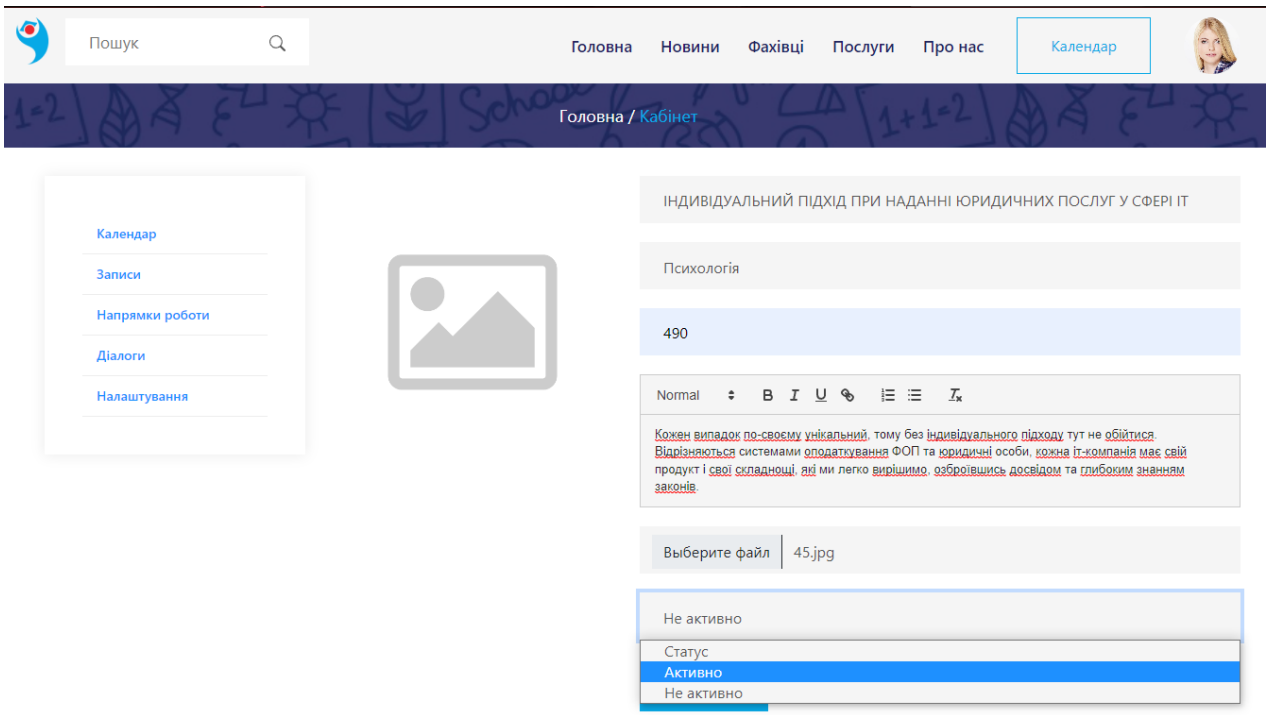


Рисунок 4.11 – Приклад створення напрямку роботи

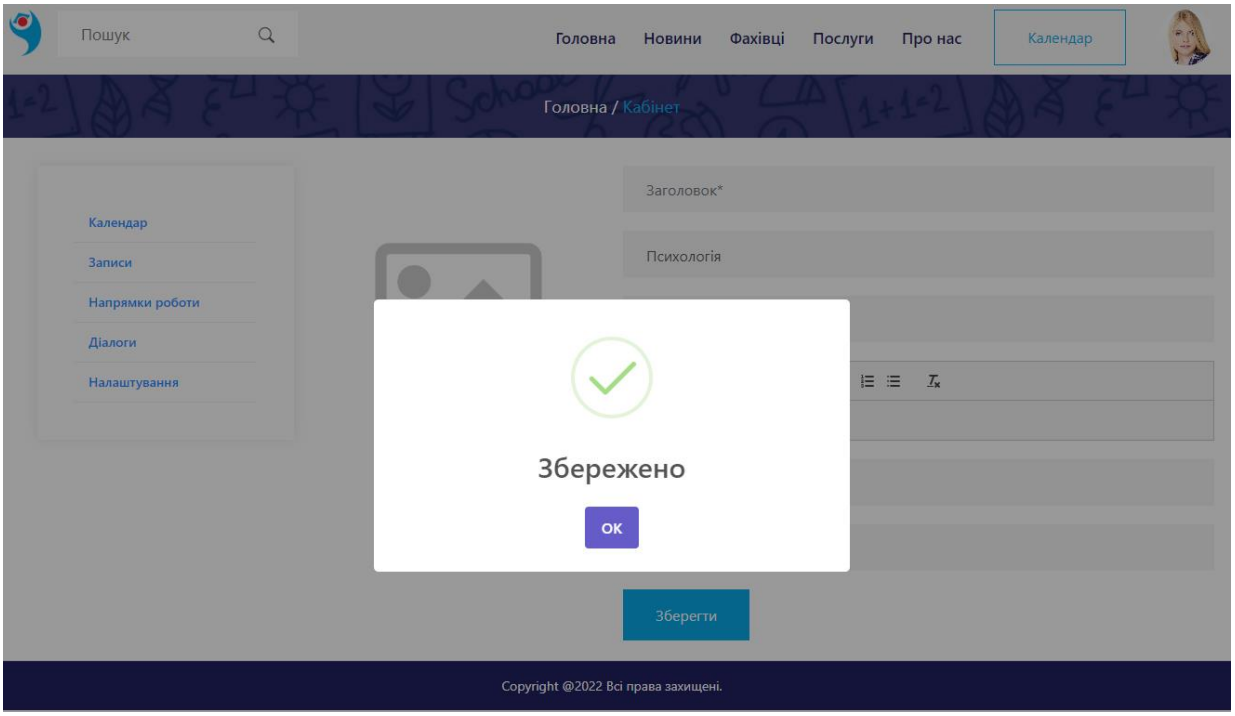


Рисунок 4.12 – Нотифікація про оновлення інформації

Після того, як перший слухач запишеться на одну з онлайн-консультацій на потрібний час, то у фахівця, в календарі відобразиться цей запис як на рисунку 4.13.

Та натиснувши на цей день, відобразиться детальна інформація про те, хто і на яку консультацію подав заявку на рисунку 4.14.

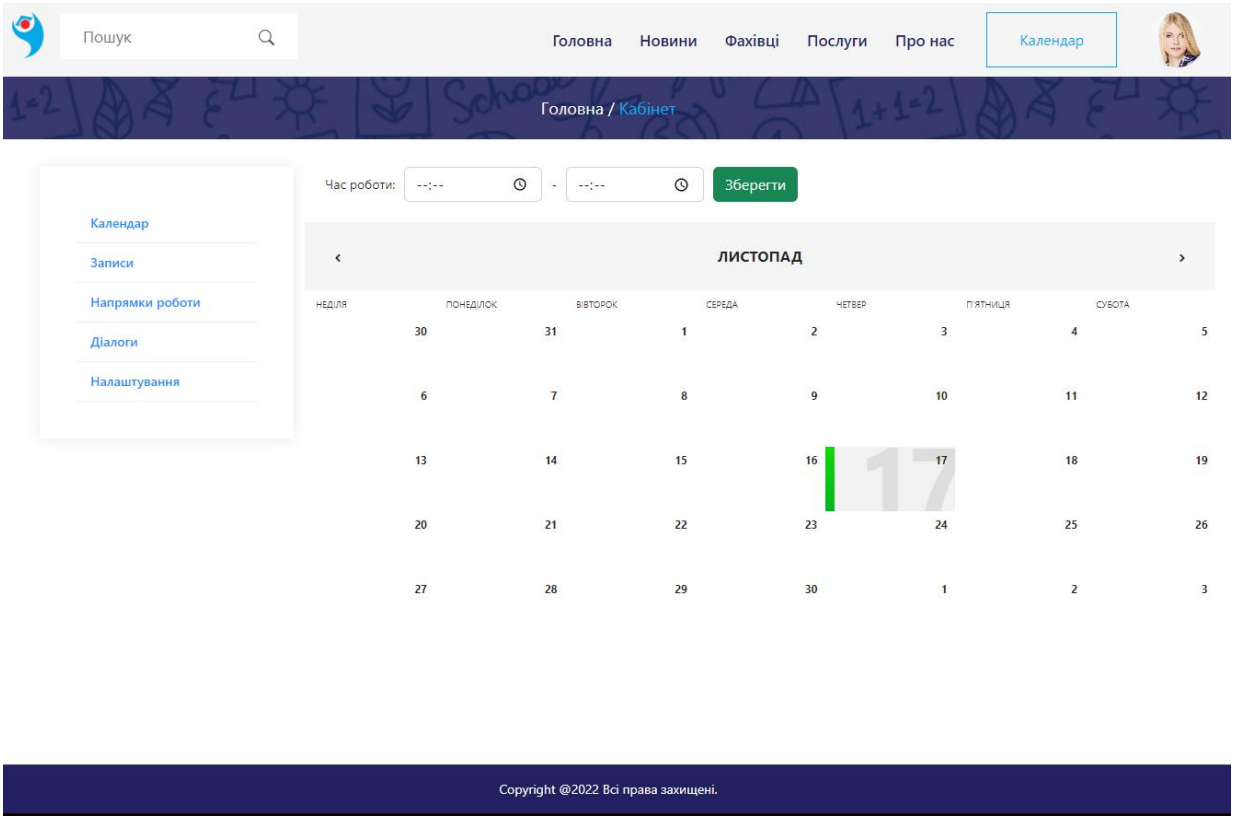


Рисунок 4.13 – Відображення онлайн-консультації в календарі

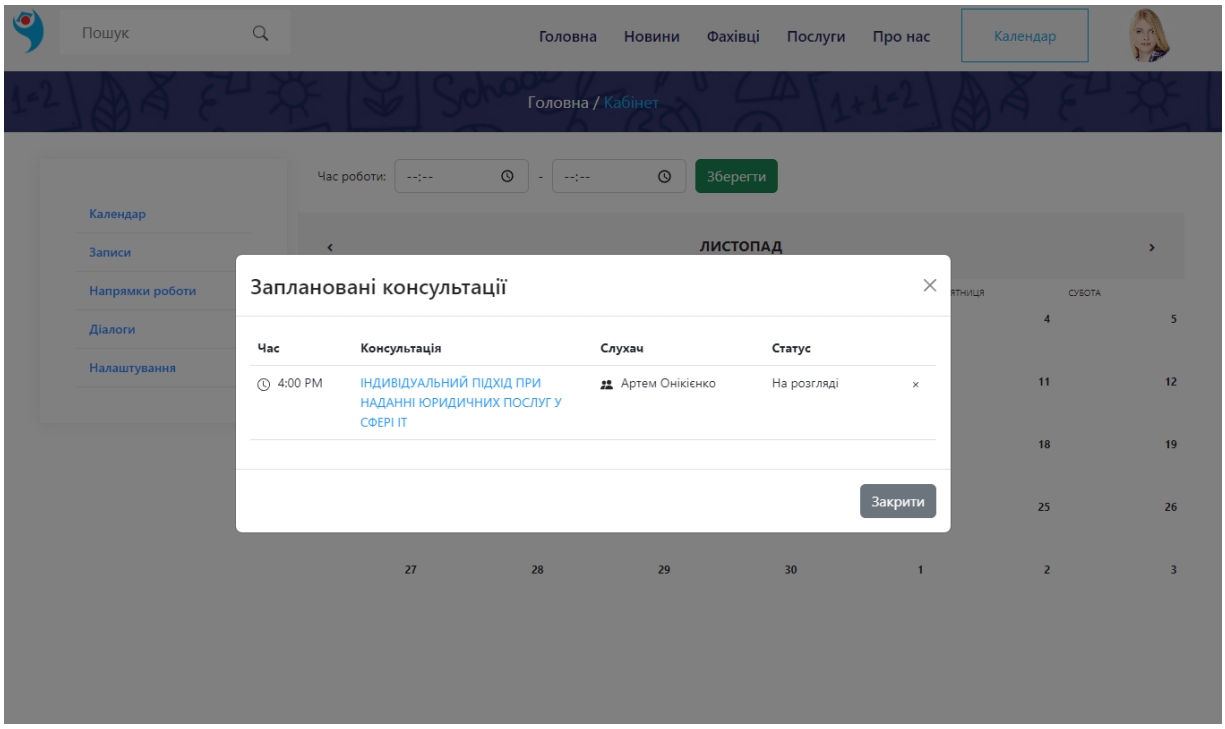


Рисунок 4.14 – Детальна інформація про заплановану консультацію

Для зміни статусу слухача, який подав заявку на онлайн консультацію потрібно перейти з головної сторінки кабінету до записів на рисунку 4.15. Там буде відображатися інформація про слухача та інформація, про те, чи зробив він оплату, як це зображено на рисунку 4.16.

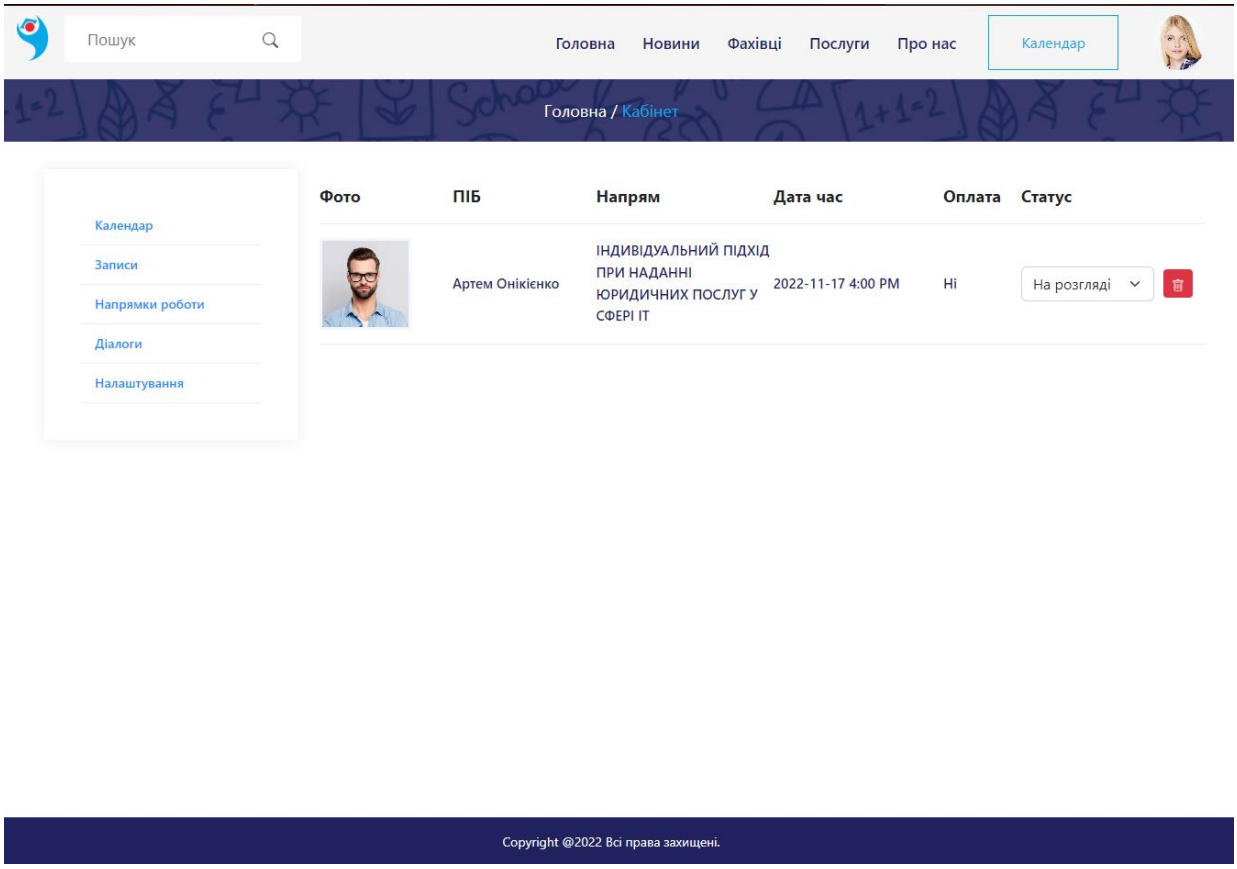


Рисунок 4.15 – Сторінка записів фахівця

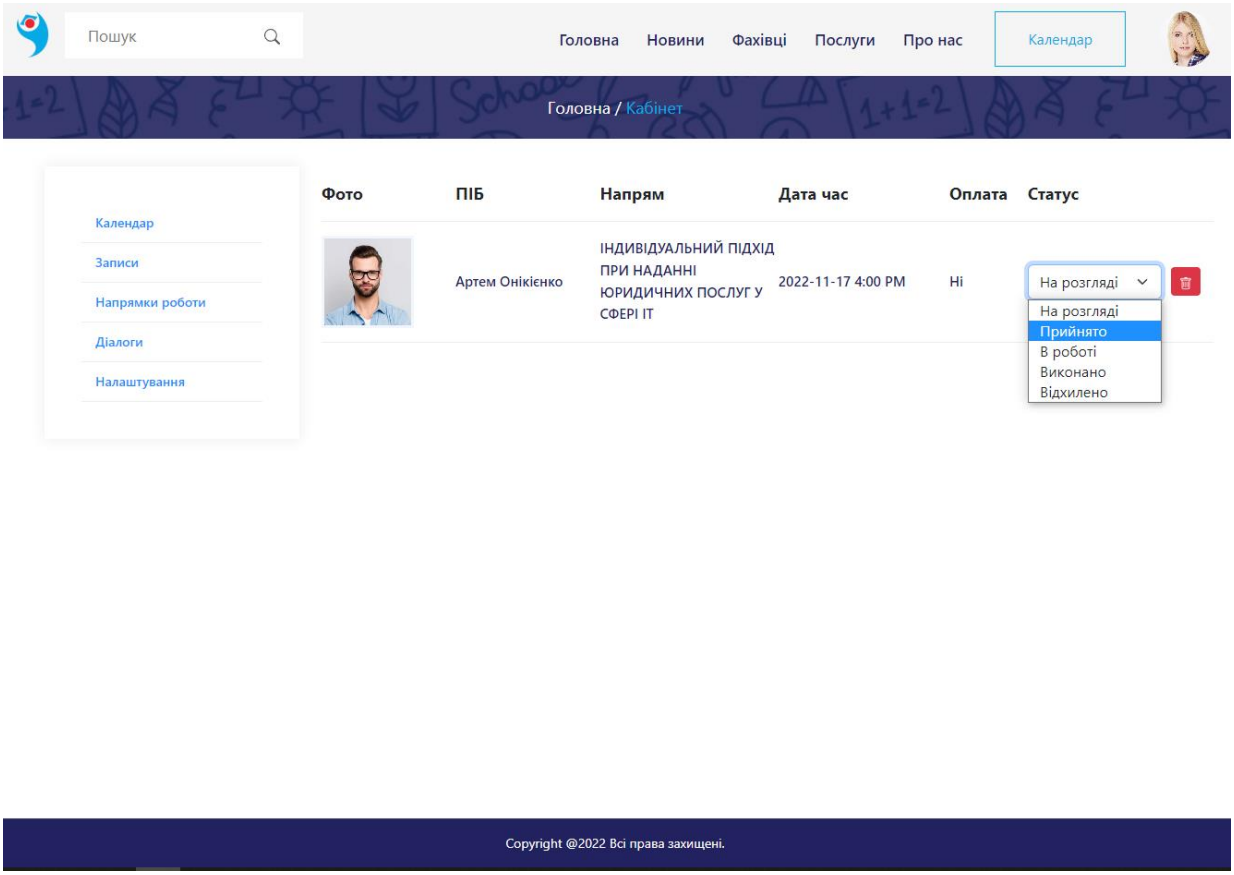


Рисунок 4.16 – Зміна статусу слухача

4.3 Використання інформаційної системи зі сторони слухача

Тепер потрібно подивитись на інформаційну систему зі сторони слухача. Далі зображена сторінка «Послуги» на рисунку 4.17. Де слухач може, використовуючи пошук, знайти потрібну йому консультацію.

Фільтрацію можливо виконати за такими критеріями:

- За категорією;
- За ціною.

Виконувати пошук можливо за назвою з головної сторінки сайту.

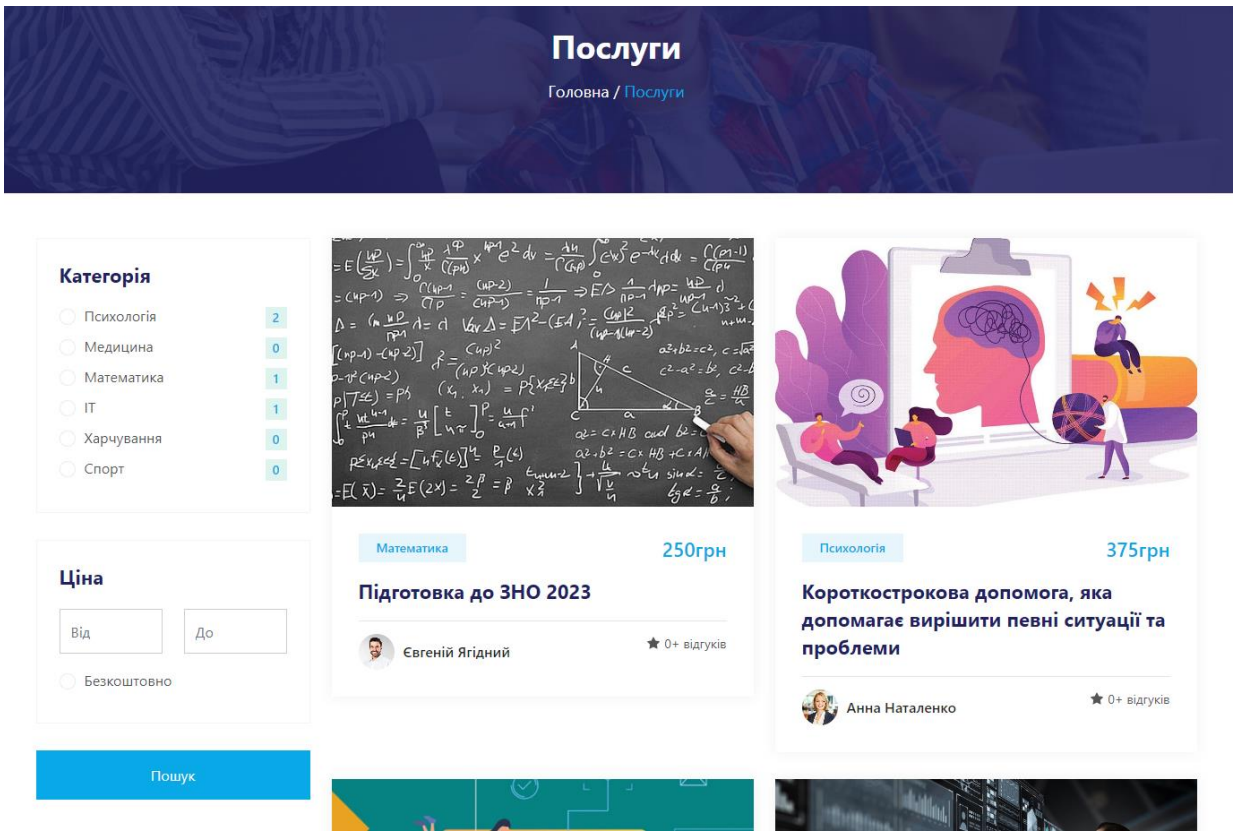


Рисунок 4.17 – Сторінка представлення послуг

Приклад використання фільтрів зображено в рисунку 4.18.

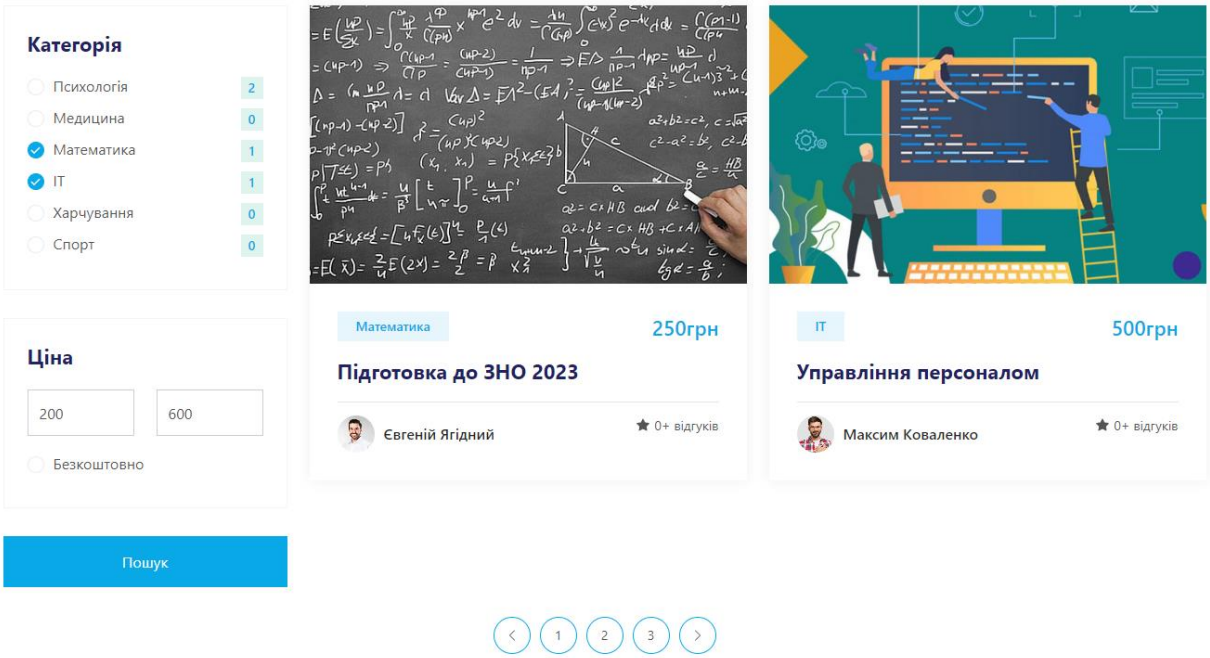


Рисунок 4.18 – Використання фільтрів за категорією та ціною

Далі зображена сторінка «Фахівці» на рисунку 4.19. Де слухач може, використовуючи пошук, знайти потрібного фахівця.

Фільтрацію можливо виконати за категоріями та країною

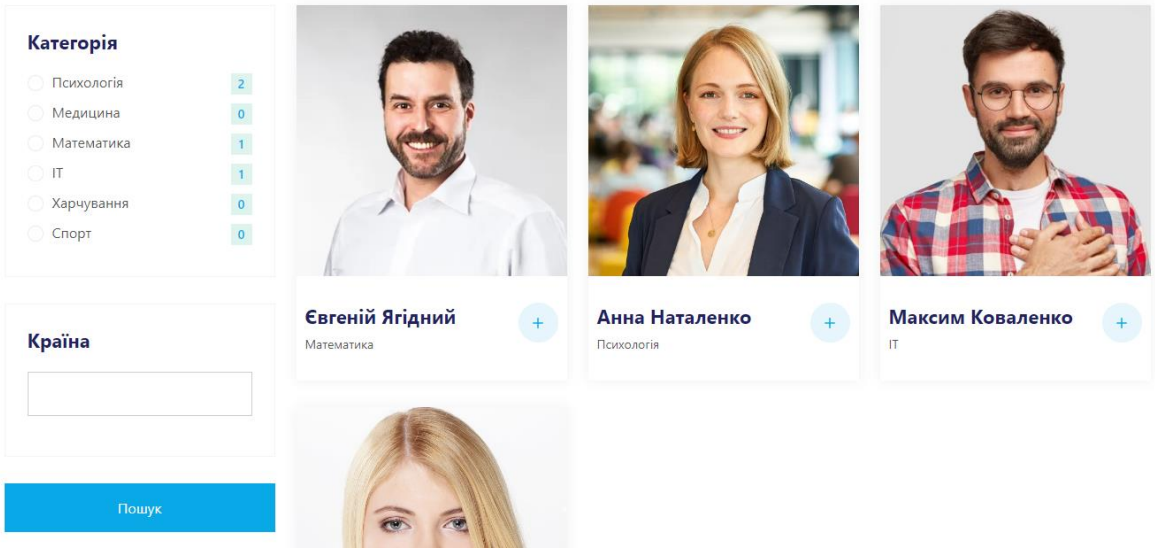


Рисунок 4.19 – Сторінка з фахівцями

Приклад використання фільтрів для пошуку потрібного фахівця зображено в рисунку 4.20.

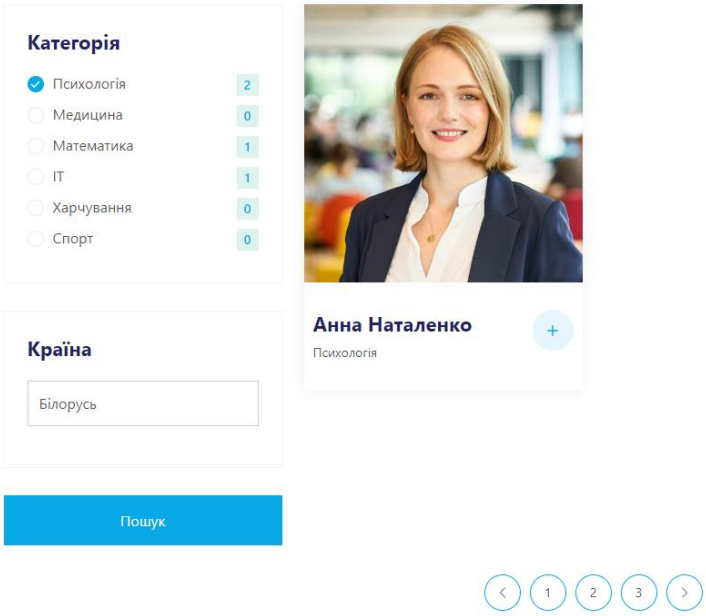


Рисунок 4.20 – Використання фільтрів за категорією та країною

Перейшовши на сторінку фахівця, можна побачити коротку інформацію про нього, перелік послуг фахівця, та можливість написати фахівцю особисте повідомлення, зображено на рисунках 4.21-4.22.

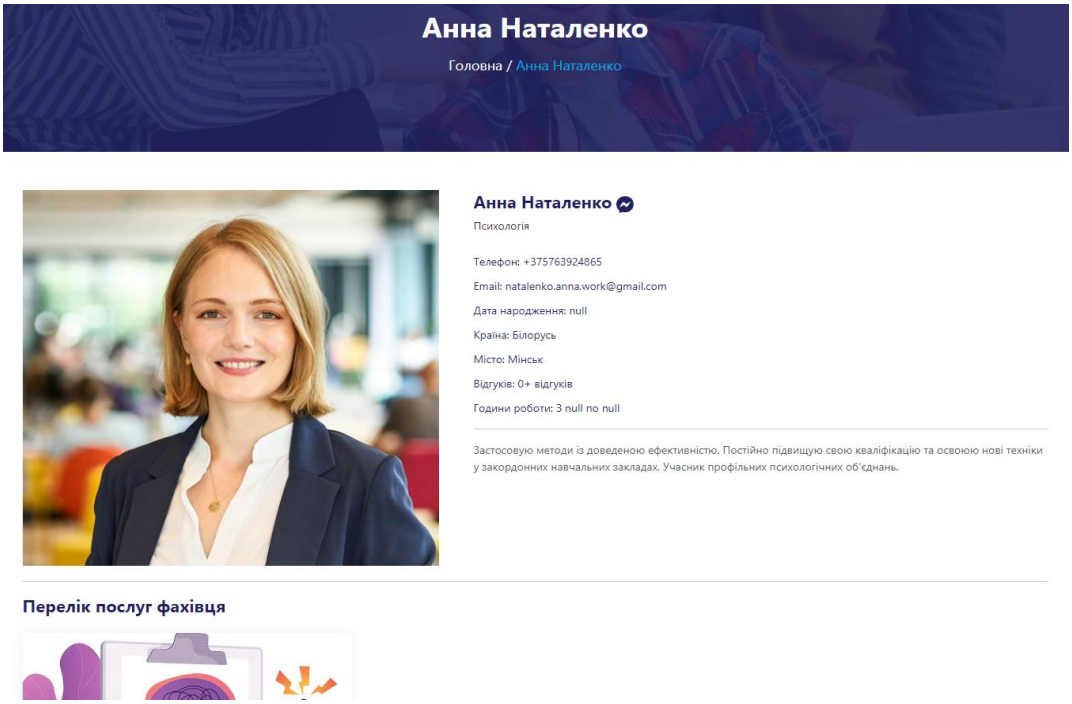


Рисунок 4.21 – Сторінка фахівця

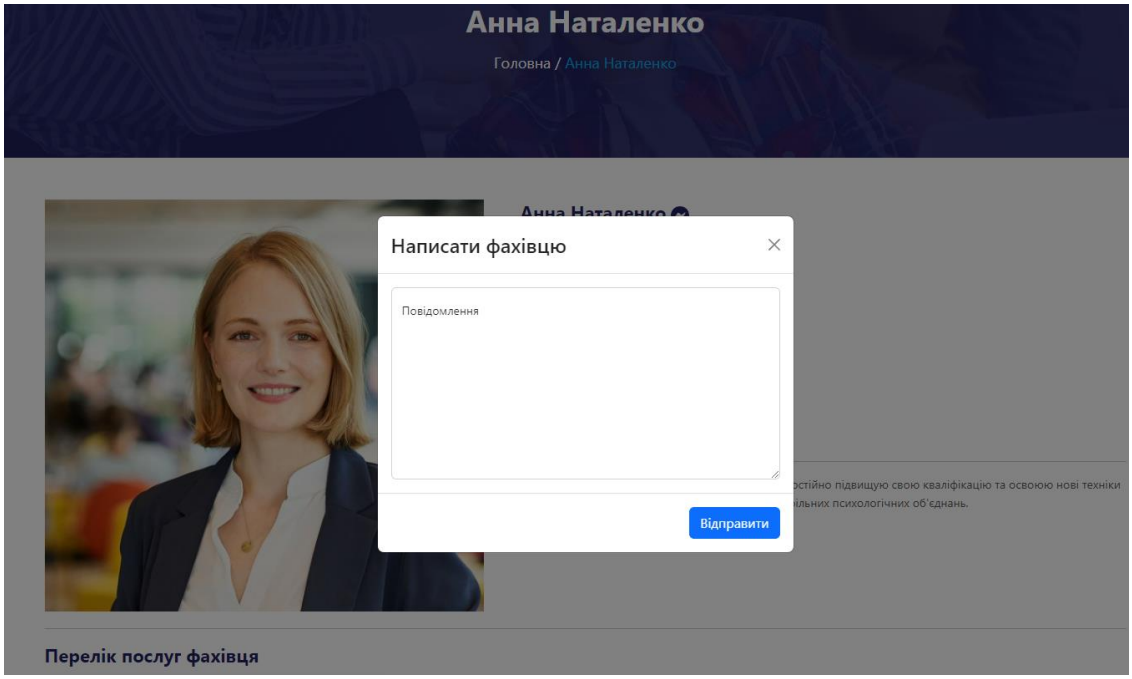


Рисунок 4.22 – Функція написання особистого повідомлення

Перейшовши на сторінку бажаної послуги відкривається можливість записати на онлайн консультацію. Наявна коротка інформація про фахівця та можливість з ним зв’язатися (рис 4.23).

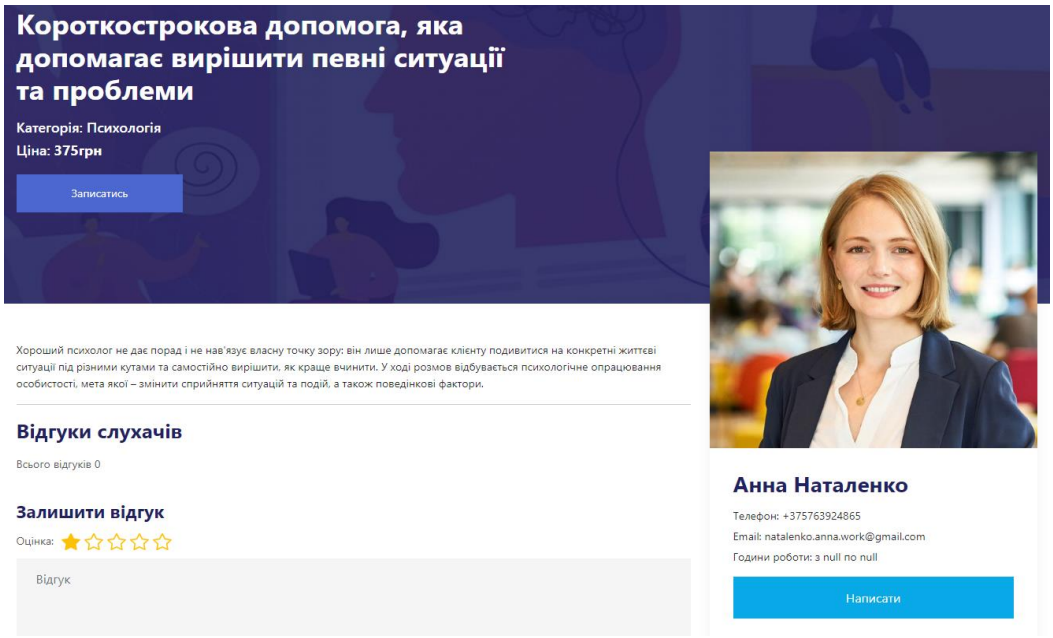


Рисунок 4.23 – Сторінка послуги фахівця

Після подання заявки, потрібно почекати на підтвердження фахівця (рис. 4.24).

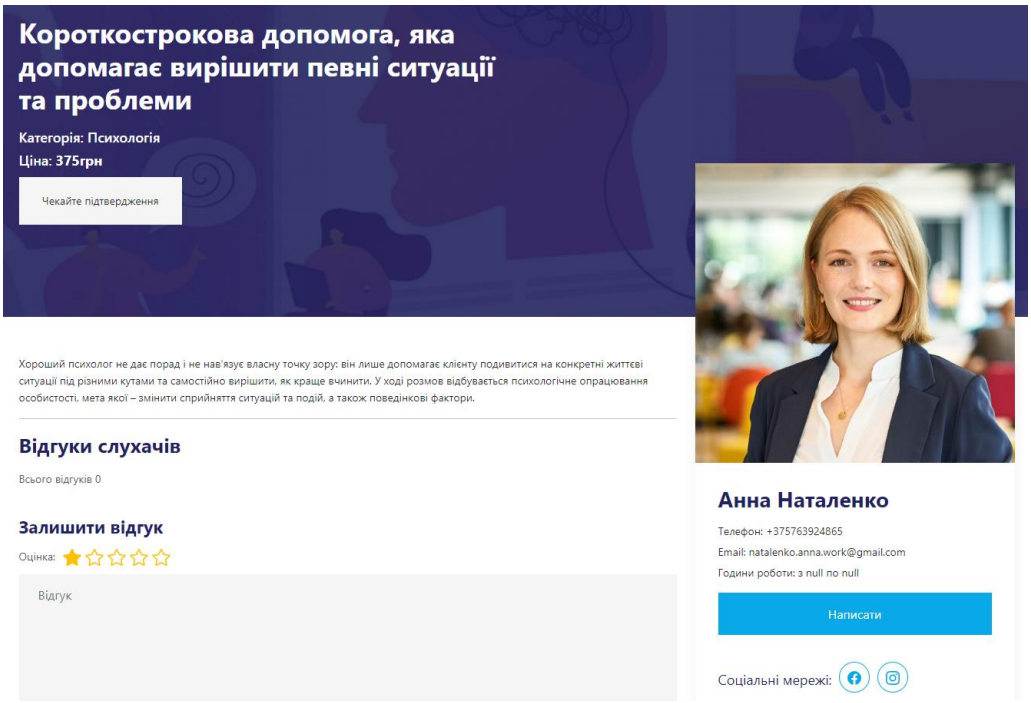


Рисунок 4.24 – Зміна статусу послуги на «очікуйте підтвердження»

Також, за зміною статусу заявки можна слідкувати в календарі особистого кабінету (рис. 4.25).

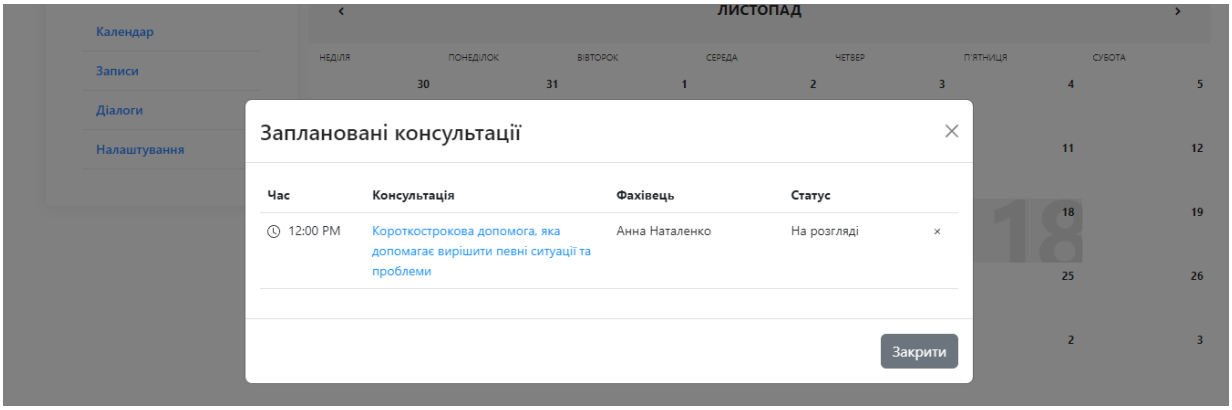


Рисунок 4.25 – Змінення статусів

Як тільки, фахівець прийме заявку, то слухач зможе перейти до оплати послуги (рис. 4.26).

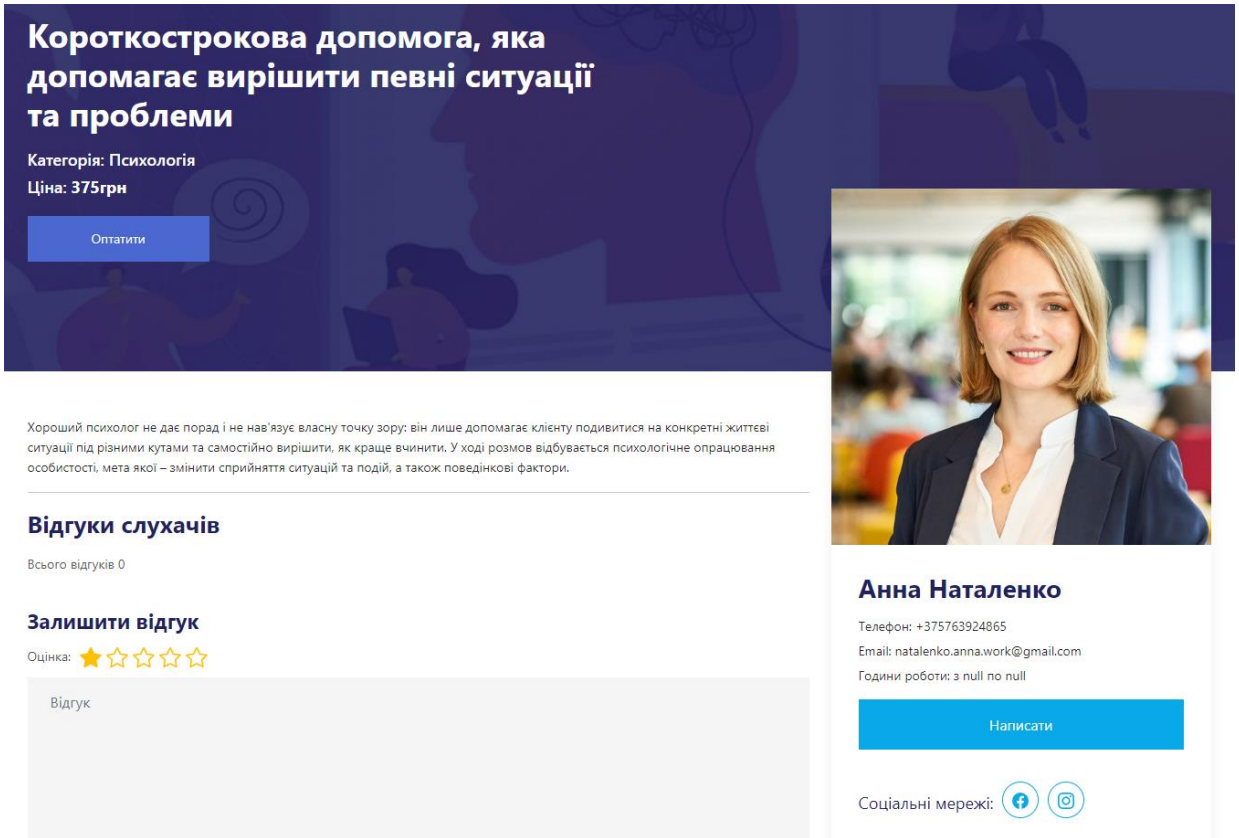


Рисунок 4.26 – Оплата послуги

Після оплати потрібно дочекатися підтвердження оплати фахівцем та переходити до відвідування онлайн консультації (рис. 4.27).

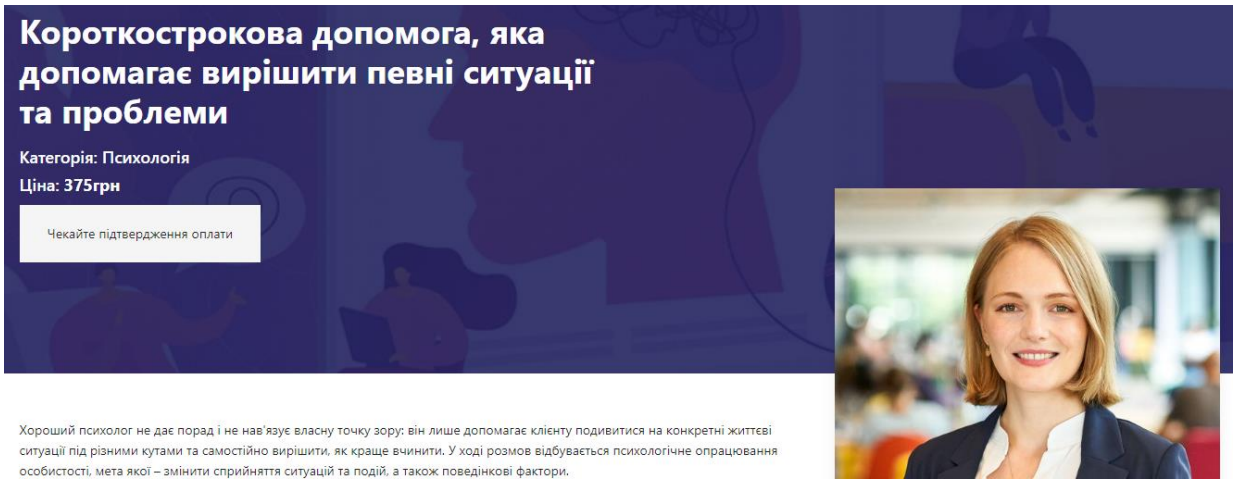


Рисунок 4.27 – Оплата послуги

Після відвідування онлайн консультації, фахівець змінює статус цієї консультації та для відвідувачів відкривається доступ до написання відгуку та залишання оцінки у вигляді 5 зірочок (рис. 4.28-4.29).

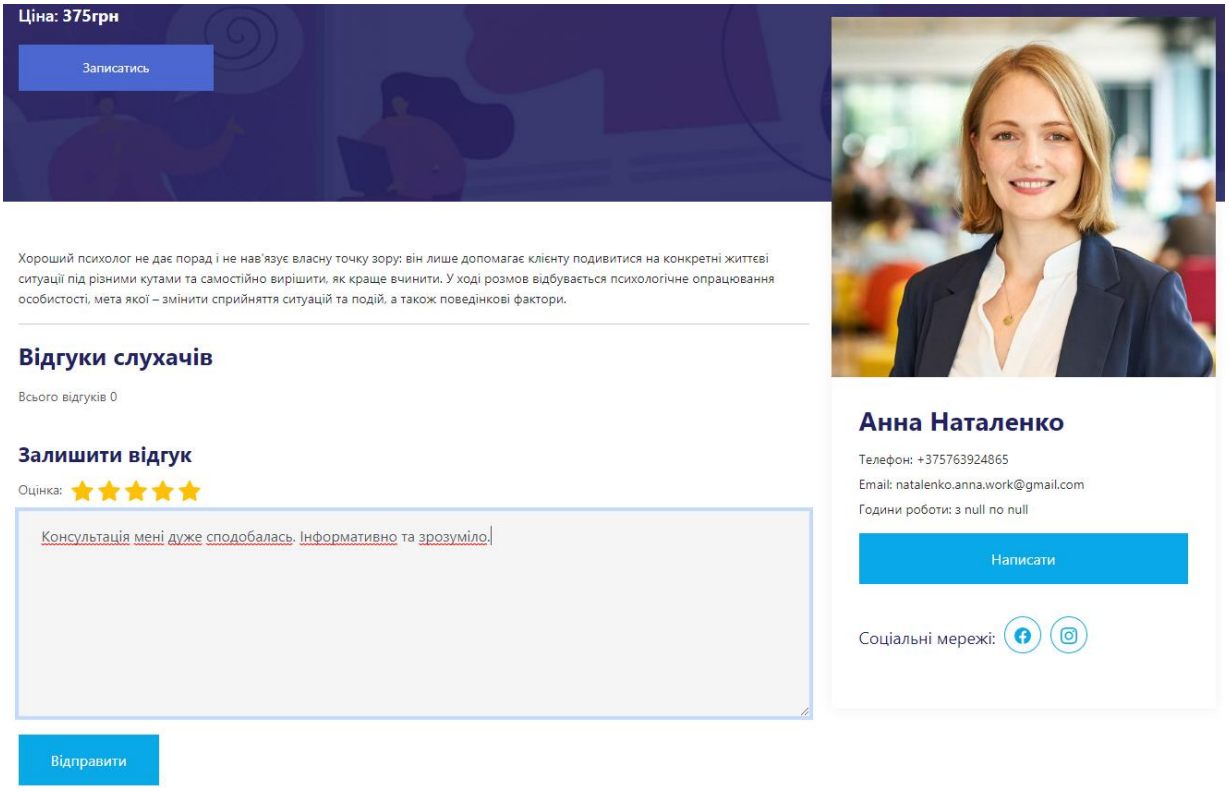


Рисунок 4.28 – Можливість залишити оцінку та відгук

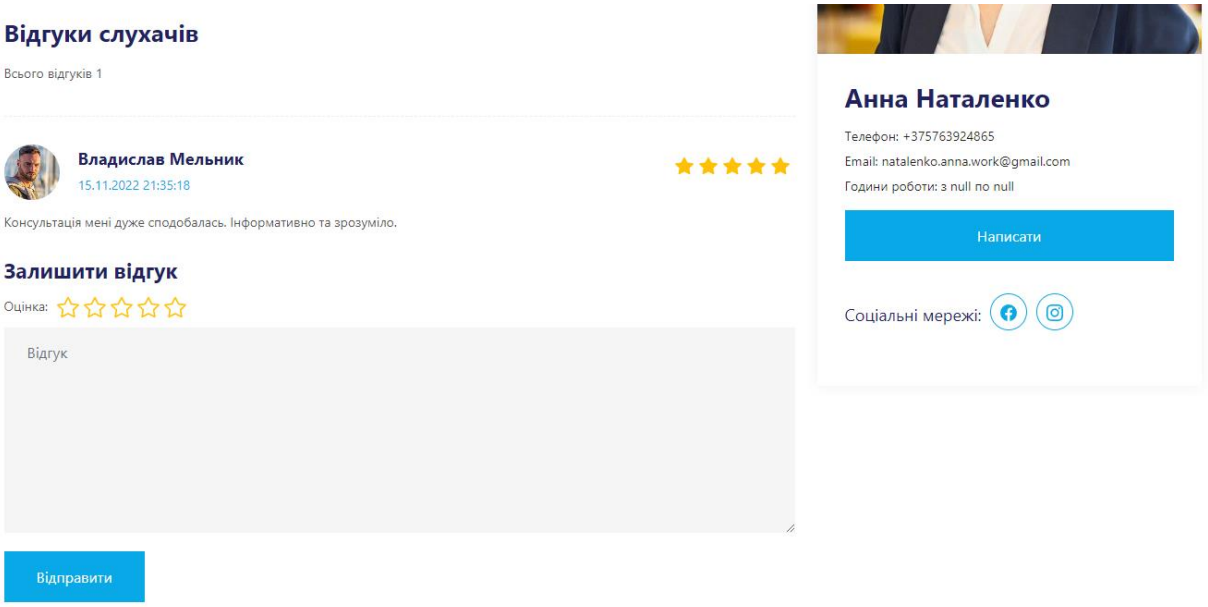


Рисунок 4.29 – Відгук та оцінка відвідувача

Особливу увагу потрібно приділити діалогу між фахівцем та слухачем який можливий через функціонал «Діалоги»(рис.4.30).

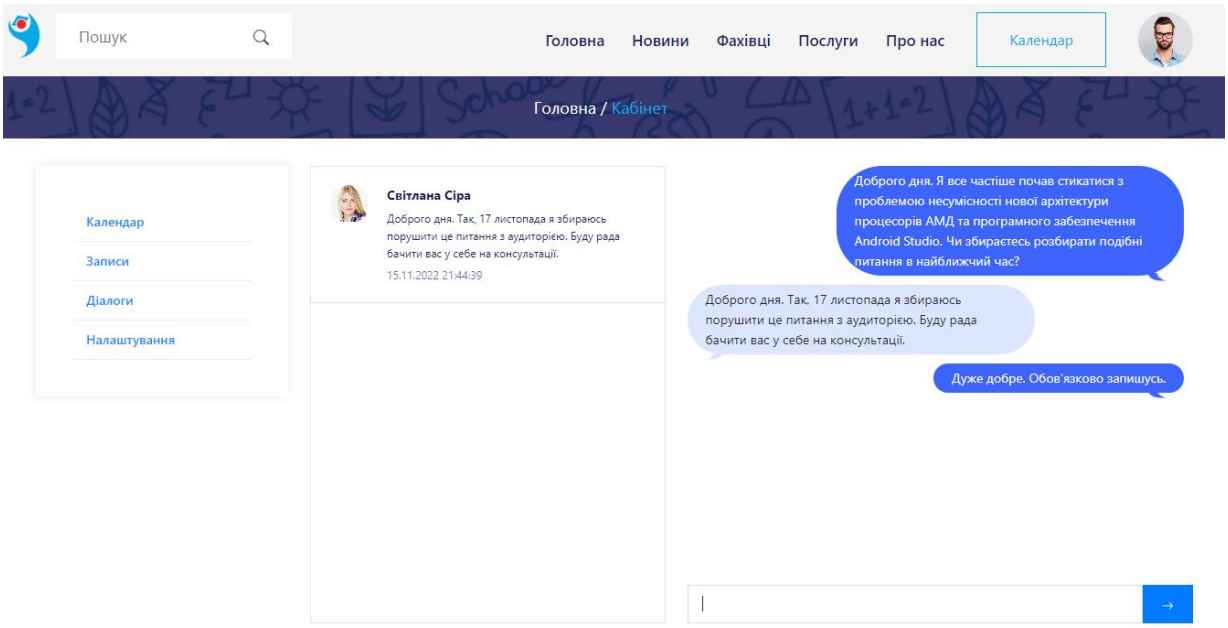


Рисунок 4.30 – Діалог між фахівцем та слухачем

4.4 Адміністрування інформаційної системи

Першим кроком є авторизація адміністратора на спеціальній сторінці (рис.4.31).

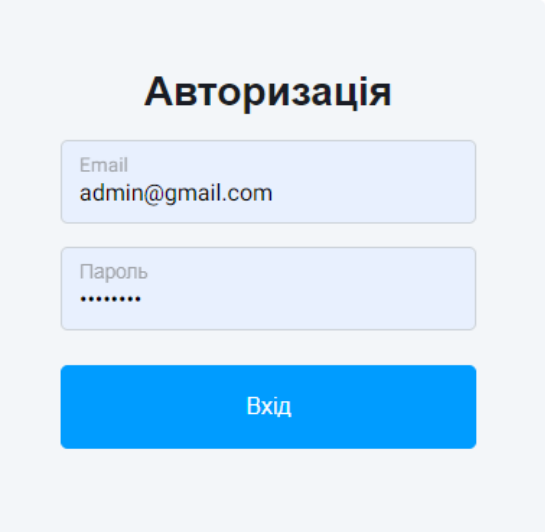


Рисунок 4.31 – Авторизація адміністратора

Адміністратор має у своєму функціоналі такі сторінки:

– Головна;
– Користувачі;
– Консультації;
– Категорії;
– Новини;
– Розсилання;
– Перейти до сайту.

На рисунках 4.32-4.34 зображена головна сторінка де можна редагувати головну інформацію про інформаційну систему, категорії та новини, а також слідкувати за новими користувачами.

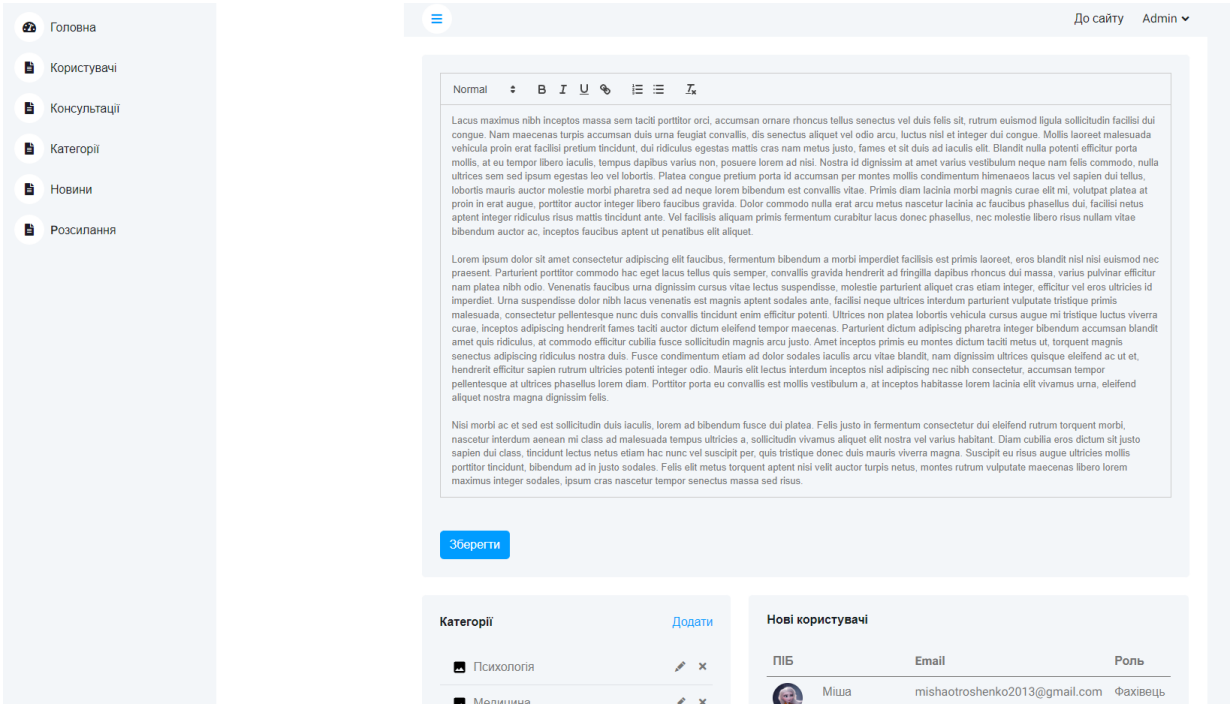


Рисунок 4.32 – Головна сторінка адміністратора

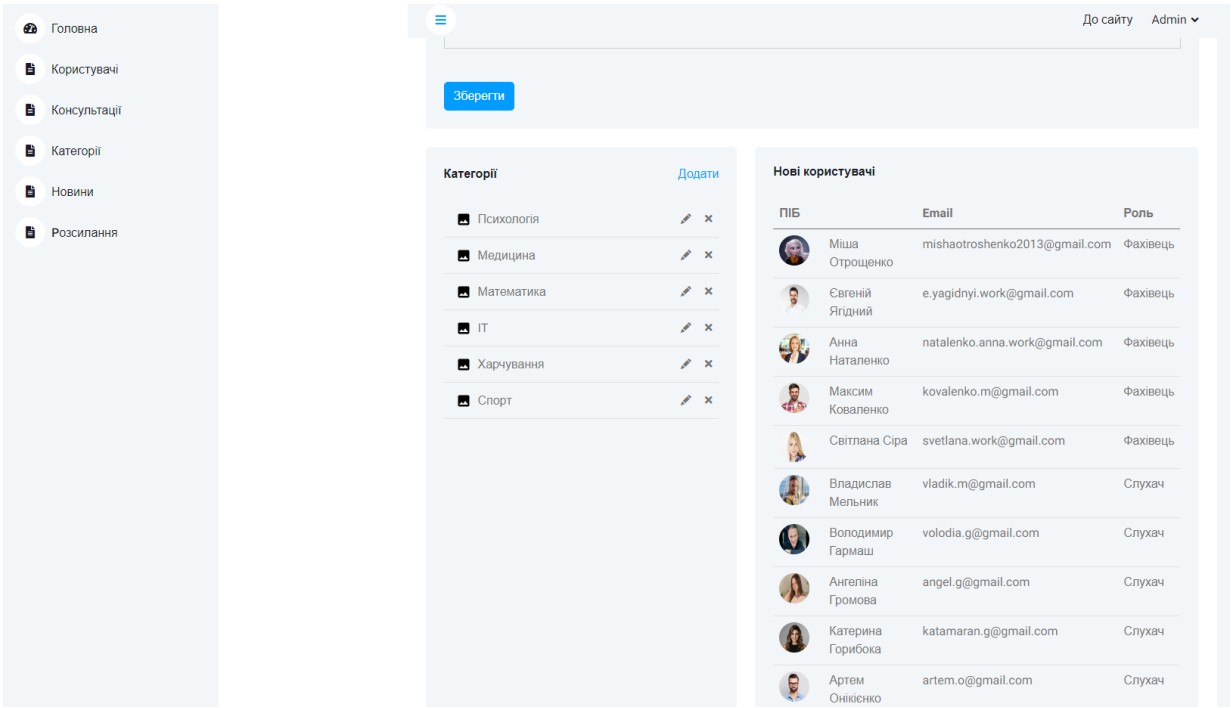


Рисунок 4.33 – Головна сторінка адміністратора (2)

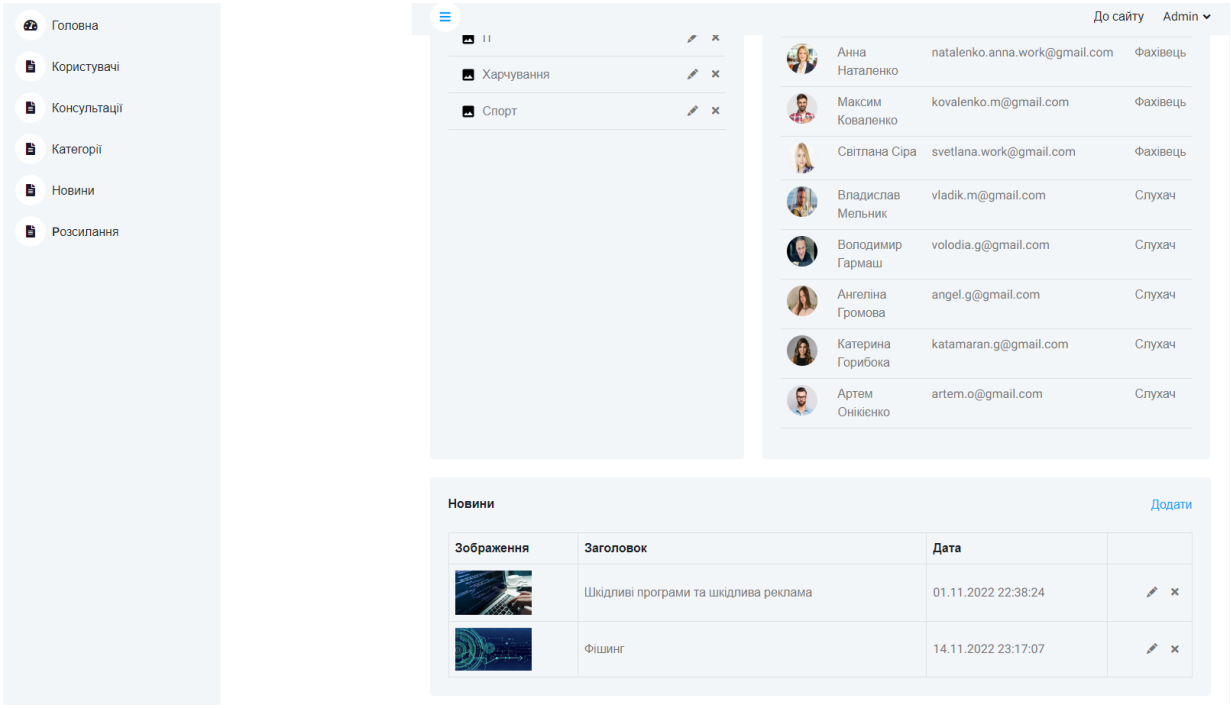


Рисунок 4.34 – Головна сторінка адміністратора (3)

На сторінці з користувачами, адміністратор може переглянути дані користувача, а також заблокувати або розблокувати користувача (рис. 4.35).

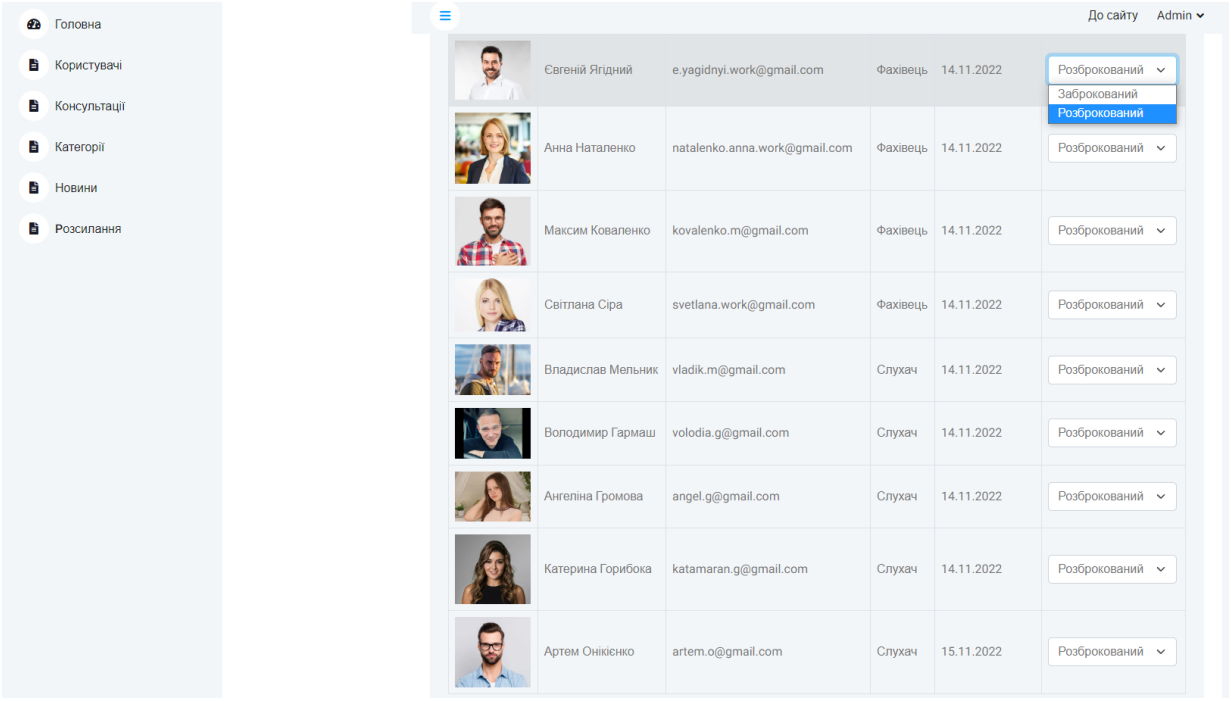


Рисунок 4.35 – Сторінка з користувачами

На сторінці «Консультації» адміністратор може переглядати всі наявні консультації, та видаляти консультації, які не відповідають вимогам або порушують правила використання інформаційної системи (рис. 4.36).

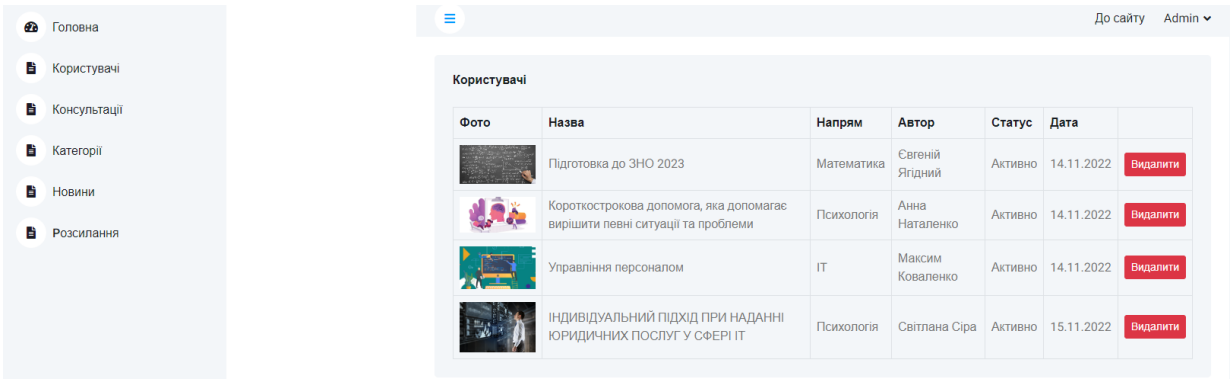


Рисунок 4.36 – Сторінка «Консультації»

На сторінці «Категорії» адміністратор може переглядати, додавати, редагувати та видаляти категорії послуг фахівців (рис. 4.37).

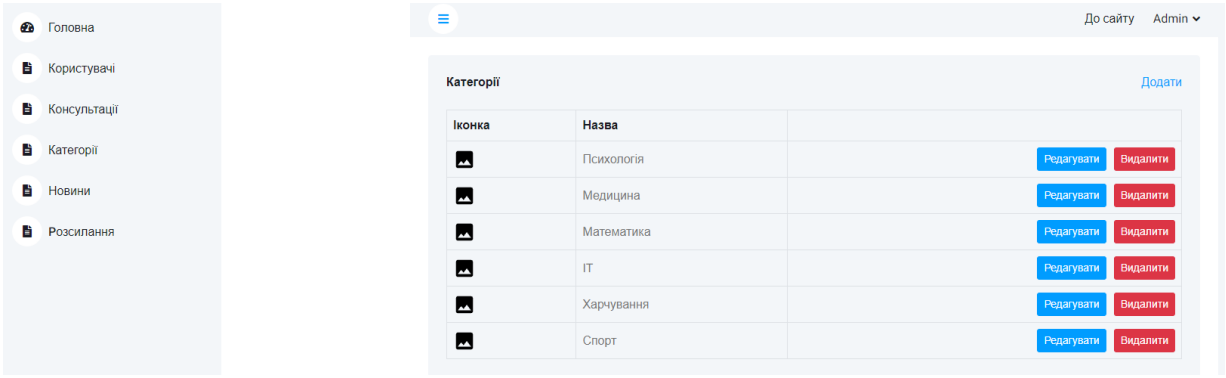


Рисунок 4.37 – Сторінка «Категорії»

На рисунках 4.38-4.39 зображено сторінка «Новини» де адміністратор може переглядати, редагувати, видаляти та додавати новини.

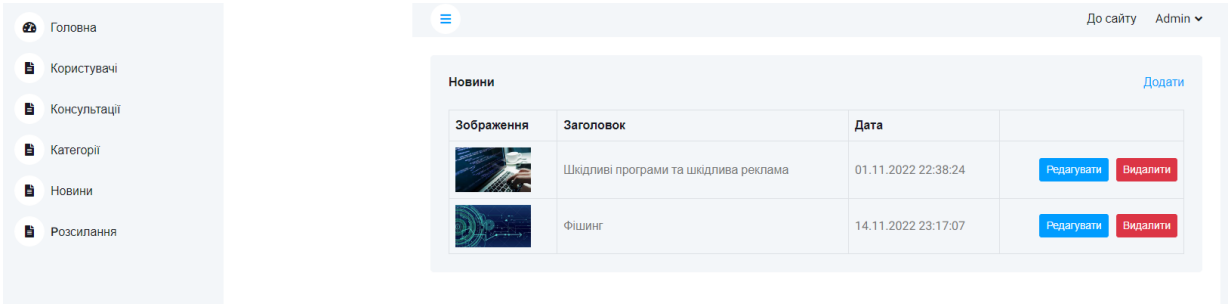


Рисунок 4.38 – Перегляд новин

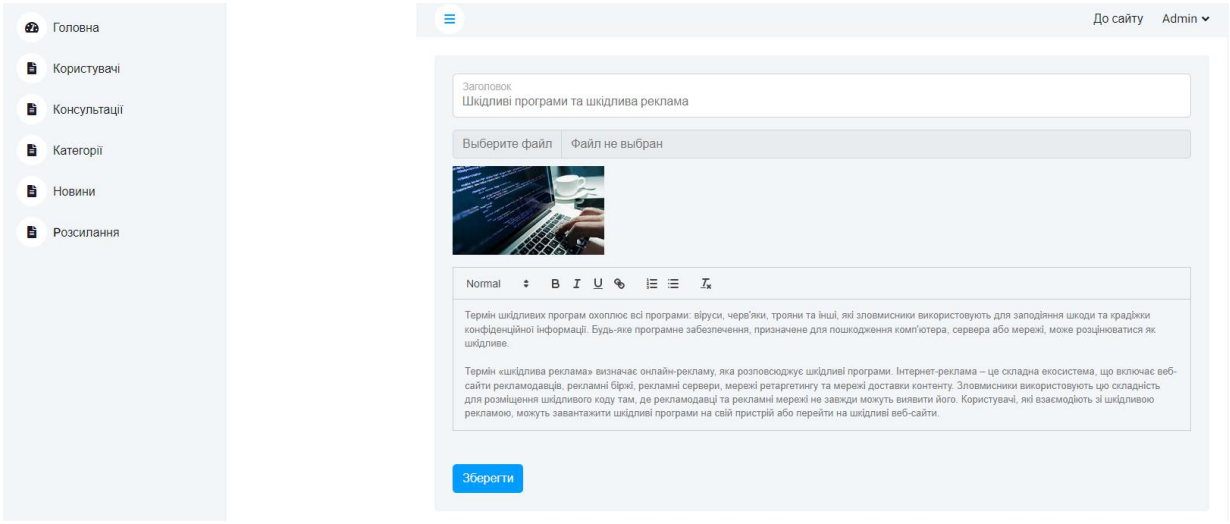


Рисунок 4.39 – Редагування новин

На сторінці «Розсилання» наявний функціонал для створення розсилки користувачам необхідної інформації, приклад зображено на рисунку 4.40.

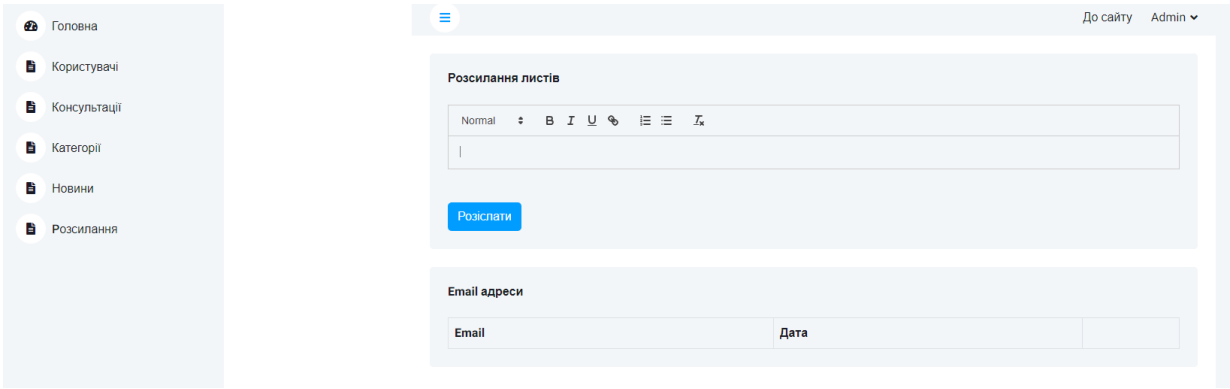


Рисунок 4.40 – Сторінка «Розсилання»

При розсиланні потрібного тексту на пошти які зазначені, висвічується віконце підтвердження операції як на рисунку 4.41.

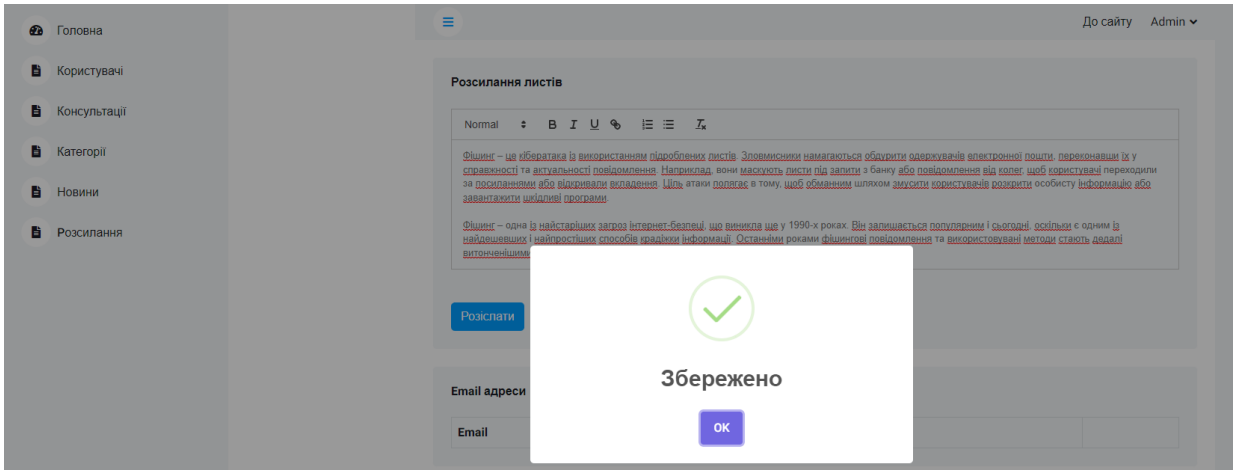


Рисунок 4.41 – Розсилання тексту

Інформаційна система також має адаптив на рисунках 4.42.

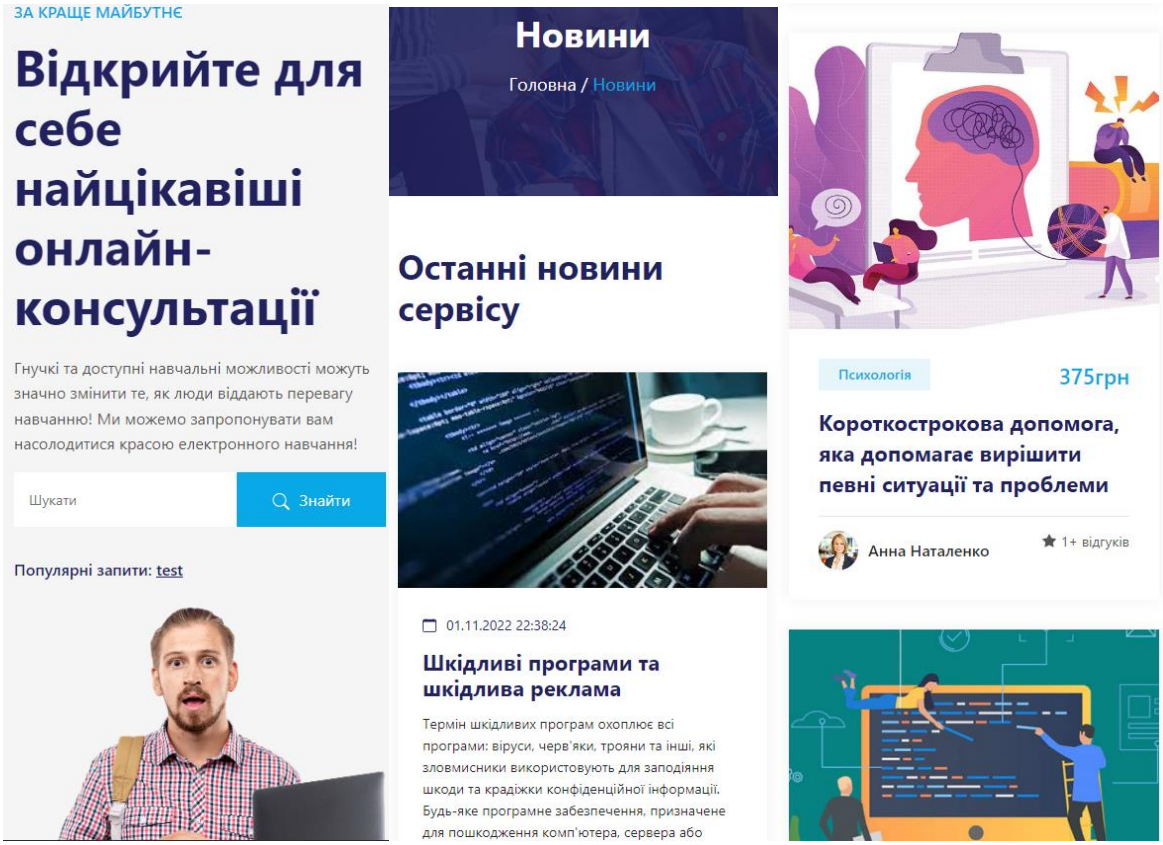


Рисунок 4.42 – Адаптив

ВИСНОВКИ

Під час виконання дипломної роботи була досягнута мета створення інформаційної системи для онлайн-консультацій за різними напрямками.

Переглянувши тему та виконавши деякі порівняння було обрано власний напрямок розвитку даної системи для покращення функціонала та для того, щоб охопити якомога більшу аудиторію. Задля цього проведено аналіз існуючих інформаційних систем аналогічного направлення. Враховуючи дослідження конкурентних інформаційних систем, було додано додаткові можливості до функціонала системи, які покращили комфорт управління. Використання календарного вигляду для орієнтації між запланованими онлайн-консультаціями оптимізувало відображення великої кількості інформації та покращило комфортність використання інформаційної системи в цілому.

Проведено планування робіт, побудовано графік робіт та виконано аналіз можливих ризиків.

Розроблена інформаційну систему має головну відзнаку від аналогів – автоматизоване створення онлайн-консультації за допомогою самостійної системи створення покликання на Zoom. При цьому інформаційна система універсальна та підтримує різні напрямки.

Після створення інформаційної системи для онлайн-консультацій проведено тестування головного функціоналу та виправлені помилки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Riaz Ahmed: Complete Web Development Package for Beginners / Independently published, 2021. – 392 с.

2. Riaz Ahmed: Create Rapid Web Applications in Oracle Application Express / Independently published, 2019. – 436 с.

3. Jennifer Robbins: Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics / O'Reilly Media, 2018. – 808 с.

4. Jon Duckett: Front-End Back-End Development with HTML, CSS, JavaScript, jQuery, PHP, and MySQL / Wiley, 2022. – 677 с.

5. Paul McFedries: Web Coding & Development All-in-One For Dummies / For Dummies, 2018. – 848 с.

6. Laurence Lars Svekis, Maaike van Putten, Rob Percival: JavaScript from Beginner to Professional / Packt Publishing, 2021. – 544 с.

7. Randy Connolly, Ricardo Hoar: Fundamentals of Web Development / Pearson, 2017. – 1248 с.

8. Mayur Borse: Modern Web Development with Deno / BPB Publications, 2022. – 200 с.

9. Terry Felke-Morris: Web Development and Design Foundations with HTML5 / Pearson, 2018. – 720 с.

10. Chris Northwood: The Full Stack Developer: Your Essential Guide to the Everyday Skills Expected of a Modern Full Stack Web Developer / Apress, 2018. – 365 с.

11. Brad Williams, Justin Tadlock, John James Jacoby: Professional WordPress Plugin Development / Wrox, 2020. – 480 с.

12. Mayur Borse: The Full Stack Developer: Modern Web Development with Deno / BPB Publications, 2022. – 200 с.

13. Noel Rappin: The Full Stack Developer: Modern Front-End Development for Rails / Pragmatic Bookshelf, 2022. – 410 с.

14. Ethan Brown: Web Development with Node and Express / O'Reilly Media, 2019. – 346 с.

15. Antonio Mele, Bob Belderbos: Django 4 By Example / Packt Publishing, 2022. – 764 с.

16. MEDIKIT [Електронний ресурс] – Режим доступу до ресурсу: https://medikit.ua/ (дата звернення: 18.10.2022)

17. Apollo 24|7 [Електронний ресурс] – Режим доступу до ресурсу: https://www.apollo247.com/ (дата звернення: 18.10.2022)

18. HARLEY THERAPY [Електронний ресурс] – Режим доступу до ресурсу: https://www.harleytherapy.co.uk/ (дата звернення: 18.10.2022)

19. Sammie Smih: FULL STACK WEB DEVELOPMENT: Everything Beginners to Expert Guide on Modern Full-Stack Web Development Using Modern Web Development Tools / Pearson, 2022. – 590 с.

20. Ben Shaw, Saurabh Badhwar, Andrew Bird, Bharath Chandra, Chris Guest: Web Development with Django: Learn to build modern web applications with a Python-based framework / Packt Publishing, 2021. – 826 с.

21. Nick Samoylov: Learn the fundamentals of Java Programming with this updated guide with the latest features / Packt Publishing, 2022. – 748 с.

22. Terry Matula: Laravel Applications Development Cookbook / Packt Publishing, 2020. – 274 с.

23. Matt Stauffer: Laravel: Ups and Running: A Framework for Building Modern PHP / O'Reilly Media, 2022. – 454 с.

24. Jamie Chan, Kindle Edition: Java for Beginners with Hands-on Project / Learn Coding Fast, 2017. – 237 с.

25. Gene Kim, Jez Humble: How to Create World-Class Agility, Reliability, & Security in Technology Organizations Paperback / IT Revolution Press, 2021. – 528 с.

26. Tanya Reilly: A Guide for Individual Contributors Navigating Growth and Change / O'Reilly Media, 2022. – 256 с.

27. Harold Abelson, Gerald Jay Sussman: Structure and Interpretation of Computer Programs / The MIT Press, 1996. – 657 c.

28. Vinicius M. Grippa, Sergey Kuzmichev: Learning MySQL: Get a Handle on Your Data / O'Reilly Media, 2021. – 632 c.

29. Joel Murach: Murach's MySQL / Mike Murach & Associates, 2019. – 628 c.

30. Charles Bell: Expert MySQL / Apress, 2021. – 626 c.

ДОДАТОК А. ПЛАНУВАННЯ РОБІТ

А.1 ІДЕНТИФІКАЦІЯ МЕТИ ІТ-ПРОЕКТУ

Сутність деталізації мети проекту за допомогою SMART-методу випливає з розшифровки термінів, які формують його назву: конкретна (Specific), вимірювана (Measurable), досяжна (Achievable), реалістична (Relevant), обмежена у часі (Time-framed).

S – конкретність, специфічність. Вимагає щоб сформульована мета давала чітке якісне уявлення про специфічні унікальні та інноваційні властивості майбутнього продукту проекту порівняно з іншими його альтернативами.

M – вимірюваність. Передбачає показників вартості які вимірюються. При відсутності фізичних способів та інструментарію виміру використовуються експерти – як інструмент для виміру.

A – узгодженість. Встановлює, що мета повинна випливати з реальних проблем, місії, стратегічних планів, планів розвитку, а також узгоджуватись з інтересами зацікавлених сторін проекту.

R – реалістичність, релевантність. Показує, що мета є такою, яку можливо досягти з урахуванням реально доступних ресурсних можливостей та обмежень (людських, фінансових тощо).

Поставлена мета є досяжною, адже вона сформулювалася на основі реально доступних ресурсних можливостей та проведеного аналізу вже наявних досліджень експертів у даній сфері.

T – обмеженість в часі. Зумовлює необхідність «прив’язки» мети до певних обґрунтованих термінів її досягнення (або початку та тривалості дій по її досягненню).

Таблиця А.1 – Деталізація мети методом SMART

Specific (конкретна)	Створити інтерактивний додаток.
Measurable (вимірювана)	Використовуючи мінімум ресурсів розробити якісний програмний продукт.
Achievable (досяжна)	Поставлена мета випливає у результаті актуальних проблем.
Relevant (реалістична)	У наявності є всі необхідні технічні та програмні засоби. Розробники достатньо кваліфіковані для виконання поставлених задач.
Time-framed (обмежена у часі)	Ціль має часове обмеження. Терміни досягнення мети проекту визначаються за домовленість замовником та виконавцем.

Після проведення аналізу методом SMART можна визначити кінцеву мету: вчасно створити якісний програмний продукт з використанням мінімальних витрат.

А.2 ПЛАНУВАННЯ ЗМІСТУ СТРУКТУРИ РОБІТ ІНФОРМАЦІЙНОЇ СИСТЕМИ

WBS – це графічне подання згрупованих елементів проекту у вигляді пакета робіт, які ієрархічно пов’язані з продуктом проекту. На верхньому першому рівні WBS фіксується продукт проекту. Він повинен відповідати продукту проекту. Наступний II рівень відповідає діям або основним заходам для досягнення продукту проекту. Потім триває розбивка цих дій доти, поки не відбувається виконання дій елементарних робіт.

Ієрархічна структура робіт являє собою, по суті, перелік завдань проекту. Вона може бути представлена в графічному вигляді або у вигляді опису, що відображає вкладення робіт. Ієрархічна структура робіт організовує і визначає весь зміст проекту. Роботи, не включені з WBS, не є роботами проекту.

Виконаємо побудову WBS структури, у якій зазначимо всі виконувані роботи в залежності від головних етапів:

Формування технічного завдання - розробка технічного завдання, що встановлює основне призначення, показники якості, техніко - економічні та спеціальні вимоги до розроблюваного інструментального засобу. Формування технічного завдання включає в себе підпункти:

– визначення предметної області;
– визначення мови написання;
– визначення цільової аудиторії;
– визначення вимог дизайну програмного продукту;
– визначення вимог засобів перегляду та вимог до системи управління контентом.

Планування проекту включає в себе розробку OBS структури, матриці відповідальності, календарного плану, а саме діаграми Ганта, управління ресурсами та ризиками.

Реалізація матиме 4 етапи:

Підготовка – збір потрібної інформації, формування можливостей та цілей проекту.

Проектування – розробка технічної моделі, створення ескізів основних об’єктів та ландшафту.

Розробка – створення програмного коду та графічної складової програми, тестування та оптимізація.

Реліз – випробування ПП, аналіз проблем та виправлення помилок.

І останній етап створення проекту завершення має на увазі здачу проекту в експлуатацію і закриття проекту.

Діаграму WBS наведено рис.А.1 та OBS на рис.А.2.

Рисунок А.1 – Структура WBS

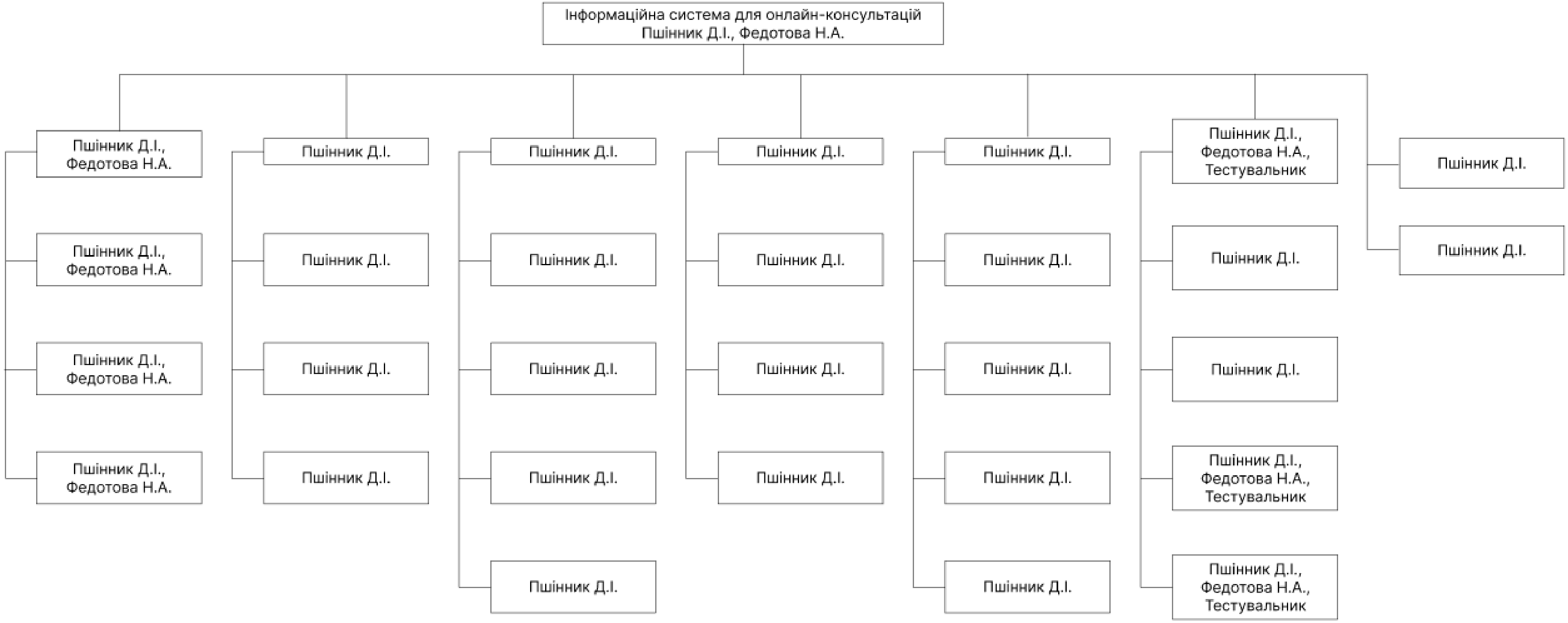


Рисунок А.2 – Структура OBS

А.3 ПОБУДОВА КАЛЕНДАРНОГО ГРАФІКУ ВИКОНАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

Для того щоб мати реальне уявлення про тривалість виконання робіт з урахуванням обмеженості у використанні ресурсів, на підставі часткової мережевої моделі будують календарний графік робіт.

Діаграма Ганта – горизонтальна лінійна діаграма, на якій задачі проекту представляються протяжними в часі відрізками, що характеризуються датами початку та закінчення, затримками і, можливо, іншими тимчасовими параметрами.

Кожен відрізок відповідає окремому завданню або підзадачі. Завдання і підзадачі, складові плану, розміщуються по вертикалі. Початок, кінець і довжина відрізка на шкалі часу відповідають початку, кінцю і тривалості завдання. На деяких діаграмах Ганта також показується залежність між завданнями.

На наступному рисунку представлено діаграму Ганта розроблюваного проекту. На рис.А.3 представлено побудовану діаграму Ганта.

Task	Start Date	Days to Complete
Аналітика	07.сент	23
Розробка дизайну	30.сент	46
Верстка сайту	15.нояб	13
База даних	28.нояб	7
Програмування	05.дек	20
Тестування	25.дек	21
Наповнення контентом	15.янв	15
Завантаження на сервер	30.янв	3

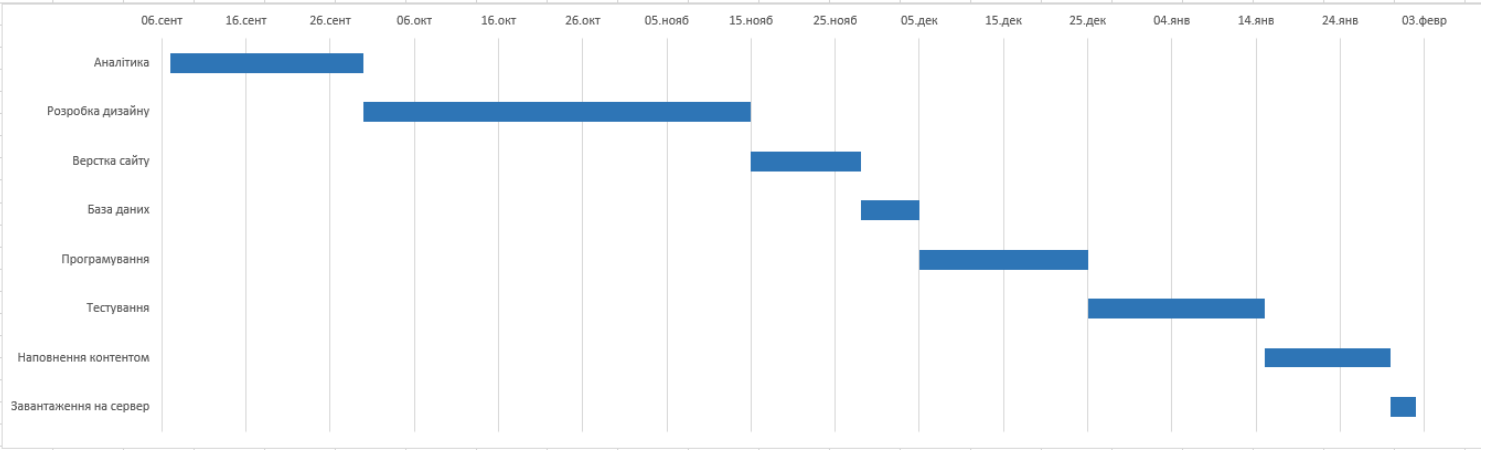


Рисунок А.3 – Діаграма Ганта

А.4 ПЛАНУВАННЯ РИЗИКІВ ПРОЕКТУ

Ризик – це ймовірнісна подія, яка у випадку своєї появи негативно або позитивно впливає на проект.

Управління ризиком – це процес реагування на події та зміни ризиків у процесі виконання проекту. При цьому важливим є проведення моніторингу ризиків.

Процес управління ризиками включає в себе такі пункти:

- Ідентифікація ризиків (виявлення ризиків);
- Оцінювання ризиків (оцінка ймовірності та впливу);
- Заходи реагування на ризики;
- Моніторинг ризиків.

Ідентифікація ризиків – це виявлення ризиків, здатних вплинути на проект, і документальне оформлення їх характеристик. Це ітеративний процес, який періодично повторюється на всьому протязі проекту, оскільки в рамках його життєвого циклу можуть виявлятися нові ризики. Найбільш розповсюдженою характеристикою ризику є загроза або небезпека виникнення невдач у тій чи іншій діяльності, небезпека виникнення несприятливих наслідків, змін зовнішнього середовища, які можуть викликати втрати ресурсів, збитки, а також небезпеку, від якої слід застрахуватися.

Планування реагування на ризики – це процес розробки шляхів і визначення дій із збільшення можливостей і зниження погроз для цілей проекту. Даний процес зачинається після проведення якісного і кількісного аналізу ризиків. В процесі аналізу для визначення числових значень ймовірності виникнення ступеня впливу, зазвичай застосовується метод експертних оцінок. На їх основі визначається ранг ризику, як потенційний вплив ризику на проект, який оцінюється як добуток ймовірності виникнення та ступеню впливу.

Ймовірність виникнення:

1. Слабоймовірно;
2. Малоймовірно;
3. Ймовірно;
4. Вельми ймовірно;
5. Майже можливо.

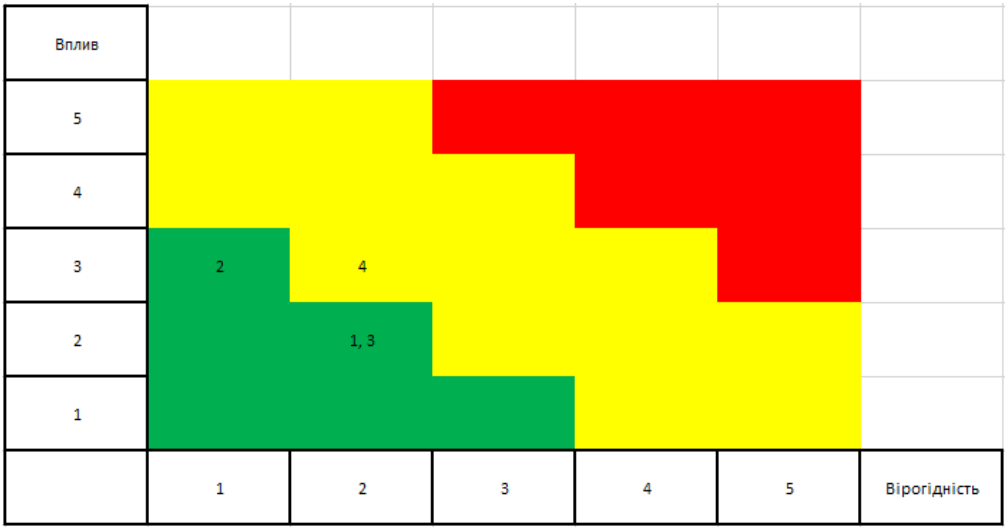
Величина втрат:

1. Мінімальна;
2. Низька;
3. Середня;
4. Висока;
5. Максимальна.

Таблиця А.4 – Ймовірність втрат

Ідентифікатор ризику	Ризики	Вірогідність	Вплив
1	Нечітка постановка задачі	2	2
2	Пошкодження даних	1	3
3	Некоректне тестування	2	2
4	Недотримання графіку робіт	2	3

Таблиця А.5 – Ймовірність втрат



Класифікація за ступенем впливу:

– ігноровані (1≤R≤4);
– незначні (5≤R≤8);
– помірні (9≤R≤11);
– вагомі (12≤R≤19);
– критичні (20≤R≤25).

Класифікація за рівнем ризику:

– прийнятні ризики;
– виправданні ризики;
– недопустимі ризики.

Таблиця А.6 – Класифікація за ступенем впливу та за рівнем ризику

Ризик		Ступінь впливу	Рівень ризику
Нечітка постановка задачі	R1	4	Ігноровані ризики
Пошкодження даних	R2	4	Ігноровані ризики
Некоректне тестування	R3	3	Ігноровані ризики
Недотримання графіку робіт	R4	6	Незначні ризики

План по усуненню ризиків:

- Чітке встановлення документації та строге її слідування.
- Резервне зберігання даних у хмарі.
- Моніторинг помилок через сторонні сервіси.
- Резервувати час на випадок помилок планування та виникнення непередбачених обставин.

ДОДАТОК Б. КОД РЕАЛІЗАЗІЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ

UserController.php

Контролер для роботи з користувачами.

```
<?php

namespace App\Http\Controllers;

use App\Models\User;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Hash;

use App\Http\Requests\UserRequest;

class UsersController extends Controller
{
    protected $publicStorage = '/uploads/';

    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index(Request $request)
    {
        $model = User::with('consultations.direction', 'role');

        if(isset($request->status)) {
            $model->where('status', 1);
        }

        if(isset($request->roles_id)) {
            $model->whereIn('roles_id', $request->roles_id);
        }

        if(isset($request->country) && $request->country != '') {
            $model->where('countries_id', $request->country);
        }

        if(isset($request->priceFrom) && $request->priceFrom != '') {
            $model->whereHas('consultations', function($query) use ($request) {
                $query->where('price', '<=', $request->priceFrom);
            });
        }

        if(isset($request->priceTo) && $request->priceTo != '') {
            $model->whereHas('consultations', function($query) use ($request) {
                $query->where('price', '>=', $request->priceTo);
            });
        }

        if(isset($request->free) && $request->free == 'true') {
            $model->whereHas('consultations', function($query) {
```

```
            $query->where('price', null);
        });
    }

    if(isset($request->directions)) {
        $model->whereHas('consultations', function($query) use ($request) {
            $query->whereIn('directions_id', $request->directions);
        });
    }

    $data = $model->get();
    return response()->json($data);
}

/**
 * Display the specified resource.
 *
 * @param  \App\Models\User  $user
 * @return \Illuminate\Http\Response
 */
public function show($id)
{
    $model = User::with('consultations.user', 'consultations.direction', 'consultations.review', 'country')->find($id);
    return response()->json($model);
}

/**
 * Update the specified resource in storage.
 *
 * @param  \Illuminate\Http\Request  $request
 * @param  \App\Models\User  $user
 * @return \Illuminate\Http\Response
 */
public function update(UserRequest $request)
{
    $user = User::find(Auth::id());
    $data = $request->validated();
    if(isset($request['photo']) && gettype($request['photo']) != 'string') {
        $puth = $this->publicStorage . Auth::id();
        $name =  $puth."/".uniqid().'.'.$request['photo']->getClientOriginalExtension();
        $request['photo']->move(public_path().$puth, $name);
        $data['photo'] = $name;
    } else {
        $data['photo'] = $user['photo'];
    }

    if($data['password'] != '') {
        $data["password"] = Hash::make($data['password']);
    } else {
        unset($data['password']);
    }

    $user->update($data);
    return response()->json(User::find(Auth::id()));
}

public function update2(Request $request, $id)
{
    $user = User::find($id);
    $data = $request->all();
```

```
        $user->update($data);
        return response('ok', 200);
    }
}
```

ConsultationsController.php

Контролер для роботи з консультаціями.

```
<?php

namespace App\Http\Controllers;

use App\Models\UserDirections;
use App\Models\Search;
use Illuminate\Http\Request;

class ConsultationsController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index(Request $request)
    {
        $model = UserDirections::with('user', 'direction', 'review')->where('status', 1);

        if(isset($request->title)) {
            $model->where('title', 'like', '%'.$request->title.'%');
            $search = new Search();
            $search->create([
                'text' => $request->title
            ]);
        }

        if(isset($request->directions)) {
            $model->whereIn('directions_id', $request->directions);
        }

        if(isset($request->priceFrom) && $request->priceFrom != '') {
            $model->where('price', '<=', $request->priceFrom);;
        }

        if(isset($request->priceTo) && $request->priceTo != '') {
            $model->where('price', '>=', $request->priceTo);
        }

        if(isset($request->free) && $request->free == 'true') {
            $model->where('price', null);
        }

        $data = $model->get();
        return response()->json($data);
    }

    /**
     * Display the specified resource.
     *
     * @param  \App\Models\UserDirections  $userDirections
     * @return \Illuminate\Http\Response
```

```
     */
    public function show($id)
    {
        $model = UserDirections::with('user', 'direction', 'meet.user', 'review.user')->find($id);
        return response()->json($model);
    }

    /**
     * Remove the specified resource from storage.
     *
     * @param  \App\Models\UserDirections  $userDirections
     * @return \Illuminate\Http\Response
     */
    public function destroy($id)
    {
        UserDirections::find($id)->delete();
        return response('ok', 200);
    }
}
```

MeetsController.php

Контролер для роботи з API Zoom.

```
<?php

namespace App\Http\Controllers;

use App\Models\Meets;
use Illuminate\Http\Request;
use App\Traits\ZoomMeetingTrait;
use Illuminate\Support\Facades\Auth;

class MeetsController extends Controller
{
    use ZoomMeetingTrait;
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index(Request $request)
    {
        $userId = Auth::id();
        $model = Meets::with('user', 'direction.user', 'status');

        if(Auth::user()->roles_id == 1) {
            $model->whereHas('direction', function($query) use ($userId) {
                $query->where('users_id', $userId);
            });
        }

        if(Auth::user()->roles_id == 2) {
            $model->where('users_id', Auth::id());
        }

        if(isset($request->status_id)) {
            $model->whereIn('status_id', $request->status_id);
        }

        $data = $model->get();
```

```
    return response()->json($data);
}

/**
 * Store a newly created resource in storage.
 *
 * @param  \Illuminate\Http\Request  $request
 * @return \Illuminate\Http\Response
 */
public function store(Request $request)
{
    $data = $request->all();
    $data['users_id'] = Auth::id();
    $data['status_id'] = 1;
    $model = new Meets();
    $model->create($data);
}

/**
 * Update the specified resource in storage.
 *
 * @param  \Illuminate\Http\Request  $request
 * @param  \App\Models\Meets  $meets
 * @return \Illuminate\Http\Response
 */
public function update(Request $request, $id)
{
    $data = $request->all();
    $model = Meets::with('direction')->find($id);
    if(!$model['direction']['price']) {
        $data['status_pay'] == 1;
    }
    $model->update($data);
    return response('ok', 200);
}

public function acceptMeet($id) {
    $model = Meets::with('direction')->find($id);
    $data = [
        'topic' => $model->direction->title,
        'start_time' => $model->time,
        'duration' => 1,
        'host_video' => 1,
        'participant_video' => 1
    ];
    $meet = $this->create($data);
    $model->update([
        'url' => $meet['data']['join_url'],
        'data' => $meet,
        'status_id' => 2
    ]);
    return response()->json($meet);
}

/**
 * Remove the specified resource from storage.
 *
 * @param  \App\Models\Meets  $meets
 * @return \Illuminate\Http\Response
 */
```

```
    public function destroy($id)
    {
        Meets::find($id)->delete();
        return response('ok', 200);
    }
}
```

ChatController.php

Контролер чатів. Працює з діалогами користувачів. Зберігає та створює нові чати між користувачами та зберігає повідомлення надіслані користувачами у відповідні чати.

Даний код надає можливість підключення до чату тільки авторизованим користувачам.

```
<?php

namespace App\Http\Controllers;

use App\Events\MessageSentEvent;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Carbon\Carbon;
use App\Models\Chats;
use App\Models\Messages;

class ChatController extends Controller
{
    function postChat(Request $request) {
        $model1 = Chats::where("account_id_1", $request->account_id)->where("account_id_2", Auth::id());
        $model2 = Chats::where("account_id_1", Auth::id())->where("account_id_2", $request->account_id);
        if($model1->exists() || $model2->exists()) {
            $message = new Messages();
            $id1 = $model1->first();
            $id2 = $model2->first();
            $message->create([
                "text" => $request->text,
                "account_id" => $request->account_id,
                "chat_id" => $model1->exists() ? $id1['id'] : $id2['id']
            ]);
        } else {
            $chat = new Chats();
            $response = $chat->create([
                "account_id_1" => Auth::id(),
                "account_id_2" => $request->account_id,
            ]);
            $message = new Messages();
            $message->create([
                "text" => $request->text,
                "account_id" => Auth::id(),
                "chat_id" => $response['id']
            ]);
        }
    }

    function getChats() {
```

```
        $chats = Chats::with('messages.user')->where("account_id_1", Auth::id())->orWhere("account_id_2", Auth::id())->get();
        foreach ($chats as $key => $chat) {
            $chat->date = Carbon::parse($chat->created_at)->format('d.m.Y H:i');
        }
        return response()->json($chats);
    }

    function postMessages(Request $request, $chat_id) {
        $user = Auth::user();

        $message = new Messages();
        $response = $message->create([
            "text" => $request->message,
            "account_id" => Auth::id(),
            "chat_id" => $chat_id
        ]);

        broadcast(new MessageSentEvent($response, $user, $chat_id))->toOthers();

        return response()->json([
            'message' => $response,
            'account' => $user
        ]);
    }

    function getMessages($chat_id) {
        $result = [];
        $data = Messages::with('user')->where('chat_id', $chat_id)->get();
        foreach ($data as $key => $chat) {
            array_push($result, [
                'message' => $chat,
                'account' => $chat['user']
            ]);
        }
        return response()->json($result);
    }
}
```

ZoomMeetingTrait.php

Трейт який розширює контроллер Zoom.

```
<?php

namespace App\Http\Controllers;

use App\Events\MessageSentEvent;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Carbon\Carbon;
use App\Models\Chats;
use App\Models\Messages;

class ChatController extends Controller
{
    function postChat(Request $request) {
        $model1 = Chats::where("account_id_1", $request->account_id)->where("account_id_2", Auth::id());
        $model2 = Chats::where("account_id_1", Auth::id())->where("account_id_2", $request->account_id);
        if($model1->exists() || $model2->exists()) {
```

```
            $message = new Messages();
            $id1 = $model1->first();
            $id2 = $model2->first();
            $message->create([
                "text" => $request->text,
                "account_id" => $request->account_id,
                "chat_id" => $model1->exists() ? $id1['id'] : $id2['id']
            ]);
        } else {
            $chat = new Chats();
            $response = $chat->create([
                "account_id_1" => Auth::id(),
                "account_id_2" => $request->account_id,
            ]);
            $message = new Messages();
            $message->create([
                "text" => $request->text,
                "account_id" => Auth::id(),
                "chat_id" => $response['id']
            ]);
        }
    }
    function getChats() {
        $chats = Chats::with('messages.user')->where("account_id_1", Auth::id())->orWhere("account_id_2", Auth::id())-
>get();
        foreach ($chats as $key => $chat) {
            $chat->date = Carbon::parse($chat->created_at)->format('d.m.Y H:i');
        }
        return response()->json($chats);
    }

    function postMessages(Request $request, $chat_id) {
        $user = Auth::user();

        $message = new Messages();
        $response = $message->create([
            "text" => $request->message,
            "account_id" => Auth::id(),
            "chat_id" => $chat_id
        ]);
        broadcast(new MessageSentEvent($response, $user, $chat_id))->toOthers();

        return response()->json([
            'message' => $response,
            'account' => $user
        ]);
    }

    function getMessages($chat_id) {
        $result = [];
        $data = Messages::with('user')->where('chat_id', $chat_id)->get();
        foreach ($data as $key => $chat) {
            array_push($result, [
                'message' => $chat,
                'account' => $chat['user']
            ]);
        }
        return response()->json($result);
    }
}
```

AuthController.php

Контролер авторизації користувачів сайту. Містить методи авторизації, реєстрації, завершення сесії та перевірки статусу авторизованого користувача. У випадку якщо сесія закінчилась відбувається переадресація на сторінку авторизації.

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;

use App\Models\User;
use Illuminate\Support\Facades\Auth;
use Cookie;

class AuthController extends Controller
{
  function checkUser()
  {
    $model = User::find(Auth::id());
    return response()->json($model);
  }

  function checkAdmin()
  {
    $model = User::where('roles_id', 3)->find(Auth::id());
    return response()->json($model);
  }

  // register
  function register(Request $request)
  {
    $request->validate([
      'email' => 'required|string|email|unique:users',
      'password' => 'required|string'
    ]);
    $user = new User();
    $data = $request->all();
    $data["password"] = Hash::make($request->password);
    $user->create($data);
    $credentials = request(['email', 'password']);
    if (!Auth::attempt($credentials)) {
      return response()->json(['message' => 'Unauthorized'], 401);
    }
    $authUser = Auth::user();
    $tokenResult = $authUser->createToken('Personal Access Token');
    $token = $tokenResult->token;
    $token->save();
    return response()->json([
      'access_token' => 'Bearer ' . $tokenResult->accessToken,
      'user' => $authUser
    ]);
  }
```

```
  function login(Request $request)
  {
    $request->validate([
      'email' => 'required|string|email',
      'password' => 'required|string',
    ]);
    $credentials = request(['email', 'password']);
    if (Auth::attempt($credentials)) {
      $user = Auth::user();
      $token =  $user->createToken('Personal Access Token')->accessToken;
      $cookie = $this->getCookieDetails($token);
      return response()->json([
        'user' => $user,
        'token' => $token,
      ], 200)
        ->cookie($cookie['name'], $cookie['value'], $cookie['minutes'], $cookie['path'], $cookie['domain'], $cookie['secure'],
$cookie['httponly'], $cookie['samesite']);
    } else {
      return response()->json([
        'error' => 'Помилка авторизації. Некоректний email, або пароль.'
      ], 422);
    }
  }

  private function getCookieDetails($token)
  {
    return [
      'name' => '_token',
      'value' => $token,
      'minutes' => 1440,
      'path' => null,
      'domain' => null,
      'secure' => null,
      'httponly' => true,
      'samesite' => true,
    ];
  }

  function logout(Request $request)
  {
    $request->user()->token()->revoke();
    $cookie = Cookie::forget('_token');
    return response('ok', 200)->withCookie($cookie);
  }
}
```

App.jsx

Головний компонент який підключає всі сторінки сайту.

```
import React from "react";

import Header from '../components/Header';
import Footer from '../components/Footer';

import {
    BrowserRouter,
    Routes,
    Route,
  } from "react-router-dom";
```

```
import Home from './Home/Index';
import News from './News';
import NewsItem from './NewsItem';
import Instructors from './Instructors';
import Instructor from './Instructor';
import Consultations from './Consultations';
import Consultation from './Consultation';
import About from './About';
import {Login} from './Login';
import {Register} from './Register';
import {Cabinet} from './Cabinet/Index';

const App = () => {
   return (
      <>
      <BrowserRouter>
         <Header />
         <main className="main">
            <Routes>
               <Route path="/" element={<Home />} />
               <Route path="/news" element={<News />} />
               <Route path="/news/:id" element={<NewsItem />} />
               <Route path="/instructors" element={<Instructors />} />
               <Route path="/instructor/:id" element={<Instructor />} />
               <Route path="/consultations" element={<Consultations />} />
               <Route path="/consultation/:id" element={<Consultation />} />
               <Route path="/about" element={<About />} />
               <Route path="/login" element={<Login />} />
               <Route path="/register" element={<Register />} />
               <Route path="/cabinet/*" element={<Cabinet />} />
            </Routes>
         </main>
         <Footer />
      </BrowserRouter>
      </>
   )
}

export default App
```

Consultations.jsx

Компонент консультацій.

```
import React, { Component } from 'react'
import API from '../Network'
import styled from 'styled-components'
import { useParams } from 'react-router-dom'
import Container from 'react-bootstrap/Container'
import Row from 'react-bootstrap/Row'
import Col from 'react-bootstrap/Col'

import InnerBanner from '../components/InnerBanner'
import Pagination from '../components/Pagination'

import ConsultationItem from '../components/ConsultationItem'

const Styles = styled.div`
  padding: 0 0 70px 0;
  .courses-page-sidebar {
```

```
.courses-page-side-bar-widget {
  margin-bottom: 30px;
  padding: 25px;
  border: 1px solid #f5f5f5;
  .title {
    font-size: 22px;
    margin-bottom: 20px;
    font-weight: 700;
    color: var(--titleColor);
    line-height: 1.4;
  }
  .courses-page-content {
    list-style-type: none;
    position: relative;
    margin: 0;
    padding: 0;
    li {
      display: block;
      margin-bottom: 10px;
      input[type='checkbox'] {
        display: none;
        &:checked + label:before {
          content: '';
          background: var(--mainColor) url('/img/check.png') no-repeat;
          background-position: center center;
          background-size: 80%;
          border: 1px solid var(--mainColor);
          font-family: 'remixicon' !important;
        }
      }
      label {
        display: inline-block;
        cursor: pointer;
        position: relative;
        padding-left: 28px;
        margin-bottom: 0;
        font-size: 16px;
        line-height: 20px;
        color: var(--bodyColor);
        width: 100%;
        &:before {
          content: '';
          display: inline-block;
          width: 18px;
          height: 18px;
          margin-right: 10px;
          position: absolute;
          left: 0;
          top: 2px;
          background-color: #fff;
          border: 1px solid #ecedf2;
          border-radius: 10px;
          -webkit-box-sizing: border-box;
          box-sizing: border-box;
          -webkit-transition: all 0.25s;
          transition: all 0.25s;
          font-family: 'remixicon' !important;
          color: #fff;
          font-size: 9px;
          font-weight: 900;
```

```
            text-align: center;
            line-height: 17px;
          }
          span {
            float: right;
            text-align: center;
            display: inline-block;
            width: 24px;
            line-height: 24px;
            height: 24px;
            background-color: #dff3ef;
            color: var(--mainColor);
            font-weight: 600;
            -webkit-transition: var(--transition);
            transition: var(--transition);
            font-size: 14px;
          }
        }
      }
    }
  }
  .form-group {
    margin-bottom: 20px;
    position: relative;
    .form-control {
      background-color: #fff;
      height: 50px;
      color: var(--bodyColor);
      border: 1px solid #cecece;
      width: 100%;
      border-radius: 0;
    }
  }
`

function withParams(Component) {
  return (props) => <Component {...props} params={useParams()} />
}

class Consultations extends React.Component {
  constructor(props) {
    super(props)
    this.state = {
      consultations: [],
      directions: [],
      user: null,
      filter: {
        directions: [],
        title: '',
        priceFrom: '',
        priceTo: '',
        free: false,
      },
    }
  }

  componentWillMount() {
    this.setParam()
    this.getConsultations()
```

```
  this.getDirections()
  this.getAuthUser()
}

getConsultations() {
  console.log(this.state.filter)

  API.get(`${this.state.user ? '' : 'guest'}/consultations`, {
    params: this.state.filter,
  }).then((res) => {
    const consultations = res.data
    this.setState({ consultations })
  })
}

setParam() {
  const queryString = window.location.search
  const urlParams = new URLSearchParams(queryString)
  this.state.filter.title = urlParams.get('search')
  if(urlParams.get('directions')) {
    this.state.filter.directions.push(urlParams.get('directions'))
  }
}

getDirections() {
  API.get(`/directions`).then((res) => {
    const directions = res.data
    this.setState({ directions })
  })
}

getAuthUser() {
  API.get(`/auth-user`)
  .then(res => {
      this.setState({ user: res.data });
  })
}

handleCheck = (event) => {
  var directions = [...this.state.filter.directions]
  if (directions.indexOf(event.target.value) < 0) {
    directions.push(event.target.value)
  } else {
    directions.splice(directions.indexOf(event.target.value), 1)
  }
  this.state.filter.directions = directions
  this.setState({ filter: this.state.filter })
  console.log(this.state.filter)
}

onChangehandler = (e) => {
  this.state.filter[e.target.name] = e.target.value
  this.setState({ filter: this.state.filter })
  console.log(this.state.filter)
}

onChangehandlerChechBox = (e) => {
  this.state.filter.free = !this.state.filter.free
  this.setState({ filter: this.state.filter })
}
```

```
render() {
  function ConsultationsItems(props) {
    const items = props.items
    const listItems = items.map((item) => (
      <Col md={6} key={item.id}>
        <ConsultationItem item={item} />
      </Col>
    ))
    return <Row>{listItems}</Row>
  }
  return (
    <>
      <Styles>
        <InnerBanner title="Послуги" />
        <Container>
          <Row>
            <Col md={3}>
              <div className="courses-page-sidebar">
                <div className="courses-page-side-bar-widget">
                  <h3 className="title">Категорія</h3>
                  <ul className="courses-page-content">
                    {this.state.directions.map((item) => (
                      <li key={item.id}>
                        <input
                          id={`check-f-${item.id}`}
                          type="checkbox"
                          name="check"
                          value={item.id}
                          onChange={this.handleCheck}
                        />
                        <label htmlFor={`check-f-${item.id}`}>
                          {item.title}{' '}
                          <span className="fl">{item.select_count}</span>
                        </label>
                      </li>
                    ))}
                  </ul>
                </div>
                <div className="courses-page-side-bar-widget">
                  <h3 className="title">Ціна</h3>
                  <Row>
                    <Col>
                      <div className="form-group">
                        <input
                          type="text"
                          className="form-control"
                          placeholder="Від"
                          name="priceFrom"
                          value={this.state.filter.priceFrom}
                          onChange={this.onChangehandler}
                        />
                      </div>
                    </Col>
                    <Col>
                      <div className="form-group">
                        <input
                          type="text"
                          className="form-control"
                          placeholder="До"
```

```
                    name="priceTo"
                    value={this.state.filter.priceTo}
                    onChange={this.onChangehandler}
                   />
                  </div>
                 </Col>
                </Row>
                <ul className="courses-page-content">
                 <li>
                  <input
                   id={`check-free`}
                   type="checkbox"
                   name="free"
                   onChange={this.onChangehandlerChechBox}
                  />
                  <label htmlFor={`check-free`}>Безкоштовно</label>
                 </li>
                </ul>
               </div>
               <button
                type="submit"
                onClick={(e) => this.getConsultations()}
                className="default-btn block"
               >
                Пошук
               </button>
               <br />
              </div>
             </Col>
             <Col md={9}>
              <ConsultationsItems items={this.state.consultations} />
             </Col>
            </Row>
            <Pagination />
           </Container>
          </Styles>
         </>
        )
       }
}

export default withParams(Consultations)
```

Instructors.jsx

Компонент фахівців.

```
import React from 'react'
import styled from 'styled-components'

import API from '../Network'

import Container from 'react-bootstrap/Container'
import Row from 'react-bootstrap/Row'
import Col from 'react-bootstrap/Col'
import Form from 'react-bootstrap/Form'
import { Plus, Facebook, Instagram } from 'react-bootstrap-icons'
import { Link } from 'react-router-dom'

import InnerBanner from '../components/InnerBanner'
```

```
import Pagination from '../components/Pagination'

const Styles = styled.div`
  padding: 0 0 70px 0;
  .instructors-card {
    margin-bottom: 30px;
    background-color: var(--whiteColor);
    -webkit-box-shadow: 0 0 15px rgb(0 0 0 / 6%);
    box-shadow: 0 0 15px rgb(0 0 0 / 6%);
    a {
      color: var(--titleColor);
    }
    img {
      width: 100%;
      height: 306px;
      object-fit: cover;
    }
    .content {
      padding: 30px 50px 30px 10px;
      position: relative;
    }
    h3 {
      margin-bottom: 5px;
      font-size: 22px;
      font-weight: 700;
      color: var(--titleColor);
      line-height: 1.4;
      font-family: var(--headerFonts);
    }
    span {
      color: var(--bodyColor);
    }
    ul.instructors-social {
      margin: 0;
      padding: 0;
      position: absolute;
      right: 10px;
      li {
        list-style: none;
        width: 45px;
        height: 45px;
        border-radius: 50px;
        background-color: rgba(8, 169, 230, 0.1);
        color: var(--mainColor);
        display: inline-flex;
        margin: 0;
        font-size: 24px;
        -webkit-transition: 0.9s;
        transition: 0.9s;
        align-items: center;
        justify-content: center;
        cursor: pointer;
        &:hover {
          background-color: var(--mainColor);
          color: var(--whiteColor);
        }
      }
      ul.dropdown {
        display: none;
        margin: 0;
```

```
      padding: 0;
      position: absolute;
      bottom: 45px;
      li {
        margin-bottom: 15px;
        display: inline-block;
        width: 40px;
        height: 40px;
        background-color: var(--whiteColor);
        color: var(--whiteColor);
        font-size: 18px;
        -webkit-transition: var(--transition);
        transition: var(--transition);
        border-radius: 50px;
        -webkit-box-shadow: 0 0 15px rgb(0 0 0 / 6%);
        box-shadow: 0 0 15px rgb(0 0 0 / 6%);
        display: flex;
        align-items: center;
        justify-content: center;
      }
    }
    li:hover ul.dropdown {
      display: block;
    }
  }
}
.courses-page-sidebar {
  .courses-page-side-bar-widget {
    margin-bottom: 30px;
    padding: 25px;
    border: 1px solid #f5f5f5;
    .title {
      font-size: 22px;
      margin-bottom: 20px;
      font-weight: 700;
      color: var(--titleColor);
      line-height: 1.4;
    }
    .courses-page-content {
      list-style-type: none;
      position: relative;
      margin: 0;
      padding: 0;
      li {
        display: block;
        margin-bottom: 10px;
        input[type='checkbox'] {
          display: none;
          &:checked + label:before {
            content: '';
            background: var(--mainColor) url('/img/check.png') no-repeat;
            background-position: center center;
            background-size: 80%;
            border: 1px solid var(--mainColor);
            font-family: 'remixicon' !important;
          }
        }
        label {
          display: inline-block;
          cursor: pointer;
```

```
        position: relative;
        padding-left: 28px;
        margin-bottom: 0;
        font-size: 16px;
        line-height: 20px;
        color: var(--bodyColor);
        width: 100%;
        &:before {
          content: '';
          display: inline-block;
          width: 18px;
          height: 18px;
          margin-right: 10px;
          position: absolute;
          left: 0;
          top: 2px;
          background-color: #fff;
          border: 1px solid #ecedf2;
          border-radius: 10px;
          -webkit-box-sizing: border-box;
          box-sizing: border-box;
          -webkit-transition: all 0.25s;
          transition: all 0.25s;
          font-family: 'remixicon' !important;
          color: #fff;
          font-size: 9px;
          font-weight: 900;
          text-align: center;
          line-height: 17px;
        }
        span {
          float: right;
          text-align: center;
          display: inline-block;
          width: 24px;
          line-height: 24px;
          height: 24px;
          background-color: #dff3ef;
          color: var(--mainColor);
          font-weight: 600;
          -webkit-transition: var(--transition);
          transition: var(--transition);
          font-size: 14px;
        }
      }
    }
  }
}
.form-group {
  margin-bottom: 20px;
  position: relative;
  .form-control {
    background-color: #fff;
    height: 50px;
    color: var(--bodyColor);
    border: 1px solid #cecece;
    width: 100%;
    border-radius: 0;
  }
```

```
  }
`

class Instructors extends React.Component {
  constructor(props) {
    super(props)
    this.state = {
      directions: [],
      countries: [],
      users: [],
      filter: {
        directions: [],
        country: "",
        priceFrom: '',
        priceTo: '',
        free: false,
        roles_id: [1],
        status: 1
      },
    }
  }

  componentDidMount() {
    this.getUsers()
    this.getDirections()
    this.getCountries()
  }

  getUsers() {
    API.get(`/users`, {
      params: this.state.filter,
    }).then((res) => {
      const users = res.data
      this.setState({ users })
    })
  }

  getDirections() {
    API.get(`/directions`).then((res) => {
      const directions = res.data
      this.setState({ directions })
    })
  }

  getCountries() {
    API.get(`/countrues`).then((res) => {
      const countries = res.data
      this.setState({ countries })
    })
  }

  handleCheck = (event) => {
    var directions = [...this.state.filter.directions]
    if (directions.indexOf(event.target.value) < 0) {
      directions.push(event.target.value)
    } else {
      directions.splice(directions.indexOf(event.target.value), 1)
    }
    this.state.filter.directions = directions
    this.setState({ filter: this.state.filter })
```

```
}

onChangehandler = (e) => {
  this.state.filter[e.target.name] = e.target.value
  this.setState({ filter: this.state.filter })
  console.log(this.state.filter)
}

onChangehandlerChechBox = (e) => {
  this.state.filter.free = !this.state.filter.free
  this.setState({ filter: this.state.filter })
}

render() {
  function InstructorsItems(props) {
    const items = props.items
    const listItems = items.map((item) => (
      <Col md={4} key={item.id}>
        <div className="instructors-card">
          <Link to={`/instructor/${item.id}`}>
            <img src={item.photo} alt="" />
          </Link>
          <div className="content">
            <ul className="instructors-social">
              <li className="share-btn">
                <Plus />
                <ul className="dropdown">
                  {item.facebook != 'null' ? (
                    <li>
                      <a href={item.facebook} target="_blank">
                        <Facebook />
                      </a>
                    </li>
                  ) : (
                    ''
                  )}
                  {item.instagram != 'null' ? (
                    <li>
                      <a href={item.instagram} target="_blank">
                        <Instagram />
                      </a>
                    </li>
                  ) : (
                    ''
                  )}
                </ul>
              </li>
            </ul>
            <h3>
              <Link to={`/instructor/${item.id}`}>
                {item.name} {item.surname}
              </Link>
            </h3>
            {item.consultations.map((item) => (
              <span key={item.id}>{item.direction.title}</span>
            ))}
          </div>
        </div>
      </Col>
    ))
```

```
 return <Row>{listItems}</Row>
}
return (
 <>
  <Styles>
   <InnerBanner title="Фахівці" />
   <Container>
    <Row>
     <Col md={3}>
      <div className="courses-page-sidebar">
       <div className="courses-page-side-bar-widget">
        <h3 className="title">Категорія</h3>
        <ul className="courses-page-content">
         {this.state.directions.map((item) => (
          <li key={item.id}>
           <input
            id={`check-f-${item.id}`}
            type="checkbox"
            name="check"
            value={item.id}
            onChange={this.handleCheck}
           />
           <label htmlFor={`check-f-${item.id}`}>
            {item.title}{' '}
            <span className="fl">{item.select_count}</span>
           </label>
          </li>
         ))}
        </ul>
       </div>
       <div className="courses-page-side-bar-widget">
        <h3 className="title">Країна</h3>
        <div className="form-group">
         <select
          className="form-control"
          name="country"
          value={this.state.filter.country}
          onChange={this.onChangehandler}
         >
          <option value="default"></option>
          {this.state.countries.map((item) => (
           <option value={item.id} key={item.id}>
            {item.name}
           </option>
          ))}
         </select>
        </div>
       </div>
       <button type="submit" onClick={e => this.getUsers()} className="default-btn block">
        Пошук
       </button>
       <br />
      </div>
     </Col>
     <Col md={9}>
      <InstructorsItems items={this.state.users} />
     </Col>
    </Row>
    <Pagination />
   </Container>
```

```
        </Styles>
      </>
    )
  }
}

export default Instructors
```