

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавр

(ступінь вищої освіти)

на тему: ПРОГРАМНА РЕАЛІЗАЦІЯ ЧАТБОТА ДЛЯ ЦИФРОВОЇ
ПЛАТФОРМИ ЗЕЛЕНОГО УНІВЕРСИТЕТУ
SOFTWARE IMPLEMENTATION OF A CHATBOT FOR THE
DIGITAL PLATFORM OF THE GREEN UNIVERSITY

Виконав(ла): студент(ка) 4 курсу, групи КНТ-131

Спеціальності 121 Інженерія програмного

(код і найменування спеціальності)

забезпечення

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

САЛІЙ Д.В.

(ПРІЗВИЩЕ та ініціали)

Керівник ПАРХОМЕНКО А.В.

(ПРІЗВИЩЕ та ініціали)

Рецензент ЗЕЛЕНЬОВА І.Я.

(ПРІЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування закладу вищої освіти)

Факультет КНТ
 Кафедра програмних засобів
 Ступінь вищої освіти бакалавр
 Спеціальність 121 Інженерія програмного забезпечення
 (код і найменування)
 Освітня програма (спеціалізація) Інженерія програмного забезпечення
 (назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
 “ ” 2025 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

САЛІЯ Данііла Володимировича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Програмна реалізація чатбота для цифрової платформи зеленого університету. Software Implementation of a Chatbot for the Digital Platform of the Green University
 керівник проєкту (роботи) к.т.н., доцент, ПАРХОМЕНКО Анжеліка Володимирівна

(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “ 28 ” квітня 2025 року № 209

2. Строк подання студентом проєкту (роботи) 02 червня 2025 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Проєктування програмного забезпечення. 3. Розробка програми. 4. Експлуатація, тестування та експериментальне дослідження програми

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)
Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	ПАРХОМЕНКО А.В., доцент		
Нормоконтроль	БЄЛОВА А.В., асистент		

7. Дата видачі завдання « 14 » квітня 2024 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка архітектури програми.	2 тиждень	Розділ 2
5	Розробка програми.	3-4 тижні	Розділ 3
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	ПЗ, Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)

_____ Данііл САЛІЙ
(підпис) (Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

_____ Анжеліка ПАРХОМЕНКО
(підпис) (Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 162 с., 25 табл., 20 рис., 5 дод., 59 джерел.

ВЕБЗАСТОСУНОК, ЧАТБОТ, ШТУЧНИЙ ІНТЕЛЕКТ, ЕКОЛОГІЯ, ЗЕЛЕНИЙ УНІВЕРСИТЕТ.

Об'єкт дослідження — інформаційна система зеленого університету.

Предмет дослідження — програмні засоби, орієнтовані на екологічну тематику та сталий розвиток університету.

Мета роботи — розробка та впровадження чатбота, який забезпечує зручний доступ до екологічної інформації та сприяє формуванню позитивного іміджу університету як активного учасника зелених ініціатив.

Матеріали, методи та технічні засоби: архітектурний шаблон MVC, мови HTML, SCSS, JavaScript, фреймворки React і Laravel, Blade-шаблони.

Результати. В результаті виконання роботи створено вебзастосунок чатбота з функціональністю пошуку за ключовими словами й тегами, розділами FAQ та академічних ресурсів. Забезпечено функціонал для адміністрування вмісту через зручну адмінпанель. Система динамічно масштабується, надає користувачам швидкий доступ до стислої екологічної інформації та демонструє активність університету в напрямку сталого розвитку.

Висновки. Розроблений чатбот є ефективним інструментом поширення екологічної інформації серед студентів та викладачів. Він сприятиме підвищенню інформованості користувачів щодо ініціатив сталого розвитку та допоможе формувати імідж зеленого університету.

Галузь використання — вебпортали вищих навчальних закладів, зацікавлених у просуванні екологічної політики, сталого розвитку та цифрової комунікації з користувачами.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 162 p., 18 tables, 25 figures, 5 appendices, 59 sources.

WEB APPLICATION, CHATBOT, ARTIFICIAL INTELLIGENCE,
ECOLOGY, GREEN UNIVERSITY.

Object of study is the information system of a green university.

Subject of study are software tools focused on environmental topics and sustainable university development.

The purpose of the work is to develop and implement a chatbot that provides convenient access to environmental information and contributes to forming a positive image of the university as an active participant in green initiatives.

Materials, methods, and tools: MVC, languages HTML, SCSS, JavaScript, React and Laravel frameworks, Blade templates.

Results. As a result of the work, a chatbot web application was developed, featuring keyword and tag-based search, as well as structured sections for FAQs and academic resources. An intuitive admin panel was implemented for content management. The system is dynamically scalable, provides users with quick access to concise environmental information, and showcases the university's engagement in sustainability efforts.

Conclusions. The developed chatbot is an effective tool for disseminating environmental information among students and faculty. It will contribute to increasing awareness of sustainability initiatives and will help shape the image of the university as a green institution.

Field of application – university web portals interested in promoting environmental policies, sustainable development, and digital communication with users.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	7
Вступ.....	8
1 Аналіз предметної області.....	9
1.1 Глобальні ініціативи в сфері зелених технологій.....	9
1.2 Інформування про екологічні досягнення університетів.....	11
1.3 Формування зеленого іміджу через інформаційні акценти.....	18
1.4 Роль ІІІ у проєктах зелених університетів	23
1.5 Постановка задач дипломної роботи	29
2 Проєктування програмного забезпечення	31
2.1 Аналіз вимог	31
2.2 Архітектура програмного забезпечення	32
2.3 Обґрунтування вибору технологій	36
2.4 Висновок до розділу 2	40
3 Розробка програми	42
3.1 Розробка серверної частини.....	42
3.2 Розробка користувацького інтерфейсу	52
3.3 Розробка адмінпанелі.....	59
4 Експлуатація, тестування та експериментальне дослідження програми	64
4.1 Тестування АРІ до бекенду	64
4.2 Тестування користувацького інтерфейсу	69
4.3 Тестування адмін панелі.....	71
4.4 Оцінка тестування	74
Висновки	76
Перелік джерел посилань	78
Додаток А Технічне завдання	84
Додаток Б Інструкція користувача	94
Додаток В Керівництво адміністратора.....	98
Додаток Д Текст програми.....	101
Додаток Е Слайди презентації	157

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

CRUD – Create, Read, Update, Delete;

БД – база даних;

НУЗП – Національний університет «Запорізька політехніка»;

СКБД – система керування базами даних;

ТЗ – технічне завдання;

ШІ – штучний інтелект.

ВСТУП

У сучасному цифровому середовищі університети, які прагнуть до статусу "зелених", повинні не лише впроваджувати екологічні ініціативи, а й ефективно комунікувати про них зі своєю аудиторією.

Національний університет "Запорізька політехніка" (НУЗП), як учасник глобального руху сталого розвитку, має потребу в сучасних ІТ-рішеннях для відкритої та зручної подачі інформації про свою екологічну діяльність. Аналіз вебпорталів провідних університетів, що займають високі позиції у рейтингу UI GreenMetric, показав, що використання чатботів на їхніх сайтах практично не зустрічається, хоча потенціал таких рішень для залучення цільової аудиторії є значним.

У зв'язку з цим виникає необхідність дослідження можливості інтеграції вебмодуля, який забезпечить інтерактивну взаємодію користувачів з екологічним контентом порталу університету.

Метою роботи є розробка та впровадження чатбота, який забезпечує зручний доступ до екологічної інформації та допомагає формувати позитивний імідж університету як активного учасника зелених ініціатив.

Для досягнення поставленої мети також потрібно дослідити альтернативні засоби комунікування з аудиторією та можливості використання штучного інтелекту (ШІ) на вебпорталі університету.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Глобальні ініціативи в сфері зелених технологій

1.1.1 Що таке зелений університет за версією GreenMetric

GreenMetric – це міжнародний рейтинг, який оцінює зусилля університетів у напрямку сталого розвитку та екологічної стійкості [3]. Згідно з GreenMetric, зелений університет характеризується низкою ключових аспектів, які відображають його відповідальність перед навколишнім середовищем:

- управління ресурсами: Зелений університет має розроблену стратегію управління природними ресурсами, яка передбачає зменшення споживання енергії та води, а також ефективне управління відходами. Це охоплює впровадження екологічних стандартів у щоденну діяльність і залучення студентів та персоналу до екологічних ініціатив [3];

- екологічні ініціативи: Зелені університети реалізують проекти, спрямовані на поліпшення екологічної ситуації в регіоні, такі як програми з посадки дерев, очищення водойм, використання відновлювальних джерел енергії та зменшення викидів CO₂. Вони також активно працюють над освітою та підвищенням обізнаності суспільства щодо екологічних проблем [3];

- впровадження сталих технологій: Зелений університет прагне інтегрувати інноваційні екологічні технології у всі свої процеси. Це може включати використання систем енергоощадження, екологічного транспорту, смарттехнологій для моніторингу та управління ресурсами, а також дослідження у сфері біотехнологій [3].

Ці характеристики роблять зелені університети важливими учасниками глобальних зусиль у боротьбі зі змінами клімату та забезпеченні сталого розвитку [3].

1.1.2 Міжнародні програми в сфері зелених технологій

У світі існує безліч міжнародних ініціатив, що сприяють розвитку зелених технологій та підтримують зусилля зелених університетів. Наприклад, програма ООН "2030 Agenda for Sustainable Development" спрямована на досягнення 17 цілей сталого розвитку, серед яких особливу увагу приділено охороні довкілля та зменшенню бідності [4]. Ці цілі підкреслюють необхідність інтеграції екологічних аспектів у всі сфери життя.

Європейський Союз також реалізує ряд ініціатив, таких як Green Deal, метою яких є зменшення викидів парникових газів і переведення Європи на безвідходну економіку [5]-[6]. У рамках цієї ініціативи університети відіграють важливу роль, отримуючи фінансування на дослідження і впровадження нових технологій [7].

Україна також бере участь у глобальних екологічних ініціативах, зокрема через проєкт NAWA "Green Transition in Ukrainian Universities", який підтримує інтеграцію принципів сталого розвитку в університетах. Ця програма спрямована на розвиток екологічних компетенцій серед викладачів і студентів та впровадження зелених технологій в освітню діяльність [8].

Крім того, різні національні програми підтримують зелені ініціативи на місцевому рівні. Наприклад, у США існує програма EPA (Environmental Protection Agency), яка фінансує проєкти, спрямовані на покращення екологічної ситуації в громадах [9].

Ці глобальні та регіональні ініціативи не лише підтримують зелені університети, а й стимулюють їх до впровадження інновацій та співпраці з іншими організаціями, що дозволяє обмінюватися досвідом та найкращими практиками у сфері сталого розвитку.

1.2 Інформування про екологічні досягнення університетів

Сучасні університети прагнуть покращити свої позиції у світових екологічних рейтингах, таких як GreenMetric, тому активно використовують свої онлайн-портали для висвітлення екологічних ініціатив. Однак способи подання цієї інформації суттєво відрізняються. Деякі університети публікують докладні звіти про впроваджені проєкти, тоді як інші використовують інтерактивні платформи, інфографіку та мультимедійний контент, щоб зробити інформацію більш доступною для широкої аудиторії [10].

Важливо оцінити, як університети представляють інформацію про свої екологічні ініціативи та які методи вони використовують для поширення цієї інформації. Дослідження допоможе визначити основні підходи та оцінити їхню актуальність.

Насамперед варто зазначити спільну рису більшості університетів світу – наявність власних репозиторіїв наукових робіт. У таких сховищах містяться детальні описи проведених досліджень, зареєстровані патенти та розробки, зокрема й ті, що стосуються екологічних ініціатив. Зазвичай ця інформація зберігається у вигляді електронних рефератів, переважно у форматі .pdf. Хоча такий спосіб збереження забезпечує повноту даних про наукову діяльність інституту, пошук необхідної інформації серед великої кількості документів може бути доволі тривалим і незручним [10]. Наприклад, на рис. 1.1 та 1.2 представлено приклади структури репозиторію наукових робіт у Національному лісотехнічному університеті України, що містить екологічні дослідження та інші академічні матеріали.

З метою спрощення доступу до інформації деякі університети використовують власні онлайн-платформи для публікації коротких новин, що висвітлюють їхні досягнення у різних сферах. Такий підхід дозволяє уникнути перевантаження деталями та забезпечує зручність сприйняття. Вдалими прикладами можна назвати новинну стрічку Wageningen University & Research

та Національного університету "Львівська політехніка". Формати представлення та поширення таких новин наведені на рис. 1.3 та 1.4 [13]-[14].

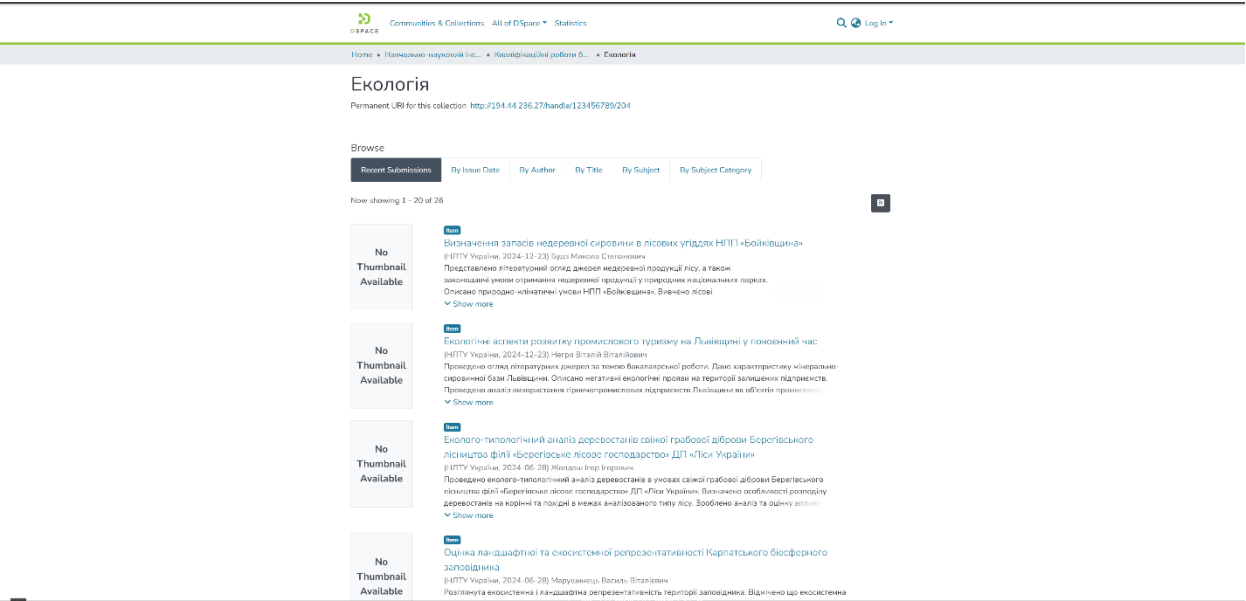


Рисунок 1.1 – Сторінка репозиторію НЛТУ на тему “Екологія” [11]

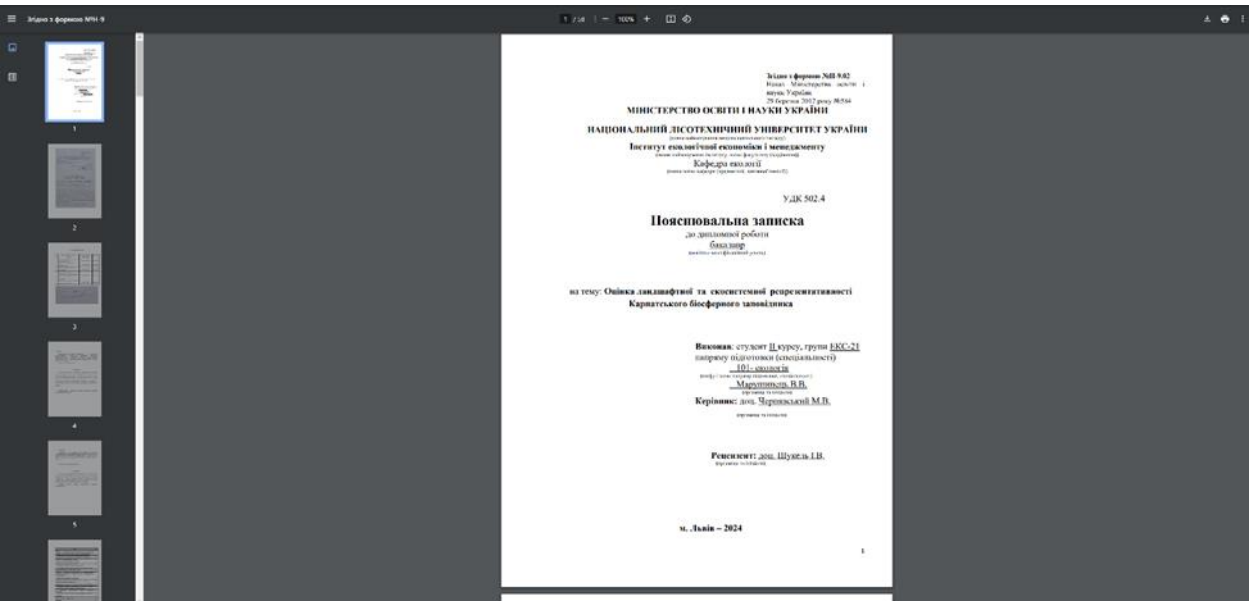


Рисунок 1.2 – Приклад оформлення рефератів наукових робіт у репозиторії [12]

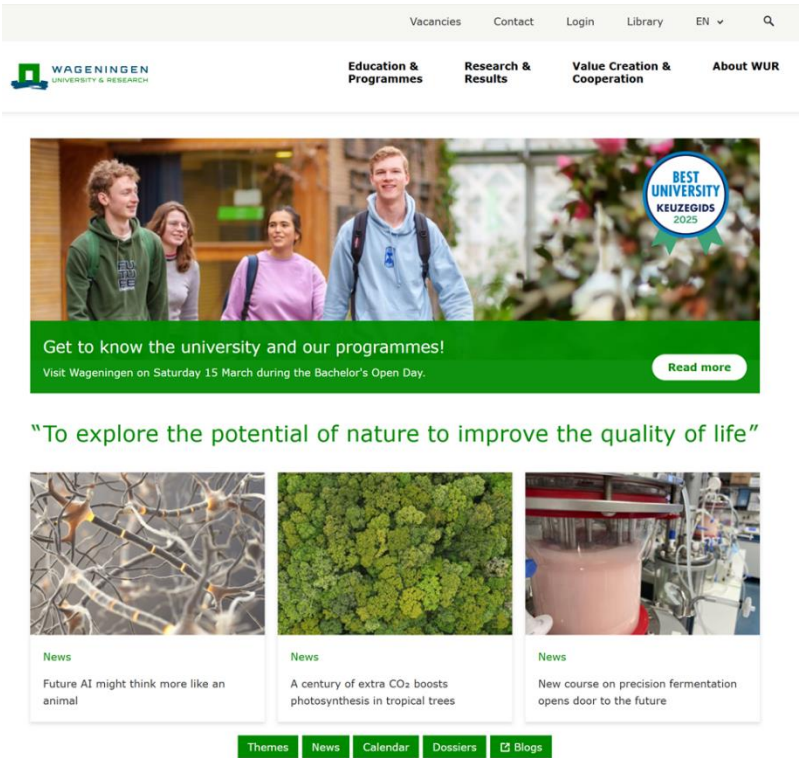


Рисунок 1.3 – Головна сторінка Wageningen University & Research з стрічкою новин [13]



Рисунок 1.4 – Приклад новини на порталі Національного університету "Львівська політехніка" [14]

Як свідчать наведені приклади, окрім детальної звітності в офіційних документах, університети активно використовують короткі новини для оперативного інформування у зручній та доступній формі. Наступним важливим етапом у висвітленні екологічних ініціатив є використання мультимедійного контенту, зокрема інтерактивних карт, фотографій та відеоголошень. Наприклад, University of California, Davis доповнює всі новини фотографіями та інтерактивними медіафайлами, а University of Nottingham активно використовує соціальні мережі, створюючи короткі відеоматеріали, що популяризують досягнення університету. Приклади таких публікацій наведено на рис. 1.5 та 1.6 [15]-[16].

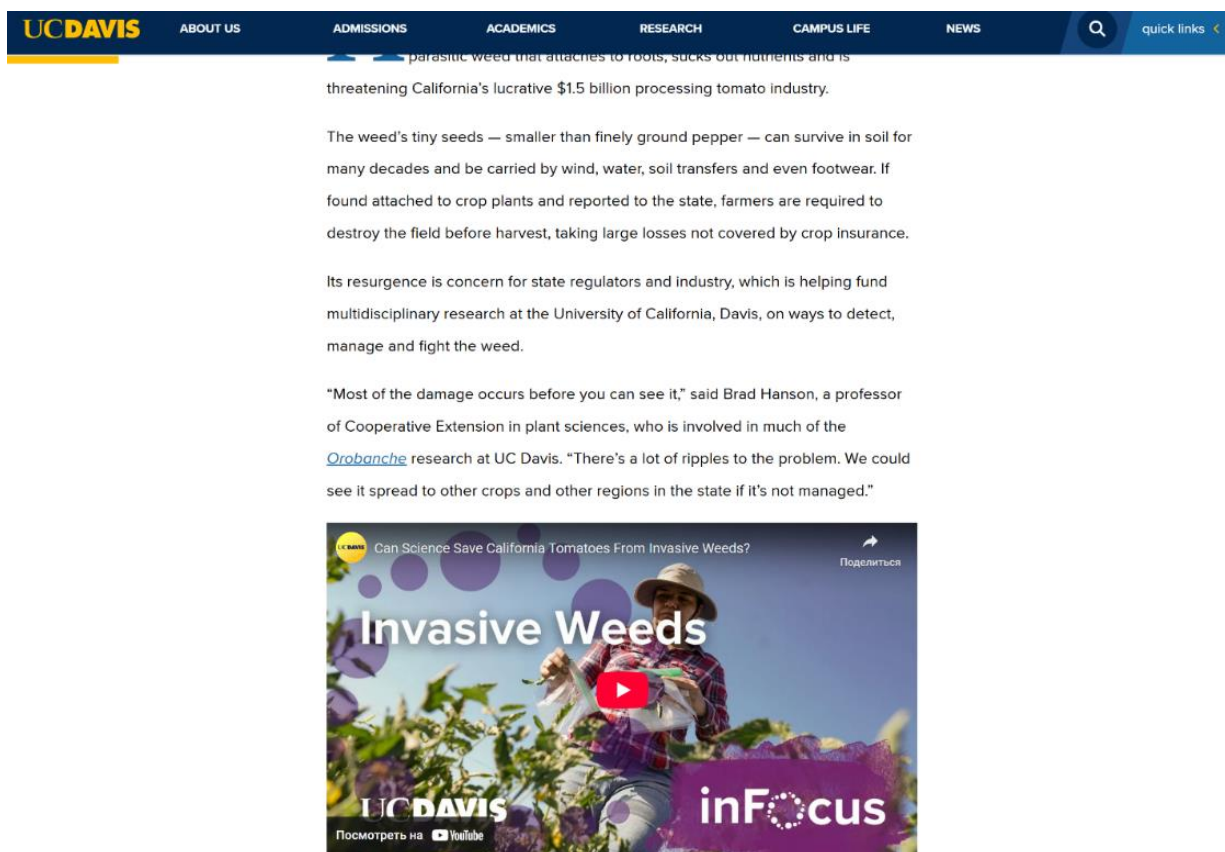


Рисунок 1.5 – Приклад публікації University of California, Davis, що доповнена відеоматеріалами [15]

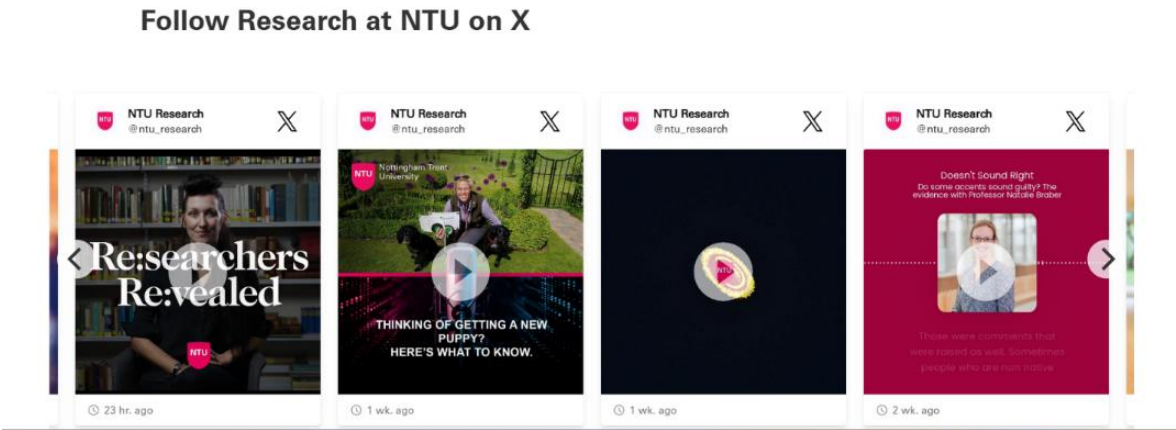
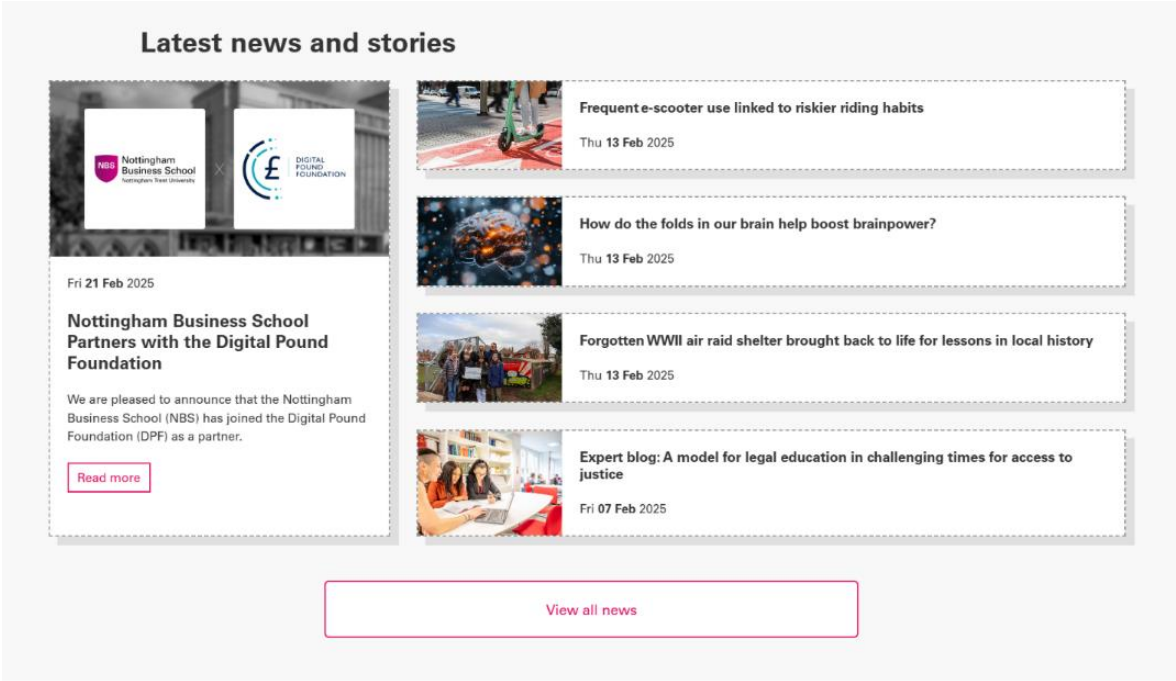


Рисунок 1.6 – Приклад порталу University of Nottingham з посиланнями на соц. мережі. [16]

На основі аналізу методів висвітлення екологічних досягнень в зелених університетах України та світу можна узагальнити результати у таблиці 1.1 та 1.2, відповідно, які дозволять порівняти різноманітність їх підходів до висвітлення екологічних ініціатив.

Таблиця 1.1 – Методи висвітлення досягнень в зелених університетах

України

№	Назва закладу	Засіб освітлення досягнень	Екологічні досягнення на головній сторінці
1	Національний лісотехнічний університет	– науковий вісник НЛТУ України. – стрічка новин; – репозитарій наукових робіт.	Інформація про екологічні досягнення представлена обмежено. Головна сторінка містить посилання на науковий вісник та репозитарій, однак стрічка новин переважно висвітлює адміністративні питання та події в університеті.
2	Сумський державний університет	– стрічка новин; – репозитарій наукових робіт.	Висвітлення екологічних досягнень є мінімальним. Головна сторінка містить стрічку новин, яка здебільшого інформує про внутрішні події університету. Також є доступ до репозитарію наукових праць та рейтингових списків, зокрема GreenMetric.
3	Національний університет «Острозька академія»	– наукові журнали; – репозитарій наукових робіт; – офіційний YouTube-канал; – стрічка новин.	Інформація про екологічні досягнення майже відсутня. На головній сторінці розміщене слайд-шоу з рекомендацією перегляду панорамного туру "зеленого університету". Стрічка новин та YouTube-канал здебільшого висвітлюють внутрішні події університету.
4	Національний європейський університет	– промоційна брошура «Green University»; – сторінки досягнень; – репозитарій наукових робіт; – звіти зі сталого розвитку та стратегічні плани.	Екологічні ініціативи висвітлені широко. Головна сторінка містить чіткі візуальні елементи, що звертають увагу на екологічній спрямованості університету. Також доступні численні посилання на стратегії сталого розвитку та відповідні звіти.
5	Національний університет «Львівська політехніка»	– стрічка новин; – репозитарій наукових робіт; – наукові журнали та збірники.	Екологічна діяльність університету представлена помірно. На головній сторінці наявні посилання на рейтингові позиції університету, включно з GreenMetric. Серед новин періодично публікуються матеріали про екологічні конференції та ініціативи.

Таблиця 1.2 – Методи висвітлення досягнень в зелених університетах світу

№	Назва закладу	Засіб освітлення досягнень	Екологічні досягнення на головній сторінці
1	2	3	4
1	Wegenigen university & research	– стрічка новин; – список нагород у зеленій сфері; – інтерактивні промопублікації активних зелених проєктів; – звіти про досягнення цілей сталого розвитку; – репозитарій наукових робіт.	Високий рівень. Головна сторінка одразу інформує відвідувачів про екологічну спрямованість університету та його першість у рейтингах, зокрема GreenMetric. Доступна відкрита структура з повним доступом до звітів про екологічні ініціативи та впровадження принципів сталого розвитку.
2	Nottingham Trent University	– стрічка новин; – інтерактивні матеріали про екологічні проєкти; – соціальні мережі; – репозитарій наукових робіт.	Помірний рівень. Головна сторінка в першу чергу висвітлює досягнення університету в культурі, науці та спорті, приділяючи менше уваги екологічним питанням. Однак в інтерактивних матеріалах про наукові здобутки значну увагу відведено вирішенню екологічних проблем та досягненню цілей сталого розвитку.
3	University of Groningen	– відкриті подкасти; – відеопублікації; – таблиця новин; – дослідницький центр; – репозитарій наукових робіт.	Високий рівень. Головна сторінка акцентує на екологічних ініціативах. Для перегляду всіх наукових здобутків є доступ до дослідницького центру, що містить відеоматеріали, подкасти, новини та репозитарій наукових робіт.
4	University College Cork	– соціальні мережі; – стрічка новин; – репозитарій наукових робіт.	Високий рівень. Головна сторінка університету одразу привертає увагу до досягнень у сфері медицини та боротьби з екологічними проблемами.

Кінець таблиці 1.2

1	2	3	4
5	Universidade de Sao Paulo USP	– стрічка новин; – репозитарій наукових робіт.	Високий рівень. Значна увага приділена екологічним досягненням, зокрема успішному впровадженню енергоефективних технологій та екологічних ініціатив у Португалії.

Провідні зелені університети активно працюють над висвітленням своїх екологічних ініціатив, використовуючи широкий спектр комунікаційних методів. З розвитком цифрових технологій заклади вищої освіти дедалі частіше переходять від традиційних форматів звітності до інтерактивних засобів комунікації – зокрема фото-, відеоматеріалів та соціальних мереж [10].

Однак для підвищення впізнаваності університету не лише в рейтингу UI GreenMetric, а й в очах зацікавлених сторін – інвесторів, партнерів, студентів – недостатньо просто використовувати сучасні канали поширення інформації. Важливо також стратегічно обирати саму інформацію, яка буде презентована, аби створити привабливий, авторитетний та послідовний образ екологічно свідомого закладу. У наступному розділі буде розглянуто офіційні вебпортали провідних «зелених» університетів світу як один із ключових інструментів формування публічного іміджу, а також проаналізовано, які саме повідомлення, досягнення й ініціативи дозволяють їм закріпити статус активного борця за сталий розвиток.

1.3 Формування зеленого іміджу через інформаційні акценти

Ефективне формування зеленого іміджу університету значною мірою залежить від того, яку саме інформацію заклад вищої освіти обирає для публічного представлення. Сайт університету – це перша точка контакту для більшості зацікавлених сторін: абітурієнтів, партнерів, дослідників, інвесторів, громадськості. Саме тому екологічна складова повинна бути не

лише наявною, а й стратегічно вмонтованою в структуру вебресурсу та інформаційну політику [17].

На прикладі сайтів провідних зелених університетів світу можна виокремити основні типи інформації, які використовуються для закріплення статусу екологічно відповідального закладу. Це, зокрема, розділи про екологічну політику, звіти щодо сталого розвитку, висвітлення студентських ініціатив, наукових досліджень, екоінфраструктури, цілей сталого розвитку тощо. У таблиці 1.3 представлено порівняльний опис структури екологічної інформації на офіційних порталах університетів, що займають провідні позиції у світовому рейтингу UI GreenMetric.

Таблиця 1.3 – Інформаційні акценти на сайтах провідних зелених університетів світу

№	Назва закладу	Інформаційний акцент	Джерело
1	2	3	4
1	Wegenigen university & research	Лозунг «To explore the potential of nature to improve the quality of life»	https://www.wur.nl/
		Чотири нагороди з вказанням місць у рейтингах.	
		Сторінка «sustainability» з вказанням цілей, політики та досягнень університету в зеленій сфері	https://www.wur.nl/en/about-wur/sustainability.htm
		Сторінка «strategy» з описом плану досліджень для покращення якості життя та екології на 2025-2028 роки	https://www.wur.nl/en/about-wur/strategy.htm
2	Nottingham Trent University	Новина про клуб садівництва	https://www.ntu.ac.uk/about-us/events/events/2025/4/gardening2
		Сторінка «strategy» з описом екологічної стратегії, стратегії впровадження сталого розвитку та вказанням другого місця в Greenmetric	https://www.ntu.ac.uk/about-us/strategy
3	University of Groningen	Новина «Smart energy management»	https://www.rug.nl/

Кінець таблиці 1.3

1	2	3	4
4	University College Cork	Новина «Harnessing storytelling and the arts to inspire climate action»	https://www.ucc.ie/en/news/2025/harnessing-storytelling-and-the-arts-to-inspire-climate-action.html
		Сторінка «World's first campus» зі статтею про тривалість зеленого статусу, місця у рейтингу та нагороди	https://www.ucc.ie/en/greencampus/
5	Universidade de Sao Paulo USP	Статус п'ятого найстійкішого університету світу.	https://www5.usp.br/
		Сторінка «Цінності» з описом стратегій та цілей університету	https://www5.usp.br/missao-visao-e-valores/
		Сторінка «sustentavel» з виділення основних досягнень в екології та досягненні цілей сталого розвитку	https://jornal.usp.br/usp-sustentavel/

Аналіз інформаційних стратегій світових лідерів дозволяє сформувати уявлення про типові акценти в подачі контенту, однак для повнішого розуміння варто також проаналізувати, як екологічна тематика репрезентована на порталах українських університетів. Це дозволить виявити національні особливості, потенційні зони росту та рівень адаптації до глобальних тенденцій. Порівняльний огляд наведено у таблиці 1.4

Аналіз змісту вебпорталів дозволив дійти висновку, що провідні університети з високими позиціями в рейтингу UI GreenMetric приділяють значно більше уваги публічному представленню своїх екологічних ініціатив, ніж українські заклади вищої освіти, що перебувають у рейтингу, починаючи з 210-го місця. Зокрема, як лідери рейтингу, так і, наприклад, Національний лісотехнічний університет, мають візуально впізнавані зелені елементи на головній сторінці сайту, що одразу сигналізує про екологічну спрямованість університету.

Таблиця 1.4 – Інформаційні акценти на сайтах провідних зелених університетів України

№	Назва закладу	Інформаційний акцент	Джерело
1	Національний лісотехнічний університет	Новини про участь університету в екологічних міжнародних проєктах.	https://nltu.edu.ua/
		Наукове фахове видання "Науковий вісник НЛТУ України"	https://nv.nltu.edu.ua/index.php/journal
		Журнал "Український ліс" та газета «Освіта лісівнича»	https://nltu.edu.ua/index.php/novyny/dodatкова-informatsiia/hazeta-osvita-lisivnycha
2	Сумський державний університет	Сторінка «Ми в рейтингах»	https://sumdu.edu.ua/uk/about-sumdu/rating.html
		Стрічка новин про досягнення університету в різних конкурсах	https://sumdu.edu.ua/uk/
3	Національний університет «Острозька академія»	Науковий журнал «Public Health Journal»	https://www.oa.edu.ua/ua/science/sjournal/
4	Національний європейський університет (МЄУ)	Сторінка «Сталість в МЄУ»	https://sustainability.ieu.edu.ua/
		Стратегія розвитку МЄУ	https://ieu.edu.ua/docs/strategiia.pdf
5	Національний університет «Львівська політехніка»	Львівська політехніка у рейтингах закладів вищої освіти	https://lpnu.ua/news/lvivska-politekhnika-u-reitynhakh-zakladiv-vyshchoi-osvity
		Стратегічний план розвитку Львівської політехніки до 2025 року	https://lpnu.ua/2025

Іноземні університети відкрито демонструють свою участь в екологічних процесах: вони активно афішують досягнення, використовують

сучасні канали інформування – відео, інтерактивні стрічки новин, а також акцентують на своїх позиціях у глобальних рейтингах [10]. Натомість в українських університетах подібна інформація часто захована в глибоких підменю, а на головній сторінці лише епізодично з'являються новини, дотичні до екології. Інформація про сталий розвиток, у кращому випадку, згадується в документах або аналітичних звітах, що не сприяє її доступності для широкої аудиторії.

Станом на сьогодні наш університет істотно поступається за рівнем «зеленості» та інформаційної представленості у глобальному просторі. Проте саме інформаційна відкритість може стати першим кроком до формування сталого іміджу. Налагодження структури екологічного контенту на вебпорталі, створення впізнаваної візуальної айдентики та системний підхід до екокомунікацій – це та база, на яку згодом будуть додаватися реальні досягнення, здобуті в процесі реалізації плану дій на 2025–2027 роки [18].

Варто зазначити, що жоден із проаналізованих вебпорталів зелених університетів не використовує інструментів на основі чатботів. Упровадження такого елементу може стати унікальною особливістю, що одразу виокремить наш університет серед конкурентів. Особливо актуальним буде використання генеративних моделей штучного інтелекту, які дозволять створити інтуїтивний інтерфейс для швидкого доступу до екологічної інформації. Це не лише покращить користувацький досвід, але й стане ефективним інструментом позиціювання університету як інноваційного, динамічного й відкритого до сучасних підходів у сфері сталого розвитку [19].

Щоб забезпечити максимально корисний функціонал такого чатбота, необхідно спиратися на ті інформаційні блоки, які вже довели свою ефективність у міжнародній практиці. Саме тому важливо проаналізувати, на які саме елементи акцентують увагу провідні зелені університети світу.

Проаналізувавши структуру їх вебпорталів, можна виокремити ключові інформаційні блоки, що формують позитивний сталий імідж в очах користувачів. Ці елементи не лише підсилюють комунікацію екологічної

політики закладу, а й створюють чітке уявлення про системність та прозорість його екодіяльності. До таких необхідних компонентів належать:

- позиція університету в екологічних рейтингах (зокрема UI GreenMetric);
- останні новини та події, пов'язані з екологічними досягненнями;
- план сталого розвитку із чіткими цілями та часовими рамками;
- публічний звіт про виконані заходи та досягнення;
- формалізована екологічна політика університету;
- посилання на наукові публікації з тематики сталого розвитку;
- виділена сторінка або розділ порталу під загальною назвою Sustainability, де зібрана уся релевантна інформація.

Цей перелік може слугувати не лише основою для побудови ефективної інформаційної структури вебпорталу, а і є ключовим для розуміння функціоналу майбутнього чатботу. Саме ці пункти повинні бути інтегровані у його навігацію та логіку, аби забезпечити швидкий, зручний і релевантний доступ до екологічної інформації для кожного користувача.

1.4 Роль ШІ у проєктах зелених університетів

1.4.1 Можливості ШІ в проєктах зелених університетів

Штучний інтелект відкриває значні можливості для вирішення складних завдань у рамках екологічних ініціатив університетів, сприяючи автоматизації процесів та підвищенню ефективності реалізації стратегій сталого розвитку. Його здатність аналізувати великі обсяги даних, автоматизувати рутинні процеси та генерувати інноваційні рішення дозволяє підвищити ефективність реалізації екологічних ініціатив [20]-[23]. Особливу роль у цьому відіграють генеративні моделі, які знаходять застосування в різних напрямках:

- моніторинг навколишнього середовища. Генеративні моделі використовуються для аналізу та прогнозування змін в екосистемах на основі

даних із супутників, дронів та інших джерел. Наприклад, ШІ може оцінювати стан лісових масивів, рівень забруднення води чи повітря та створювати візуальні звіти для своєчасного виявлення критичних змін [20];

- оптимізація енергоспоживання. ШІ здатний моделювати та прогнозувати споживання енергії в університетських кампусах, що дозволяє мінімізувати витрати, скорочувати викиди та знижувати навантаження на енергетичні мережі. Зокрема, він може створювати сценарії оптимізації для різних типів будівель, адаптуючи їх під потреби користувачів [21]-[23];

- управління відходами. Завдяки алгоритмам ШІ можна прогнозувати кількість відходів, їх компоненти та розробляти ефективні системи переробки. Це особливо актуально для великих університетських кампусів, де використання таких моделей дозволяє оптимізувати сортування та забезпечувати повторне використання матеріалів [23];

- проектування екологічних конструкцій. Алгоритми ШІ можуть створювати 3D-моделі екологічно стійких будівель, аналізувати їх ефективність з урахуванням кліматичних умов, енергозбереження та використання екологічних матеріалів. Це дає змогу університетам оптимізувати інфраструктуру, підвищуючи її відповідність принципам сталого розвитку [23];

- підтримка наукових досліджень. ШІ надає можливість автоматизувати обробку великих обсягів інформації, виконувати складні симуляції та прогнозувати результати досліджень. Наприклад, він може допомагати в моделюванні кліматичних змін і оцінювати ефективність екологічних заходів [23].

Отже, штучний інтелект виступає не лише засобом оптимізації існуючих екологічних процесів, а й фундаментальним інструментом для розробки інноваційних рішень у сфері сталого розвитку університетів.

1.4.2 Приклади успішного використання ШІ в екологічних проєктах

Штучний інтелект вже сьогодні активно використовується для вирішення складних екологічних завдань, демонструючи свою ефективність у численних реальних проєктах. Такі ініціативи ілюструють, як застосування ШІ може сприяти сталому розвитку, підвищенню екологічної безпеки та ефективності управління ресурсами [24].

Реальні приклади таких проєктів наведено в таблиці 1.5, яка узагальнює ключові ініціативи та їхній вплив на екологію.

Таблиця 1.5 – Приклади використання ШІ в екологічних проєктах

№	Назва проєкту	Опис
1	DeepMind's AI for Wind Energy [25]	Компанія DeepMind розробила алгоритм ШІ, який прогнозує виробництво енергії на вітрових електростанціях за 36 годин до фактичного вироблення. Це дозволяє оптимізувати подачу енергії в мережу та підвищити ефективність використання вітрової енергії.
2	IBM's Green Horizon Project [26]	Проєкт IBM використовує ШІ для прогнозування рівня забруднення повітря в міських районах. Система аналізує погодні умови, дані про трафік та інші фактори, щоб надавати точні прогнози та допомагати містам впроваджувати ефективні заходи для покращення якості повітря.
3	Google's Environmental Insights Explorer [27]-[28]	Google розробила інструмент, який використовує ШІ для аналізу супутникових знімків та інших даних з метою оцінки викидів вуглецю в містах. Це допомагає муніципалітетам розробляти стратегії зі зменшення викидів та підвищення енергоефективності.
4	Wildlife Insights by Conservation International [29]	Платформа, що використовує ШІ для аналізу фотографій з прихованих камер, автоматично ідентифікуючи види тварин. Це значно прискорює процес збору даних про біорізноманіття та допомагає в охороні дикої природи.
5	Microsoft's AI for Earth Program [30]	Ініціатива Microsoft надає ресурси та підтримку проєктам, які використовують ШІ для вирішення екологічних проблем, включаючи моніторинг сільськогосподарських угідь, управління водними ресурсами та збереження біорізноманіття.

Хоча значну частину успішних проєктів із використання ІІІ реалізовано комерційними компаніями, наукові дослідження та ініціативи університетів відіграють не менш важливу роль у розвитку цієї сфери. Університети не лише застосовують ІІІ для вирішення екологічних задач у своїх кампусах, але й розробляють інноваційні рішення, які сприяють пошуку нових підходів до сталого розвитку. Приклади таких досліджень і проєктів наведено в таблиці 1.6.

Таблиця 1.6 – Використання ІІІ в екологічних проєктах університетів

№	Назва проєкту	Опис
1	BAIR Climate Initiative (Університет Каліфорнії Берклі) [31]	Ініціатива активно використовує технології штучного інтелекту для аналізу даних, що стосуються змін клімату, зокрема для моделювання екосистем і прогнозування впливу різних факторів на навколишнє середовище.
2	Система управління енергоспоживанням в кампусі (MIT) [32]	Використання ІІІ для моніторингу та оптимізації споживання енергії в університетських кампусах, що знижує викиди CO ₂ та підвищує енергоефективність будівель.
3	Автоматизований догляд за рослинами (University of Melbourne) [33]	ІІІ забезпечує аналіз даних про вологість ґрунту, освітленість і температуру, оптимізуючи догляд за рослинами та зменшуючи використання води.
4	Використання ІІІ для аналізу кліматичних змін (Stanford University) [34]	ІІІ аналізує великі обсяги кліматичних даних, моделює зміни клімату та оцінює вплив людської діяльності на навколишнє середовище.
5	Інтелектуальна система моніторингу парків (ХНУРЕ) [35]	Розроблено систему, яка використовує методи ІІІ для аналізу стану зелених насаджень і моніторингу парків, допомагаючи покращити управління екосистемами міста.
6	Західноукраїнський Національний Університет (WUNU) [36]	Розроблено віджет SaveEcoBot - єдину в Україні екологічну систему, яка поєднує дані про поточний стан довкілля, про забруднення, забруднювачів та інструменти захисту довкілля. Цей віджет виводить ці дані на зручній інтерактивній мапі, у графіках та числовій оцінці.

Аналіз наведених прикладів свідчить про зростання поширення технологій ІІІ як у комерційному секторі, так і в університетському середовищі, де вони застосовуються для досліджень і впровадження

інноваційних рішень. Університети не лише адаптують готові рішення, але й активно досліджують нові підходи до застосування штучного інтелекту для сталого розвитку.

Хоча ШІ ефективно використовується для обробки даних та управління автономними системами, це лише частина його функціоналу [2]. Наразі набуває популярності використання ШІ для взаємодії з користувачами, зокрема у форматі чатботів, однак заклади освіти ще не поспішають активно впроваджувати ці технології на свої портали. Це відкриває перспективи для дослідження можливостей застосування чатботів у сфері екологічної комунікації. У зв'язку з цим доцільно розглянути успішні реалізації подібних проєктів і оцінити потенціал їх інтеграції на порталах зелених університетів.

1.4.3 Успіхи чатботів на основі ШІ та їх впровадження в університетське середовище

Сучасні генеративні моделі штучного інтелекту, такі як ChatGPT від OpenAI та Gemini від Google, зробили значний прорив у сфері автоматизованої взаємодії з користувачами. Вони демонструють високу якість розуміння природної мови, здатність генерувати зв'язні та інформативні відповіді, а також інтегруватися у різні системи для вирішення широкого спектра завдань [37].

ChatGPT є однією з найпопулярніших моделей генеративного штучного інтелекту, що використовується для автоматизованого спілкування з користувачами у форматі текстового чату. Завдяки великій базі навчальних даних та гнучким налаштуванням, ця модель здатна надавати детальні відповіді, підтримувати діалог і навіть адаптуватися до потреб конкретних запитів [37].

З іншого боку, модель Gemini від Google також є важливим гравцем у сфері ШІ. Вона спеціалізується на мультимодальній обробці інформації, що дозволяє працювати не лише з текстом, але й із зображеннями, відео та аудіо.

Це відкриває нові можливості для університетських порталів, де користувачі можуть отримувати різносторонню екологічну інформацію у зручному форматі [38].

Також однією з провідних компаній у сфері інтеграції чатботів є Startduck, яка розробила модуль для впровадження ChatGPT у різноманітні вебресурси. Ця технологія дозволяє легко підключати інтелектуального чатбота до різних платформ – від форумів до комерційних проєктів, налаштовуючи його для виконання автоматизованих консультативних та інформаційних завдань [39].

Основні можливості модуля Startduck:

- інтеграція ChatGPT з будь-якими вебресурсами, включаючи форуми, корпоративні сайти та комерційні проєкти;
- гнучке налаштування ролі бота для автоматизованої взаємодії з користувачами;
- автоматизований збір інформації для CRM-систем та аналітики;
- підключення до сервісів Google, зокрема Google Drive, Google Meet та Google Calendar;
- синхронізація з популярними месенджерами для розширення охоплення аудиторії.

Завдяки таким рішенням власники вебресурсів можуть значно покращити взаємодію з користувачами, зробивши доступ до інформації більш інтерактивним та персоналізованим. Чатботи на основі ШІ дозволяють автоматизувати комунікаційні процеси, спрощуючи пошук необхідної інформації та підвищуючи рівень залученості аудиторії. Подібний модуль може бути адаптований для університетських порталів, оскільки вони часто функціонують як інформаційні хаби та стрічки новин [39].

1.5 Постановка задач дипломної роботи

Аналіз сучасних підходів до поширення інформації в університетах демонструє активне впровадження передових технологій, зокрема мультимедійних форматів та цифрових платформ. Університетські портали використовують широкий спектр інформаційних рішень, включаючи офіційні звіти, новинні стрічки, інтерактивні елементи та соціальні мережі [10]. Водночас розвиток штучного інтелекту та чатботів відкриває новий етап у комунікації, що поєднує концентрованість текстових форматів та інтерактивність сучасних медіа [37].

Хоча технології ШІ вже активно застосовуються у сфері освіти, їх використання переважно обмежується автоматизацією автономних систем та обробкою великих масивів даних. Системи управління енергоресурсами, аналіз екологічних показників та оптимізація внутрішніх процесів університетів є ключовими напрямками впровадження ШІ [23]. Однак, його потенціал для покращення взаємодії з користувачами залишається недостатньо розкритим.

При цьому комерційний сектор вже активно використовує ШІ для взаємодії з користувачами через чатботи. Такі рішення демонструють високу ефективність у наданні інформації, персоналізації комунікації та автоматизації консультаційних процесів. Впровадження подібних технологій у діяльність університетів може сприяти підвищенню доступності екологічної інформації та покращенню комунікації з аудиторією [37]-[39].

З огляду на це, метою роботи є розробка та впровадження чатбота, який забезпечує зручний доступ до екологічної інформації та допомагає формувати позитивний імідж університету як активного учасника зелених ініціатив.

Для досягнення цієї мети необхідно вирішити такі завдання:

- провести аналіз сучасних методів поширення інформації в університетах;

- дослідити інформаційні акценти, на які повинен звертати увагу чатбот;
- дослідити наявні технології ШІ, що використовуються для автоматизації комунікацій;
- визначити успішні комерційні приклади використання чатботів на основі ШІ;
- запропонувати концепцію інтеграції чатбота для університетського порталу;
- програмно реалізувати прототип чатботу для вебпорталу університету;
- оцінити перспективи впровадження такої технології у сфері екологічних комунікацій.

2 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Аналіз вимог

Першим кроком роєктування програмного забезпечення виступає аналіз технічного завдання та аналіз зазначених вимог. Виходячи з інформації у додатку А, основною метою розробка та впровадження чатбота, який забезпечує зручний доступ до екологічної інформації та допомагає формувати позитивний імідж університету як активного учасника екологічного захисту.

Згідно з додатком А, можна винести загальні вимоги до системи:

- доступність – чатбот має бути доступним через вебінтерфейс, що забезпечить простоту використання;
- інтерактивність – бот має підтримувати режим взаємодії з користувачем через меню, реалізоване у вигляді динамічного інтерфейсу з кнопками, що забезпечує навігацію, пошук за ключовими словами та категоріями;
- швидкість доступу – інформація повинна буди логічно розташована з швидким інтуїтивно зрозумілим інтерфесом;
- гнучкість оновлення – можливість швидкого редагування та оновлення контенту адміністраторами без потреби внесення змін у код;
- масштабованість – проєкт має мати потенціал розширення функціоналу, зокрема інтеграції з системами аналітики, опитуваннями, базами даних тощо;

Функціональні вимоги:

- надання короткої довідки про рейтинг GreenMetric та його критерії;
- демонстрація екологічних ініціатив університету у сфері енергетики, води, транспорту, відходів та освітніх проєктів;
- відповіді на часті запитання, пов'язані з екологічною діяльністю;
- навігація за темами через меню;

- підтримка мовної адаптації (українська мова як основна, з можливістю додавання інших мов).

Нефункціональні вимоги:

- сумісність із сучасними браузерами та мобільними пристроями;
- простий та інтуїтивний інтерфейс користувача;
- висока надійність і мінімізація помилок у роботі.

Функціональні вимоги до системи можна візуалізувати у вигляді діаграми прецедентів UML, яка відображає основні сценарії взаємодії користувачів із системою (рис. 2.1).

2.2 Архітектура програмного забезпечення

Архітектура програмного забезпечення визначає логічну структуру системи, взаємодію її компонентів, способи обробки запитів та інтеграції з зовнішніми сервісами. Для реалізації чатбота, що інформує про екологічну діяльність університету, обрано клієнт-серверну модель із підтримкою MVC-архітектури (Model-View-Controller), яка забезпечує чітке розділення обов'язків між компонентами системи та сприяє її масштабованості та підтримуваності [40].

Архітектурний шаблон MVC дозволяє ефективно відокремити логіку представлення, обробки даних і управління, що є надзвичайно важливим для підтримки й розширення системи, яка взаємодіє з REST API, планується оновлюватись у майбутньому та підтримувати нові інтерфейси. Це також забезпечує спрощену командну розробку, коли фронтенд і бекенд розробляються паралельно незалежними людьми [40].



Рисунок 2.1 – Діаграма прецедентів

Для кращого розуміння переваг архітектури MVC було проведено порівняння з іншими популярними підходами, зокрема MVP, MVVM і мікросервісною архітектурою (табл. 2.1).

На основі виявлених вимог сформовано архітектурне та функціональне бачення системи, яке буде розкрито у наступних підрозділах.

Таблиця 2.1 – Порівняння архітектурних підходів за ключовими критеріями [41]

Критерій	MVC [40]	MVP [42]	MVVM [43]	Мікросервіси [44]
Чіткість розподілу ролей	4	3	4	5
Потенціал для розширення системи	4	3	4	5
Складність впровадження	3	3	4	5
Зручність для односторінкових застосунків	5	3	5	2
Відповідність REST/IoT сценаріям	4	2	4	5
Швидкість початкового налаштування	5	5	3	2
Типові галузі застосування	Вебінтерфейси	Робочі станції	Мобільні/SPA	Хмарні платформи

Як видно з таблиці, хоча мікросервісна архітектура й забезпечує найвищу масштабованість і гнучкість, вона суттєво ускладнює реалізацію, вимагає розподіленої інфраструктури та не є виправданою для чатбота, що виконує порівняно прості функції обробки запитів і передачі інформації. Аналогічно, MVVM чудово підходить для складних SPA-додатків із високим

рівнем інтерактивності, але ускладнює реалізацію через необхідність реалізації двостороннього зв'язку між компонентами, що є надмірним для нашого завдання [43]-[44].

MVP переважно орієнтований на десктопні застосунки, а рівень розділення обов'язків у ньому не такий чіткий, як у MVC. Це може ускладнити масштабування чатбота в майбутньому або його інтеграцію з зовнішніми екосистемами [42].

Таким чином, MVC виявляється найкращим варіантом: він забезпечує чітку структуру коду, достатню масштабованість, підтримує інтеграцію з REST API, а також дозволяє легко впровадити як вебінтерфейс, так і зв'язок зі сторонніми сервісами. Порівняно з іншими архітектурами, MVC зберігає оптимальний баланс між функціональністю, гнучкістю та відносною простотою реалізації, що робить його цілком придатним для задач даного проєкту [40].

Компоненти системи організовані наступним чином:

- `model` – забезпечує взаємодію з базою даних, включаючи зберігання, оновлення та отримання структурованої інформації про екологічні ініціативи, рейтинги та запити користувачів;
- `view` – містить локальний інтерфейс адміністратора та тестову реалізацію чату, однак основна взаємодія з користувачем відбувається через зовнішній фронтенд;
- `controller` – відповідає за обробку вхідних запитів, керування логікою роботи бота, взаємодію з моделями та надання REST API для зовнішніх клієнтів.

Додатково в рамках архітектури реалізовано:

- сервіси – окремі програмні компоненти, що займаються обробкою даних, зверненнями до зовнішніх сервісів, а також реалізацією механізмів черг;
- фронтенд – створений за допомогою React, забезпечує зручний інтерфейс для користувача та адміністратора. Обмін даними з сервером

здійснюється через REST API, що забезпечує обробку запитів у реальному часі [41];

- інтеграційні модулі – передбачають можливість підключення до сторонніх сервісів для збагачення інформації, наприклад, екологічними новинами або даними з відкритих джерел.

Обрана архітектура забезпечує достатню гнучкість для подальшого розвитку проєкту, зокрема інтеграції машинного навчання, багатомовної підтримки або розширення функціоналу чатбота. На рисунку 2.2 візуально зображено архітектуру екологічного чатботу.

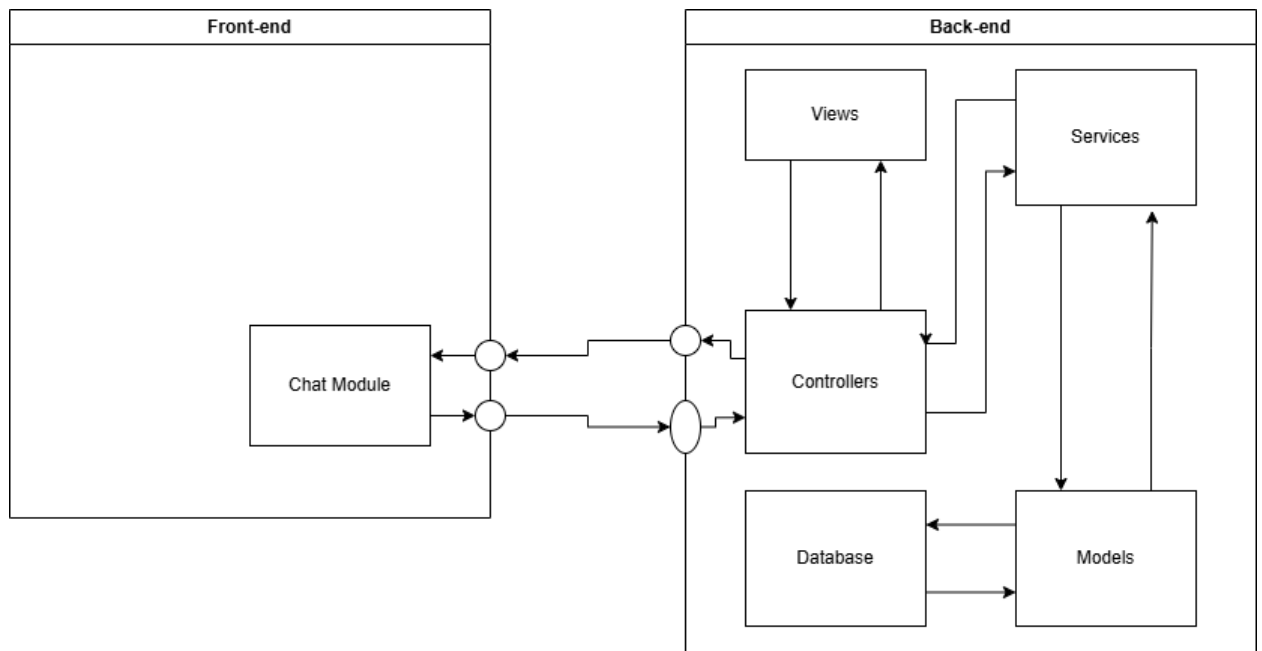


Рисунок 2.2 – Архітектура екологічного чатботу

2.3 Обґрунтування вибору технологій

Для реалізації веборієнтованої системи чатбота, що інформує користувачів про екологічну політику університету, було обрано набір технологій, які забезпечують високу ефективність, масштабованість, гнучкість та відповідність сучасним вимогам до розробки інтерактивних інформаційних систем. Обґрунтування вибору кожної з ключових технологій наведено нижче з урахуванням їх переваг порівняно з альтернативами.

2.3.1 Back-end: Laravel (PHP)

У якості серверного фреймворку було обрано Laravel – сучасний PHP-фреймворк, що реалізує архітектурний шаблон MVC (Model-View-Controller). Laravel відзначається широким набором вбудованих функцій: підтримка REST API, робота з БД через ORM Eloquent, обробка запитів, система маршрутизації, підтримка подій, черг, а також легкої інтеграції із зовнішніми сервісами. Також Laravel має високу популярність і потужну спільноту, що забезпечує активну підтримку, наявність великої кількості пакетів і розширень [45]. Детальніше порівняння серверних фреймворків зображено у таблиці 2.2.

Таблиця 2.2 – Порівняння серверних фреймворків

Критерій	Laravel (PHP) [45]	Django (Python) [46]	Express.js (Node.js) [47]	Spring Boot (Java) [48]
Архітектура	MVC	MTV (аналог MVC)	Мінімалістична	MVC
Швидкість розробки	Висока	Середня	Залежить від реалізації	Низька
Продуктивність	Висока	Висока	Дуже висока	Дуже висока
Інтеграція з фронтом	Зручна (React/Vue)	Складніше	Ручна інтеграція	Складна
Простота налаштування	Висока	Відносно проста	Багато ручної роботи	Складна
Наявність бібліотек	Велика екосистема	Велика	Фрагментовані	Дуже велика
Підтримка REST API	Вбудована	Вбудована	Потребує додаткового налаштування	Вбудована
Масштабованість	Висока	Висока	Висока	Дуже висока
Придатність до проекту	Оптимальна	Обмежено придатна	Може ускладнити реалізацію	Надмірна складність

Вибір Laravel зумовлений його балансом між швидкістю розробки, документацією, гнучкістю архітектури та інтеграційними можливостями.

2.3.2 База даних: MySQL / PostgreSQL

Для зберігання структурованої інформації (новини, наукові роботи, тощо) використовуються реляційні системи керування базами даних MySQL або PostgreSQL. Обидві СКБД підтримуються Laravel «з коробки» та забезпечують високу швидкість виконання запитів, підтримку транзакцій, індексацію та можливість масштабування. Порівняльні характеристики систем СКБД наведено в таблиці 2.3 [49].

Таблиця 2.3 – Порівняння СКБД

Критерій	MySQL [49]	PostgreSQL [50]	SQLite [51]	MongoDB (NoSQL) [52]
Тип	Реляційна	Реляційна	Вбудована реляційна	Документоорієнтована
Підтримка Laravel	Повна	Повна	Часткова	Через окремі пакети
Придатність до масштабування	Висока	Висока	Низька	Дуже висока
Робота з транзакціями	Так	Так	Обмежено	Обмежено
Побудова зв'язків	Чітка (JOIN, FK)	Чітка (JOIN, FK)	Обмежена	Слабка
Швидкість при великих обсягах	Висока	Висока	Середня	Висока
Призначення	Вебсистеми, CMS	Високонавантажені проекти	Мобільні, тестування	Big Data, гнучкі дані
Придатність до проекту	Оптимальна	Оптимальна	Непридатна	Неоптимальна

У рамках обраної архітектури використання MySQL/PostgreSQL є оптимальним рішенням, що дозволяє чітко структурувати дані і легко реалізувати зв'язки між сутностями, такими як наукові роботи, потоки, новини [49]-[50].

2.3.3 Front-end: React

Інтерфейс користувача реалізовано за допомогою React – популярної бібліотеки JavaScript для створення компонентного динамічного інтерфейсу. React дозволяє створити односторінковий застосунок (SPA) із високою швидкодією, повторним використанням компонентів та зручною роботою зі станами. Порівняльні характеристики інструментів створення інтерфейсів користувача можна побачити в таблиці 2.4 [53].

Таблиця 2.4 – Порівняння інструментів створення інтерфейсів користувача

Критерій	React [53]	Vue.js [54]	Angular [55]	jQuery [56]
Тип	Бібліотека	Бібліотека	Фреймворк	Бібліотека
Підтримка SPA	Повна	Повна	Повна	Обмежена
Поріг входження	Середній	Низький	Високий	Низький
Придатність для великих проєктів	Висока	Середня	Висока	Низька
Продуктивність	Висока	Висока	Висока	Низька
Повторне використання компонентів	Так	Так	Так	Ні
Популярність	Дуже висока	Висока	Висока	Низька
Придатність до проєкту	Оптимальна	Можлива	Неоптимальна	Застаріла

Таким чином, на основі таблиці 2.4, React обрано як найбільш популярну і підтримувану технологію, яка дозволяє ефективно реалізувати як чатінтерфейс, так і адміністративну панель [53]-[56].

2.3.4 Додаткові технології

JavaScript + HTML/CSS – використовуються для створення адаптивного інтерфейсу, що відповідає принципам доступності, інклюзивності та юзабіліті [57].

Laravel Queue та Horizon – застосовуються для обробки фонових задач, таких як оновлення новин, пошук по БД або інші ресурсоємні операції. Horizon надає зручний вебінтерфейс для моніторингу черг, що дозволяє своєчасно виявляти і усувати помилки в обробці запитів [58]-[59].

Таким чином, якщо зібрати усі обрані технології та розташувати їх в раніше наведену схему на рис. 2.2, то отримуємо бачення технічної реалізації чатботу як на рис. 2.3.

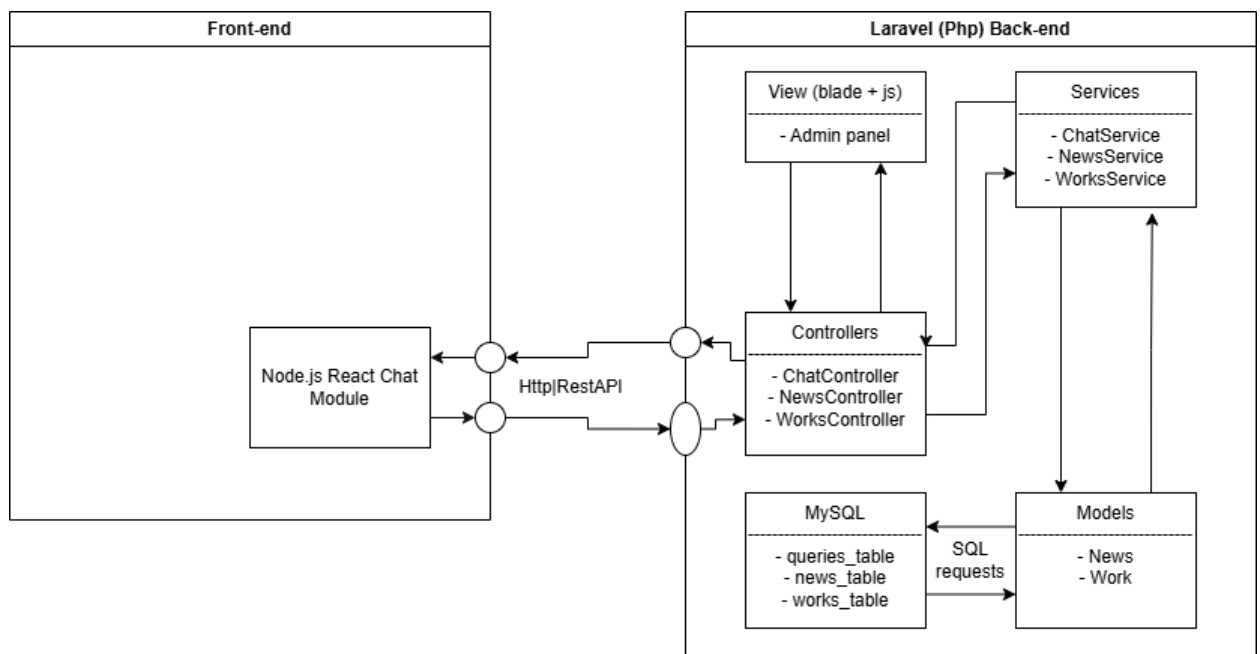


Рисунок 2.3 – Схема технічної реалізації обраної архітектури

2.4 Висновок до розділу 2

У результаті аналізу можливих технологічних рішень для реалізації чатбота, що інформує про екологічну політику університету, було сформовано оптимальний набір інструментів, який поєднує продуктивність, зручність розробки, підтримку спільноти та потенціал до масштабування. Основу серверної частини складає фреймворк Laravel, який реалізує шаблон MVC (Model-View-Controller), забезпечуючи чітке розділення логіки, представлення та взаємодії з базою даних. Для збереження структурованої інформації обрано реляційну СКБД MySQL, яка повністю інтегрується з Laravel через ORM

Eloquent. Фронтенд реалізовано за допомогою React, що дозволяє створити динамічний та адаптивний інтерфейс користувача.

Використання архітектури MVC дає змогу підтримувати високу структурованість коду, спрощує тестування та супровід проекту. Laravel Queue та Horizon дають змогу ефективно працювати з фоновими задачами, не перевантажуючи основний потік запитів. Обраний стек технологій передбачає масштабування системи за рахунок горизонтального розширення та дозволяє легко інтегрувати новий функціонал у майбутньому, включаючи додавання нових модулів, зовнішніх API чи інтерфейсів керування. Це забезпечує надійність і гнучкість розробленого рішення в контексті довготривалої експлуатації.

Обрані технології забезпечують необхідний функціонал для реалізації усіх вимог, визначених у технічному завданні. Фронтенд-частина на базі React дозволяє створити легкий, динамічний та швидкодіючий модуль інтерфейсу користувача, оптимізований для інтерактивної взаємодії. Використання Laravel на серверній стороні гарантує наявність вбудованих засобів для реалізації адміністративної панелі, локалізації контенту та повноцінного управління базою даних через ORM Eloquent. Таким чином, сформовано стабільне технологічне підґрунтя, що дозволяє перейти безпосередньо до етапу програмної розробки функціональних компонентів модуля.

3 РОЗРОБКА ПРОГРАМИ

На основі проведеного аналізу та визначення оптимального стеку технологій, сформовано чітке бачення архітектури системи. Вибір технологічних рішень, таких як Laravel для серверної частини, React для клієнтського інтерфейсу та MySQL як СКБД, забезпечує стабільну основу для реалізації функціоналу чатботу. Визначено структуру бази даних, принципи взаємодії між компонентами та канали обміну повідомленнями в реальному часі.

З урахуванням цього, можна переходити до безпосередньої розробки програмного забезпечення, яка включає реалізацію серверної логіки, створення клієнтського інтерфейсу, побудову адміністративної панелі, проведення тестування та впровадження готового рішення.

3.1 Розробка серверної частини

3.1.1 Структура бази даних

У системі використовується реляційна СКБД MySQL, до якої створено міграції для збереження кнопок та академічних ресурсів. Таким чином база даних має наступний вигляд.

Стандартні таблиці Laravel:

- users – зберігає облікові записи користувачів;
- failed_jobs – зберігає невдалі черги задач;
- password_resets – обробка скидання паролів;
- personal_access_tokens – для авторизації через Laravel Sanctum;
- migrations – для керування міграціями.

Кастомні таблиці:

- а) buttons – зберігає структуру меню, включаючи:
 - 1) id – стандартне поле будь-якої таблиці;

- 2) `ua_title`, `en_title` – назви кнопок українською та англійською мовами;
 - 3) `ua_description`, `en_description` – описи на двох мовах;
 - 4) `url` – посилання на джерело інформації;
 - 5) `parent_id` – ідентифікатор батьківської кнопки для створення вкладеної навігації;
 - 6) `timestamps`;
- б) `works` – зберігає екологічні ініціативи та проєкти:
- 1) `id` – стандартне поле будь-якої таблиці;
 - 2) `tag` – тег академічного ресурса («Звіт», «Конференція», «Новина»);
 - 3) `ua_name`, `en_name` – назви ресурсів українською та англійською мовами;
 - 4) `ua_description`, `en_description` – опис ресурсів на двох мовах;
 - 5) `url` – посилання на повний ресурс;
 - 6) `timestamps`.

Структуру бази даних доцільно зобразити у вигляді ER-діаграми, на якій можна побачити зв'язки між таблицями.

Для початкового наповнення даними створено три Seeder-класи:

- `UserSeeder`;
- `WorkSeeder`;
- `ButtonSeeder`.

Детальний опис класів, що відповідають за створення БД наведено у таблиці 3.1. Код можна знайти у Додатку Д (Д.1.1.1 – Д.1.2.4).

Таблиця 3.1 – Опис класів відповідних за створення БД

№	Назва класу	Метод	Призначення
1	2	3	4
1	<code>Create_users_table</code>	<code>up()</code>	Створення таблиці користувачів, в нашому випадку адміністраторів.
		<code>down()</code>	Видалення таблиці користувачів з усіма даними.

Кінець таблиці 3.1

1	2	3	4
2	Create_password_resets_table	up()	Стандарення таблиці запитів на сброс паролів, в цьому проєкті надлишкова, але є частиною стандартних таблиць Laravel.
		down()	Видалення таблиці з усіма даними.
3	Create_failed_jobs_table	up()	Створення таблиці для обробки помилкових асинхронних функцій.
		down()	Видалення таблиці з усіма даними.
4	Create_personal_access_tokens	up()	Створення таблиця збереження токенів для доступу до API.
		down()	Видалення таблиці з усіма даними.
5	Create_works_table	up()	Створення власної таблиці для збереження академ. ресурсів.
		down()	Видалення таблиці з усіма даними.
6	Create_buttons_table	up()	Створення власної таблиці для збереження кнопок меню.
		down()	Видалення таблиці з усіма даними.
7	Create_sessions_table	up()	Створення стандартної таблиці збереження сесій для авторизації через web.
		down()	Видалення таблиці з усіма даними.
8	UserSeeder	run()	Запуск скрипту заповнювання БД даними авторизації для адмінів.
9	WorkSeeder	run()	Запуск скрипту заповнювання БД тестовими академ. ресурсами.
10	ButtonSeeder	run()	Запуск скрипту заповнювання БД тестовими кнопками FAQ меню.
11	DatabaseSeeder	run()	Запуск скрипту, що викликає усі Seeder класи для заповнення БД даними..

ER-діаграму бази даних на якій зображено зв'язки між таблицями наведено на рисунку 3.1.

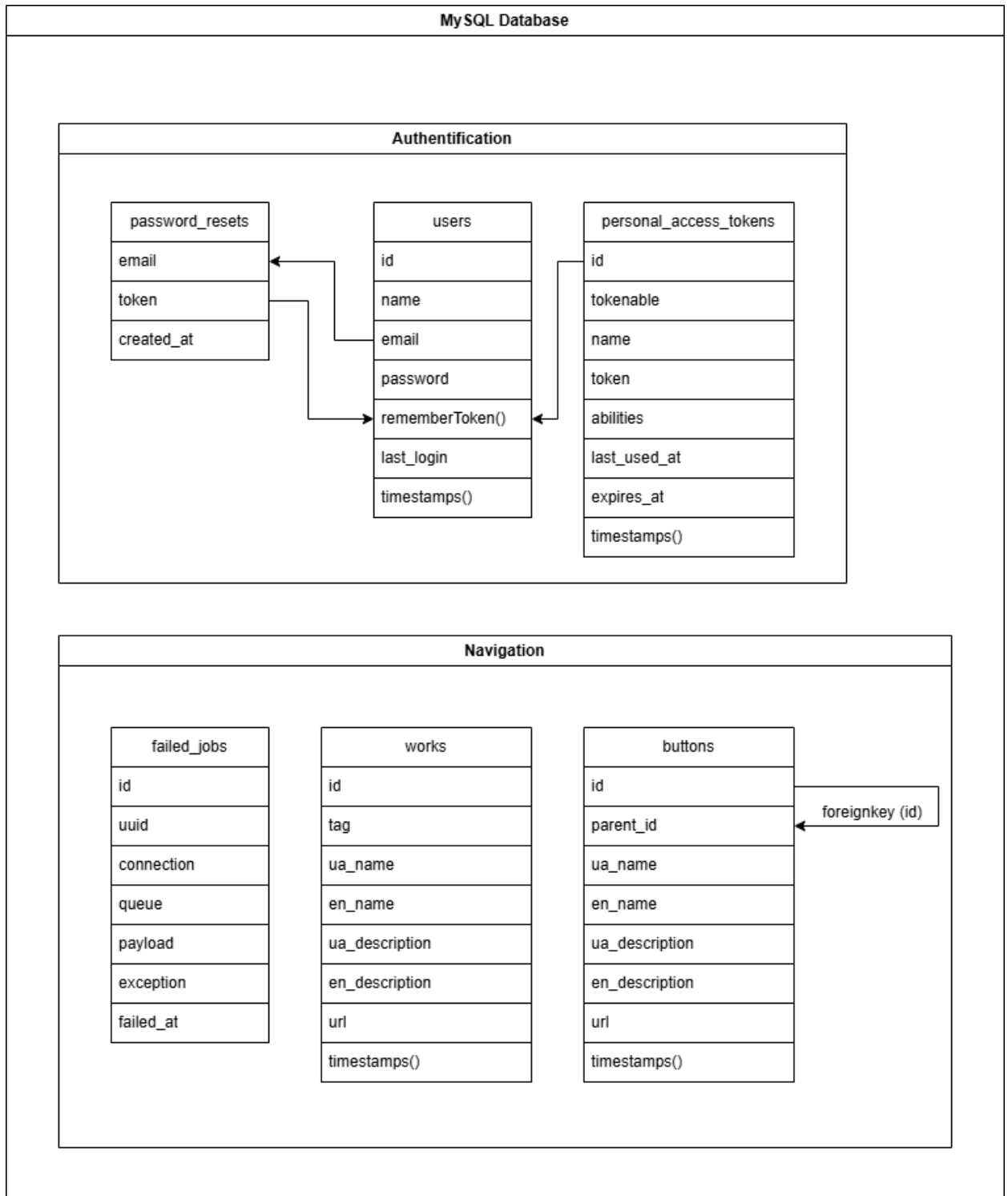


Рисунок 3.1 – ER-діаграма бази даних

3.1.2 Моделі

Реалізовано відповідні Eloquent-моделі:

- User;

- Button;
- Work.

Код моделей можна подивитись у підрозді Д.1.3 додатку Д. Детальний опис методів моделей показано у таблиці 3.2.

Таблиця 3.2 – Опис класів Eloquent-моделей

№	Назва класу	Оголошені змінні	Метод	Призначення
1	User	<code>\$fillable = ['name', 'email', 'password'];</code> <code>\$hidden = ['password', 'remember_token'];</code> <code>\$casts = ['email_verified_at' => 'datetime'];</code>		
2	Button	<code>\$fillable = ['ua_title', 'en_title', 'ua_description', 'en_description', 'url', 'parent_id'];</code> <code>\$hidden = ['created_at', 'updated_at'];</code>	children()	Отримання колекції об'єктів кнопок дочірнього підменю
			parent()	Отримання об'єкту батьківської кнопки
3	Work	<code>\$fillable = ['tag', 'ua_name', 'en_name', 'ua_description', 'en_description', 'url'];</code>		

3.1.3 Контролери

Логіка обробки запитів реалізована в наступних контролерах:

- AuthController – обробка логіну/логауту користувачів;
- ButtonController – видача структури меню та дочірніх елементів, управління таблицею Buttons та редагування її змісту;
- WorkController – управління сутністю екологічних робіт/проектів;
- HandBook\TagEnumController – надає перелік доступних тегів для категоризації.

Детальний опис класів контролерів наведено у таблиці 3.3. Код цих контролерів описано в Додатку Д, підрозділі Д.1.4.

Таблиця 3.3 – Опис класів контролерів

№	Назва класу	Метод	Призначення
1	AuthController	__construct()	Підключення сервісу автентифікації.
		login()	Виклик методу авторизації в вервісі з даними за запиту.
		logout()	Видалення токена користувача.
2	ButtonController	getMenus()	Віддає головне меню у вказаній локалізації.
		getChildren()	Віддає колекції дочірніх кнопок за вказаним id та локалізацією.
		store()	Створює нову запис в БД з даних надісланих у запиті.
		show()	Показує деталі певної кнопки разом з дочірніми кнопками.
		update()	Оновлює дані у БД згідно з запитом.
		destroy()	Видаляє запис про кнопку з БД.
3	WorkController	__construct()	Підключення сервісу роботи з академ. ресурсами.
		index()	Звертається до сервісу роботи з ресурсам та віддає колекцію академ. ресурсів за фільтром.
		store()	Звертається до сервісу роботи з ресурсам з даними для створення запису в БД.
		show()	Відображає академ. ресурс за id.
		update()	Звертається до сервісу роботи з ресурсам з даними для оновлення запису в БД.
		destroy()	Видаляє запит з БД.
4	TagEnumController	index()	Віддає список тегів з описов на двох мовах.

3.1.4 Перелічувані типи (Enum)

Для класифікації проєктів використано Enum-клас TagEnum, що визначає набір категорій для тегування робіт. Код Enum, класів атрибутів та методів можна подивитися у підрозділі Д.1.5 Додатку Д.

3.1.5 Сервіси

У межах принципів SOLID серверна логіка розділена на сервіси:

- AuthService (інтерфейс AuthServiceInterface) – авторизація.
- WorkService (інтерфейс WorkServiceInterface) – логіка обробки екологічних робіт.

Детальний опис методів класів сервісів можна подивитись у табл. 3.4. Код повної реалізації сервісів надано в Додатку Д, підрозділі Д 1.6.

Таблиця 3.4 – Опис класів сервісів

№	Назва класу	Метод	Призначення
1	2	3	4
1	AuthService	checkUserPassword()	Перевіряє чи відповідає зашифрований пароль користувача введеному.
		login()	Авторизує користувача та повертає bearer-token.
2	AuthServiceIntarface	checkUserPassword()	Інтерфейс методу checkUserPassword().
		login()	Інтерфейс методу login().
3	WorkService	createWork()	Створює запис про академ. ресурс згідно отриманих даних.
		updateWork()	Оновлює запис про академ. ресурс згідно отриманих даних.
		showWorks()	Повертає знайдені академ. ресурси згідно фільтрам.
		applySearch()	Фільтрує та обробляє локалізацію згідно фільтрам.

Кінець таблиці 3.4

1	2	3	4
4	WorkServiceInterface	createWork()	Інтерфейс методу createWork().
		updateWork()	Інтерфейс методу updateWork().
		showWorks()	Інтерфейс методу showWorks().
5	AppServiceProvider	register()	Зв'язує абстрактні інтерфейси з конкретною реалізацією у сервіс-контейнері.

3.1.6 Маршрути API

Взаємодія з клієнтською частиною відбувається через такі API-маршрути:

Публічні:

- POST api/login – Авторизація в адмін панель;
- POST api/logout – Вихід з адмін панелі;
- GET api/faq – Отримання головного меню FAQ;
- GET api/faq/{button} – Отримання дочірнього меню за кнопкою;
- GET api/tags – Отримання словника тегів для пошуковика;
- GET api/works – Отримання «Академ. ресурсів» за фільтром.

Захищені:

- CRUD api/works – Набір функцій для управління академічними ресурсами в БД;
- CRUD api/buttons – Набір функцій для управління академічними кнопками в БД.

3.1.7 Валідатори запитів (Requests)

Для зазначених маршрутів було створено валідатори запитів, які дозволять вносити чи діставати інформацію за фільтрами:

- а) Auth:

- 1) LoginRequest;
- б) FAQ:
 - 1) GetMenusRequest;
 - 2) GetChildrenRequest;
- в) Button:
 - 1) StoreButtonRequest;
 - 2) UpdateButtonRequest;
- г) Work:
 - 1) FindWorksRequest;
 - 2) StoreWorkRequest;
 - 3) UpdateWorkRequest.

Детальний опис роутингу та валідаторів можна побачити у табл. 3.5.
Код валідаторів та маршрутизації надано у підрозділах 1.8 та 1.9 Додатку Д.

Таблиця 3.5 – Опис валідаторів

№	Назва класу	Метод	Призначення
1	LoginRequest	rules()	Повертає правила валідації запитів на авторизацію.
2	StoreButtonRequest	rules()	Повертає правила валідації запитів на створення нової кнопки меню.
3	UpdateButtonRequest	rules()	Повертає правила валідації запитів на оновлення кнопки меню.
4	GetChildrenRequest	rules()	Повертає правила валідації запитів отримання підменю кнопки FAQ.
5	GetMenusRequest	rules()	Повертає правила валідації запитів на отримання головного меню FAQ.
6	FindWorksRequest	rules()	Повертає правила валідації запитів на пошук академ. ресурсів за фільтром.
7	StoreWorksRequest	rules()	Повертає правила валідації запитів на створення нового запису про академ. ресурс.
8	UpdateWorksRequest	rules()	Повертає правила валідації запитів на оновлення запису про академ. ресурс.

На основі реалізованих класів було побудовано діаграму класів (рис. 3.2), яка ілюструє усі створені класи та основні методи, необхідні для реалізації поставленої задачі. На рисунку представлено спрощену структуру бази даних, оскільки її детальний опис наведено на рисунку 3.1. Діаграма не охоплює елементи адміністративної панелі та клієнтської частини, оскільки їх розробка буде розглянута у наступних підпунктах.

Як видно з діаграми (рис. 3.2), реалізація повністю відповідає архітектурі MVC та дотримується принципів SOLID. Представлені класи та їх методи забезпечують повний контроль над даними, що зберігаються в базі, та створюють основу для подальшої реалізації і тестування інтерфейсів для користувачів і адміністраторів.

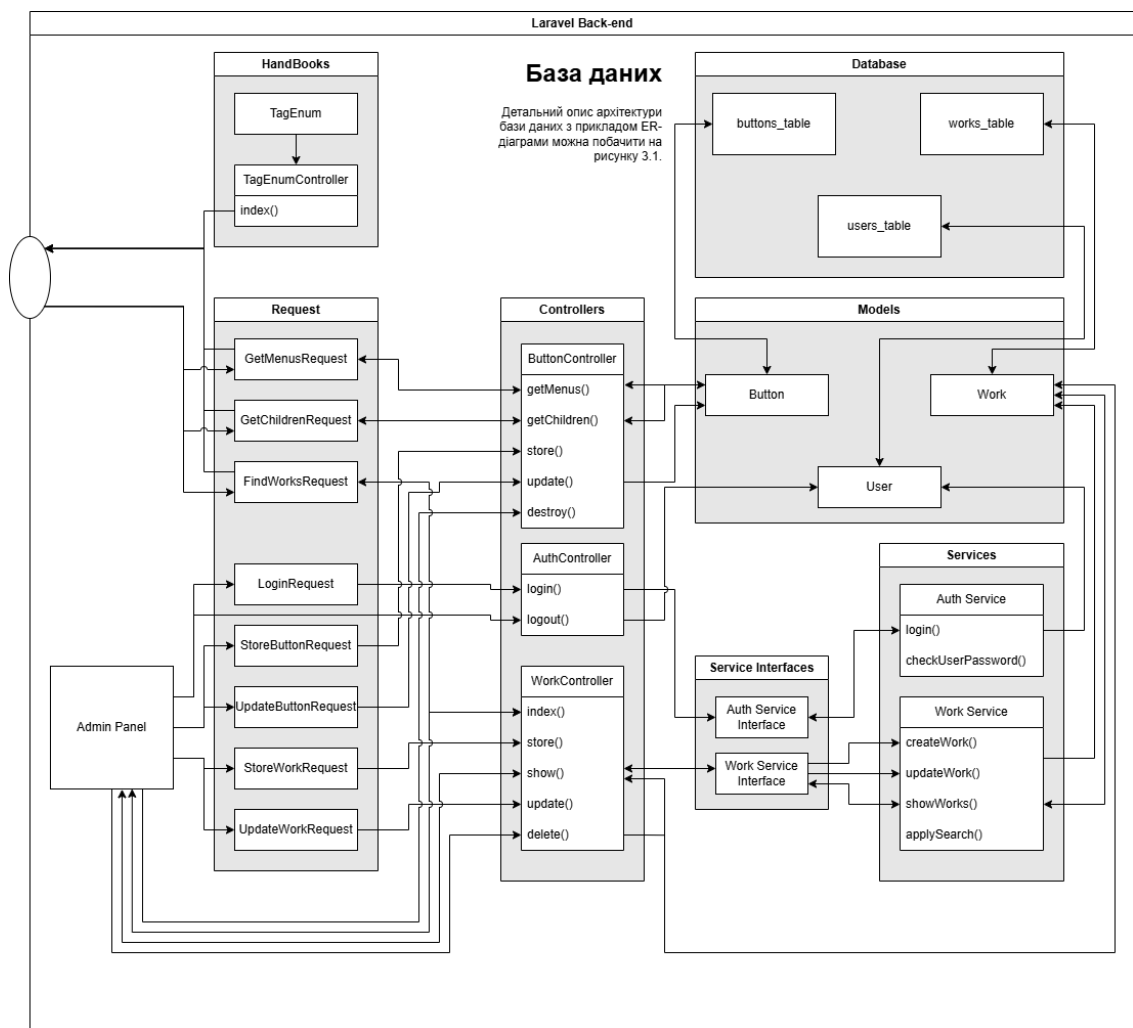


Рисунок 3.2 – Діаграма класів back-end частини (Laravel)

3.2 Розробка користувацького інтерфейсу

3.2.1 Розробка дизайну

Оскільки чатбот має екологічну тематику, при розробці його візуального оформлення було обрано відповідну кольорову гаму. Основним кольором інтерфейсу виступає зелений, який асоціюється з природою, сталим розвитком і загальною екологічною спрямованістю рішення. Допоміжними кольорами є білий (фон) та чорний (текст), що забезпечує високий рівень контрастності, доступності та зручності читання. Крім того, часткове використання синього кольору обумовлено стилістикою основного вебпорталу університету, який побудовано на контрасті білого та синього. Проте синій колір використовується обмежено, аби не знецінити екологічний акцент інтерфейсу.

Чатбот реалізований у вигляді віджету, що розміщується у правому нижньому куті сторінки (рис. 3.3). Він прив'язаний до елементів відображення браузера, що забезпечує його фіксацію на екрані незалежно від змісту сторінки. Віджет відкривається поверх контенту та надає доступ до двох основних вкладок: "FAQ" та "Академічні ресурси" (рис. 3.4).



Рисунок 3.3 – Віджет у згорнутому стані на платформі зеленого університету

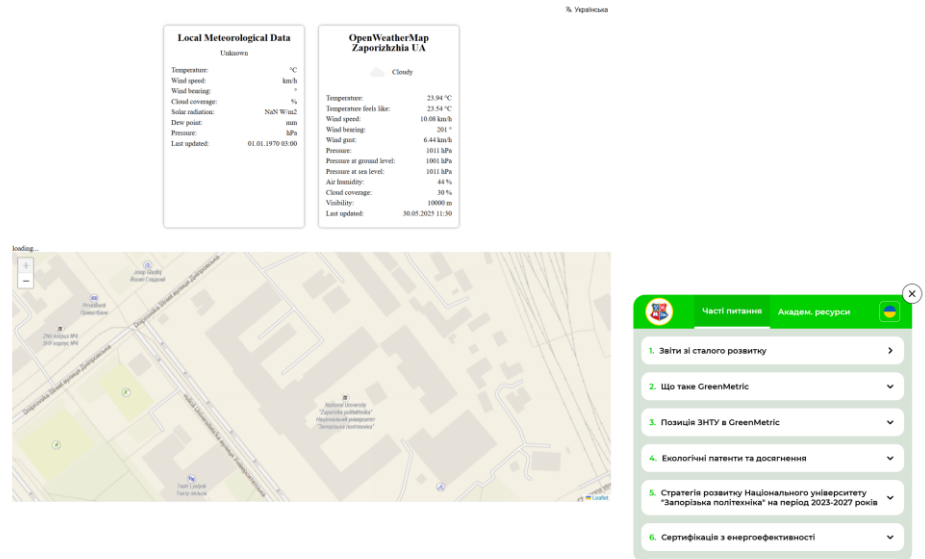


Рисунок 3.4 – Віджет у розгорнутому стані на цифровій платформі зеленого університету

Вкладка FAQ реалізована у вигляді ієрархічного списку кнопок, що можуть розгортатися за допомогою акордеону. Це дозволяє користувачам переглядати короткий зміст відповідної теми, переходити за гіперпосиланням до джерела інформації або відкривати підменю з додатковими підпунктами (рис. 3.5). Для зручної навігації реалізовано механізм «хлібних крихт», що дає змогу легко повертатися на попередні рівні меню. Важливою функціональністю є підтримка двох мов – української та англійської. Локалізація визначається на основі поточних налаштувань сайту, проте користувач може змінити мову інтерфейсу безпосередньо у віджеті.

Вкладка "Академічні ресурси" реалізує функціонал пошуку матеріалів за заданими критеріями. Користувач може обрати тег (категорію) з випадаючого списку та ввести ключове слово для точнішого пошуку. Результати виводяться у вигляді структурованих блоків-кнопок з акордеонами, аналогічно до FAQ, де також міститься короткий опис ресурсу та активне посилання на першоджерело (рис. 3.6).

Такий підхід дозволяє у компактному форматі віджету чатбота ефективно надавати розширену текстову інформацію щодо екологічної

діяльності університету. Використана кольорова палітра інтуїтивно асоціюється з екологічністю, підсилюючи тематичну спрямованість інтерфейсу. Функціональні можливості розробленого дизайну повністю відповідають вимогам до користувацького інтерфейсу зазначеним в додатку А. Таким чином, на основі створеного дизайну в підрозділі 3.3.2 було розпочато верстку віджету та інтеграцію з базою даних і серверною частиною для роботи з реальними даними.

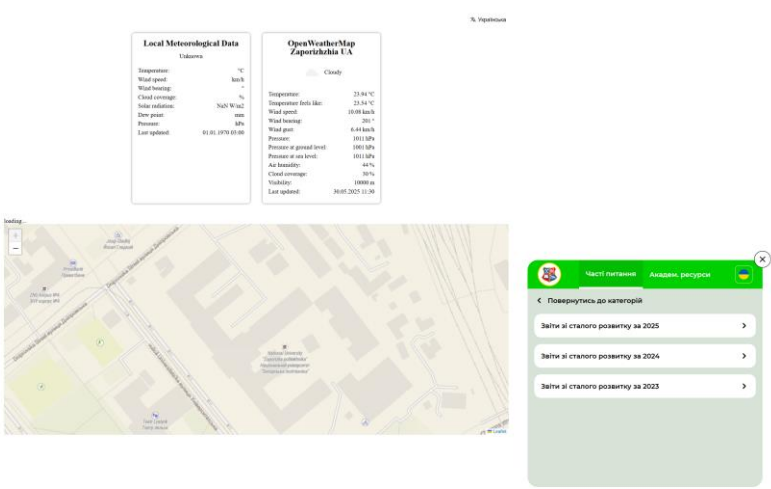


Рисунок 3.5 – Віджет FAQ з меню другого ступеню кладеності

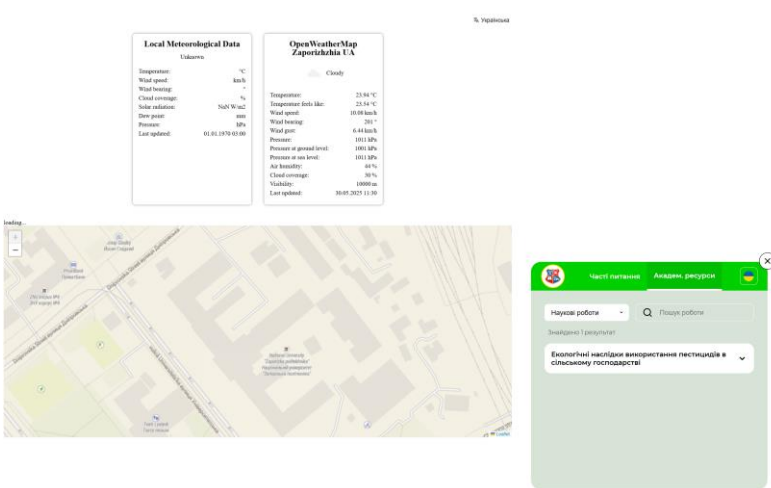


Рисунок 3.6 – Віджет «Академ. ресурси» з пошуковиком та акордеоном

3.2.2 Відтворення дизайну на React

Після завершення розробки дизайну користувацького інтерфейсу віджету чатбота, наступним етапом стала його технічна реалізація засобами фронтенд-розробки. Основним інструментом для відтворення інтерфейсу було обрано JavaScript-бібліотеку React, яка забезпечує компонентний підхід до побудови інтерфейсів, високу продуктивність та зручну інтеграцію з REST API.

У рамках реалізації було використано низку додаткових бібліотек і інструментів, які дозволили ефективно структурувати програмний код, організувати навігацію, підключити зовнішні сервіси, забезпечити кешування запитів і спростити стилізацію компонентів. Також для забезпечення швидкої розробки було обрано сучасний інструмент збірки Vite, що значно пришвидшує компіляцію та запуск середовища розробки. Додаткові бібліотеки та інструменти можна побачити у таблицях 3.6 та 3.7.

Таблиця 3.6 – Використані бібліотеки для реалізації інтерфейсу

Технологія	Призначення
React, ReactDOM	Основний фреймворк для побудови користувацького інтерфейсу
react-router-dom	Організація маршрутизації між вкладками (FAQ, Академічні ресурси)
@mui/material, emotion	Компоненти бібліотеки Material UI та інструменти для CSS-in-JS стилізації
@tanstack/react-query	Ефективне керування запитами до серверу, кешування результатів
axios	Бібліотека для виконання HTTP-запитів до бекенд API
sass	Препроцесор для стилів, що дозволяє створювати структурований SCSS-код

Таблиця 3.7 – Інструменти розробки

Інструмент	Призначення
vite	Швидка збірка, гаряче оновлення під час розробки
eslint	Аналіз якості коду, виявлення синтаксичних та логічних помилок
@vitejs/plugin-react	Плагін для коректної підтримки JSX та React у Vite

На основі розробленого дизайну (див. підрозділ 3.3.1) та обраних технологій було реалізовано користувацький інтерфейс віджету чатбота. React дозволив створити компонентну архітектуру, яка сприяє модульності, повторному використанню коду та зручному масштабуванню проєкту. Основними вимогами до функціоналу були: підтримка FAQ меню з вкладеністю, пошуковик академічних ресурсів, перемикач мов, селекція тегів, а також зручна навігація між вкладками. На основі побудованих компонентів було побудовано діаграму, наведену на рис. 3.7.

Основні компоненти:

- FAQPage – окремий компонент-сторінка, що реалізує розділ FAQ. Його завдання – вивід списку питань, побудованих за ієрархічною структурою. Взаємодія з цим компонентом дозволяє користувачу розкривати підменю та переглядати зміст з посиланнями на джерела.
- ResourcesPage – окрема вкладка, що відповідає за пошук академічних екологічних ресурсів. У межах цієї сторінки реалізовано фільтрацію за тегами та ключовими словами.
- Accordion – базовий елемент віджету, що забезпечує компактне відображення великої кількості інформації, дозволяючи поступово її розкривати. Використовується у FAQ та в результатах пошуку.
- SearchField – поле для введення ключового слова. Поєднується із селектором тегів для більш точного пошуку інформації у вкладці "Академічні ресурси".

- Button* – узагальнений компонент кнопок, реалізований із можливістю гнучкої генерації в залежності від переданих параметрів.
- RedirectButton, TagButton – спеціалізовані кнопки. Перша – відповідає за переходи до внутрішніх або зовнішніх джерел, друга – фільтрує ресурси за тегами.
- WidgetHeader, WidgetIcon – візуальні елементи віджету, включають логотип, заголовок, кнопку закриття та інші навігаційні іконки. Забезпечують інтуїтивну взаємодію з користувачем.

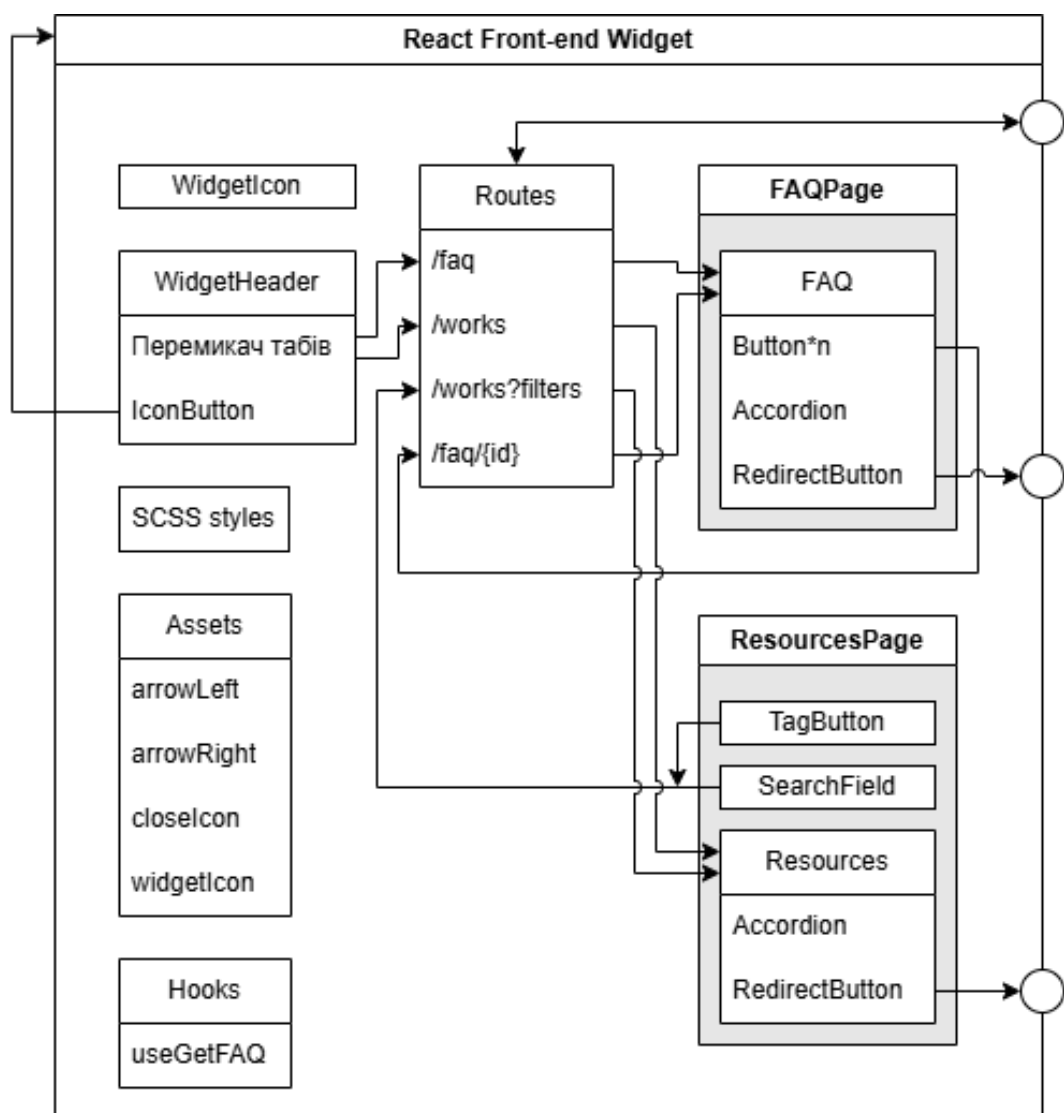


Рисунок 3.7 – Діаграма компонентів React віджету користувацького інтерфейсу

Логіка маршрутизації наступна.

Маршрути побудовані за допомогою бібліотеки react-router-dom:

- /faq – головне FAQ меню;
- /faq/{id} – вкладені сторінки FAQ для дочірніх кнопок;
- /works – інтерфейс перегляду академічних матеріалів;
- /works?filters – реалізація пошуку з параметрами фільтрації (теги, ключове слово).

Кастомні хуки: useGetFAQ – хук для асинхронного отримання структури FAQ з API. Кешується за допомогою React Query, що забезпечує продуктивність та плавність взаємодії.

Графіка та стилізація: весь інтерфейс стилізовано з використанням SCSS, що дозволило гнучко налаштовувати кольорову палітру та відступи, відповідно до дизайну. Візуальні елементи, такі як стрілки, іконки закриття, значки меню, зберігаються у папці assets, і використовуються як SVG або PNG-графіка для збереження якості.

Додаткові логічні компоненти:

- перемикач табів (FAQ / Академічні ресурси) дозволяє користувачу обирати тип контенту в межах одного вікна;
- перемикач мови (UA/EN) реалізовано у шапці віджету – значення мови автоматично підтягується з веб-порталу, але може бути змінено користувачем вручну;
- React Front-end Widget – кореневий контейнер, який об'єднує усі зазначені компоненти в єдину структуру, доступну з будь-якої сторінки сайту.

Такий підхід дозволив ефективно відтворити інтерфейс віджету відповідно до поставленого завдання, забезпечивши користувачам інтуїтивну навігацію, зручний пошук інформації та логічну структуру подання матеріалів.

Детальний опис розроблених компонентів можна подивитись у таблиці 3.8. Код розроблених копонентів надано у підрозділі Д.2 Додатку Д.

Таблиця 3.8 – Опис компонентів

№	Назва компонента	Метод	Призначення
1	Index.html	<code><script type="module" src="/src/main.jsx"></code>	Звичайна html сторінка що викликає запуск віджету через скрипт main.jsx.
2	Main	render()	Заміняє елемент сторінки з id #root на віджет App.jsx.
3	App	App()	Повертає віджет та відслідковує зміни в ньому.
4	Widget	Widget()	Відмальовує рамку меню з перемикачем мов та табів.
		toggleModal()	Відстежує перемикання табу.
		toggleLanguage()	Відстежує перемикання мов.
5	WidgetHeader	WidgetHeader()	Створює логіку обробки натисків в залежності від стану Widget.
		handleTabChange()	Замінює маршрутизацію в залежності від обраної локалізації.
6	Faq	fetchTopMenus()	Завантажує головне меню в залежності від обраної локалізації (українська за замовченням).
		Faq()	Відмальовує кнопки в головному меню, отримані з бек-енду через fetchTopMenus().
7	FaqDesc	fetchChildMenus()	Завантажує дочірні підменю за ід батьківської кнопки, та занчення локалізації.
		FaqDesc()	Виводить отримані об'єкти у вигляді кнопок.
8	Resources	useSearchParams()	При зміні запиту оновлює зміст меню
		Resources()	Виводить меню академ ресурсів з селектором тегів та полем вводу
9	Cbutton	Cbutton()	Створює елемент кнопки, з налаштованим гіперпосиланням
10	Accordion	Accordion()	Створює елемент аккордиону з логікою розвертання та сховання інформації

3.3 Розробка адмінпанелі

3.3.1 Розробка дизайну

Першим етапом створення адміністративної панелі стала побудова зручної навігації, орієнтованої на функціональність, визначену у технічному завданні. Оскільки адмінпанель має забезпечувати авторизацію користувача

та виконання CRUD-операцій із базою даних, інтерфейс містить форми з текстовими полями, списками та таблицями записів.

Дизайн було реалізовано у світлій темі, відповідно до візуального стилю основної частини вебпорталу. Для пришвидшення розробки був використаний шаблон адмінпанелі, стилістично наближений до WordPress, що дозволило зосередитися на функціональності без втрати естетики.

Головне меню адміністративної частини складається зі списку записів у базі даних, розділених за двома категоріями: FAQ-кнопки та Академічні ресурси. До кожного запису доступні кнопки редагування, додавання та видалення, які ведуть до відповідних форм введення або підтвердження дії. Усі інтерактивні елементи виконані в зеленій кольоровій палітрі, що підкреслює екологічний характер розробленого чатбота.

Результати проєктування інтерфейсу адміністративної панелі представлено на рисунках 3.8 – 3.10. Таким чином на основі шаблонів 3.8 – 3.10 можна почати верстати адмінпанель, додавати JavaScript та прописувати маршрутизацію.

Вхід в систему
адміністрування

Введіть свою електронну адресу та
пароль, щоб отримати доступ до свого
облікового запису.

Електронна пошта

vashaposhta@gmail.com

Пароль

Увійти

Рисунок 3.8 – Форма авторизації в адмінпанель

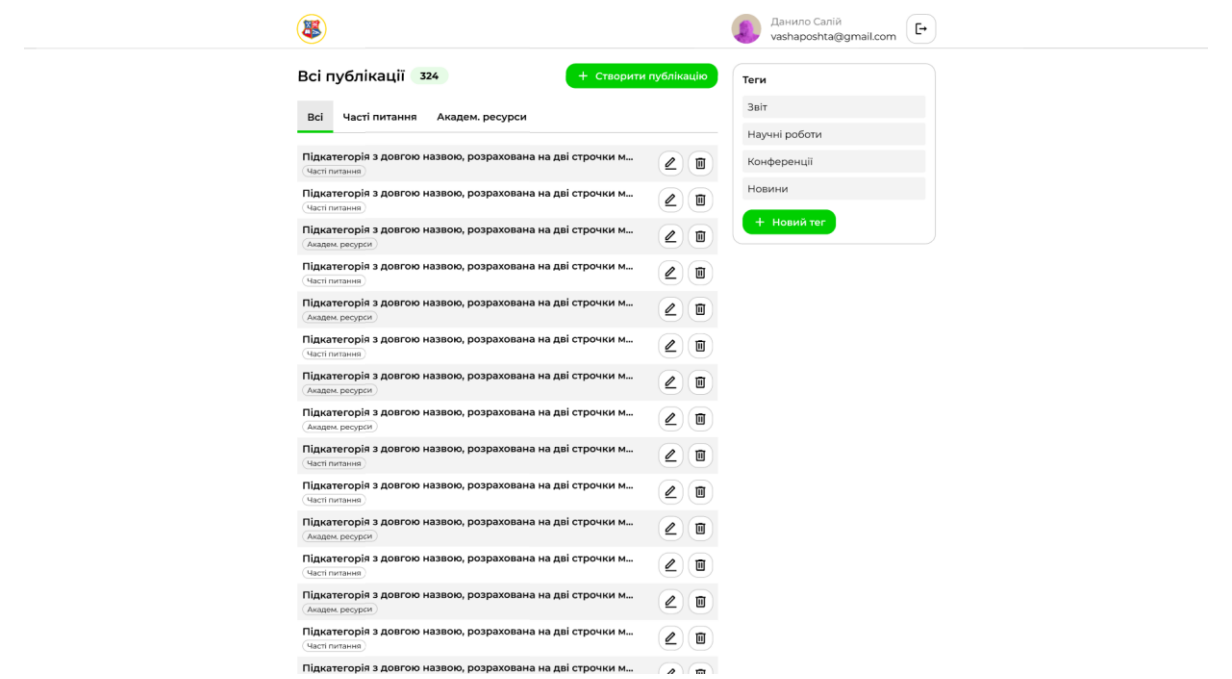


Рисунок 3.9 – Головне меню адмінпанелі

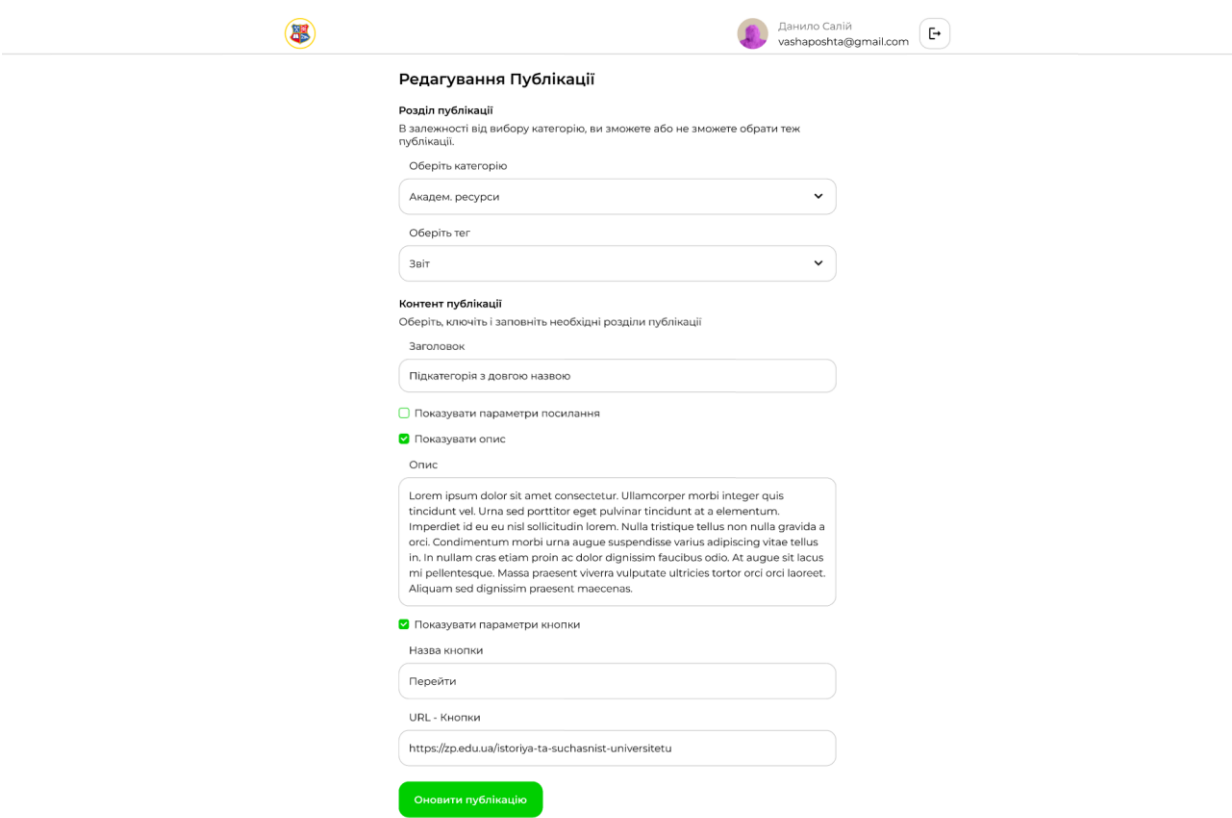


Рисунок 3.10 – Форма редагування запису у БД в адмін панелі

3.3.2 Відтворення дизайну адмінпанелі на blade шаблонах

Після затвердження структури та зовнішнього вигляду адмінпанелі другим етапом стало безпосереднє втілення дизайну у вигляді Blade-шаблонів у фреймворку Laravel. Blade — це потужний шаблонізатор, вбудований у Laravel, який дозволяє ефективно відокремлювати логіку від відображення, що сприяє чистішому та структурованішому коду.

Кожна сторінка адмінпанелі була реалізована як окремий шаблон або компонент, що розширює базовий макет (layout). Цей базовий шаблон містить спільні елементи інтерфейсу — хедер, підключення стилів та скриптів. Основна частина контенту підставляється динамічно, залежно від активного розділу.

Для забезпечення зручності й гнучкості, форми редагування та створення записів реалізовані із застосуванням валідації на клієнтській та серверній стороні. Всі поля форм відповідають структурі бази даних та дозволяють змінювати мультимовний контент, посилання, категорії тощо. Дані, отримані через API, відображаються у таблицях за допомогою циклів Blade (@foreach), а кнопки дій (редагування, видалення) підключають відповідні маршрути з використанням CSRF-захисту.

Окрему увагу приділено відповідності дизайну: кольорова гама, відступи, шрифти та інші стилістичні елементи ретельно адаптовані до HTML-структури шаблонів для забезпечення візуальної єдності з початковим дизайном.

Таким чином, завдяки використанню Blade-шаблонів вдалося досягти високої гнучкості, повторного використання коду та швидкого оновлення інтерфейсу без дублювання логіки. Діаграму компонентів отриманого інтерфейсу адмінпанелі можна побачити на рис. 3.11.

Детальний опис Blade-шаблонів можна подивитися у таблиці 3.9. Код шаблонів (html + php логіка) можна побачити у підрозділі Д.3 додатку Д.

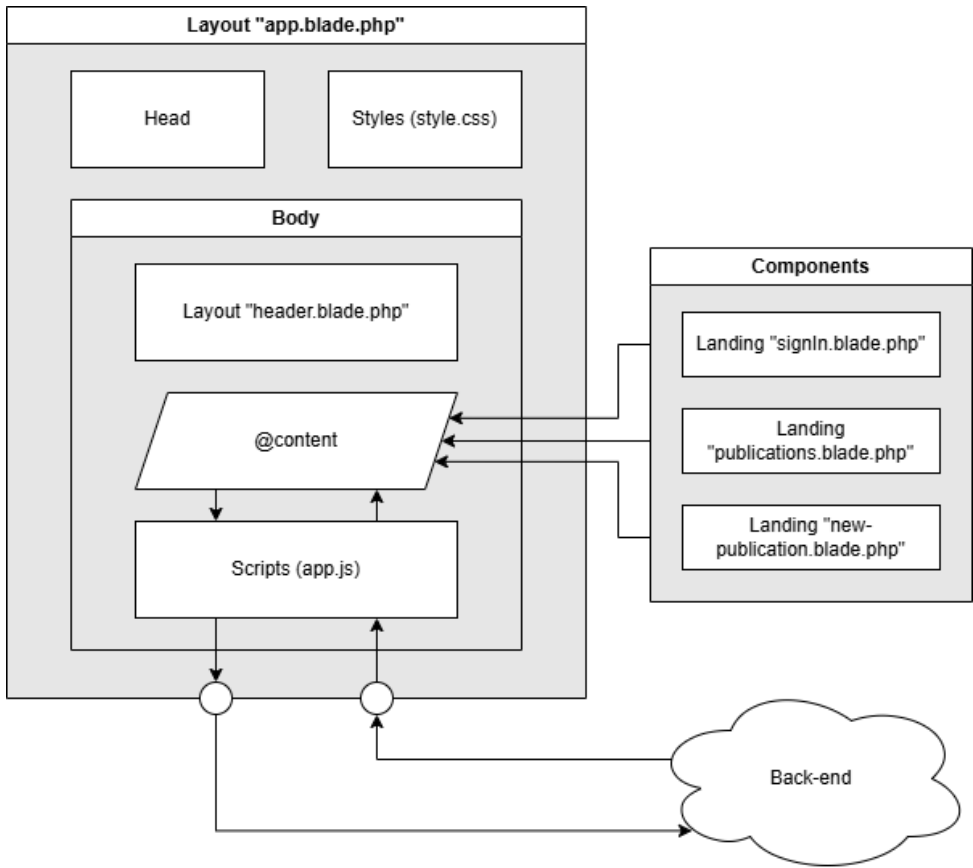


Рисунок 3.11 – Діаграма компонентів адмінпанелі

Таблиця 3.9 – Опис blade-шаблонів

№	Назва шаблону	Призначення
1	2	3
1	Layout.app.blade	Містить head, підключає скрипти, header, зміст та csrf_token
2	Layout.header.blade	Містить header та кнопку виходу з акаунту.
3	Landing.signIn.blade	Містить сторінку авторизації з полями та кнопкою входу (без header)
4	Buttons.table.blade	Містить блок що виводить записи кнопок меню з БД у вигляді списку з кнопками управління до кожного об'єкту.
5	Buttons.create.blade	Містить форму для створення нової кнопки меню Faq
6	Buttons.edit.blade	Містить форму для редагування певної кнопки меню Faq
7	Buttons.table.blade	Містить блок що виводить список академ. ресурсів у вигляді списку з кнопками управління до кожного об'єкту.
8	Buttons.create.blade	Містить форму для створення нової кнопки меню Faq
9	Buttons.edit.blade	Містить форму для редагування певної кнопки меню Faq

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

Після завершення розробки чатбота для інформування про екологічну політику університету було проведено всебічне тестування, яке мало на меті перевірити відповідність системи вимогам технічного завдання, виявити можливі помилки, забезпечити стабільність роботи та зручність взаємодії з кінцевими користувачами.

Загальну процедуру тестування було поділено на три ключові етапи:

- тестування API до бекенду – первинне тестування функціональної основи чатбота, що охоплює обробку HTTP-запитів, автентифікацію, обробку помилок, структуру відповідей, формат переданих даних тощо. Цей етап є критичним, оскільки саме через API фронтенд отримує доступ до інформації та логіки обробки запитів;
- тестування користувацького інтерфейсу – проводиться після підтвердження стабільної роботи API. На цьому етапі перевіряється відображення елементів інтерфейсу, перемикання між вкладками, багатомовність, зручність навігації, а також відповідність візуального оформлення принципам UI/UX;
- тестування адміністративної панелі – окремий напрям перевірки, спрямований на функціональність інструментів керування вмістом, тегами, мовами, а також доступністю функцій, зарезервованих для адміністратора системи.

4.1 Тестування API до бекенду

Бекенд чатбота реалізовано з використанням фреймворку Laravel. Основна функція серверної частини — забезпечити надійну логіку обробки запитів, доступ до бази даних, автентифікацію, управління контентом та підтримку багатомовності.

Тестування API проводилося шляхом надсилення HTTP-запитів до окремих маршрутів (endpoint'ів) з метою перевірки:

- відповідності структури запитів та відповідей;
- коректності обробки параметрів;
- стабільності роботи при типових і граничних навантаженнях;
- обробки помилок та відсутності критичних вразливостей.

Для тестування використовувалися інструменти Postman та Laravel HTTP tests. Результати тестування можна подивитися у таблиці 4.1.

Таблиця 4.1 – Позитивні тест-кейси для API до бекенду

№	Назва тесту	Метод / Маршрут	Вхідні дані	Очікуваний результат	Статус
Публічні API без необхідності аутентифікації (bearer токenu)					
1	2	3	4	5	6
1	Отримання верхнього рівня меню FAQ	GET /api/faq	—	200 ОК, масив об'єктів кнопок з назвою, описом двома мовами, та посиланням на джерело.	Успіх
2	Отримання верхнього рівню меню FAQ обраної локалізації	GET /api/faq?locale=en	locale = en/uk.	200 ОК, масив об'єктів кнопок з назвою, описом обраною мовою, та посиланням на джерело.	Успіх
3	Отримання підменю FAQ за ID батьківської кнопки в обраній локалізації	GET /api/faq/3?locale=en	id = int; locale = en/uk.	200 ОК, масив об'єктів кнопок з відповідним parent_id з назвою, описом обраною мовою та посиланням на джерело.	Успіх
4	Отримання списку категорій на двох мовах	GET /api/tags	—	200 ОК, масив об'єктів тегів з назвою двома мовами та значенням для пошуку в БД.	Успіх
5	Отримання повного списку академ. ресурсів без фільтрів	GET /api/works	—	200 ОК, масив об'єктів академ. ресурсів з повним описом на двох мовах та посиланням на першоджерело.	Успіх

Продовження таблиці 4.1

1	2	3	4	5	6
6	Отримання відфільтрованого списку академ. ресурсів з фільтрами	GET /api/works ?tag= &locale= &per_page= &search=Аналіз	tag=1-4; locale=en/uk; per_page=int; search=string.	200 OK, масив об'єктів академ. ресурсів, що відповідають фільтрам з описом на обраній мові та посиланням на першоджерело	Успіх
Закриті API з необхідністю аутентифікації (адмін панель)					
8	Авторизація в адмін панель	POST api/login	email:string; password:string; remember:bool.	200 OK, унікальний Bearer токен для кожної сесії авторизації	Успіх
9	Завершення сеансу адмін панелі	POST api/logout	—	200 OK, «Logged out».	Успіх
10	Створення нової кнопки для меню FAQ	POST api/buttons	ua_title:string; en_title:string; ua_description:string; en_description:string; url:string; parent_id:int.	200 OK, Створений об'єкт з повним описом полів, збережених у БД.	Успіх
11	Оновлення даних у вже створеній кнопці меню FAQ	PUT api/buttons/{id}	ua_title:string; en_title:string; ua_description:string; en_description:string; url:string; parent_id:int	200 OK, Оновлений об'єкт з повним описом полів, збережених у БД.	Успіх
12	Видалення кнопки меню FAQ з БД	DELETE api/buttons/{id}	—	200 OK, «Button deleted successfully»	Успіх
13	Отримати дані про певну кнопку меню FAQ	GET /api/buttons/{id}	—	200 OK, Об'єкт кнопки за ID з повним описом полів збережених у БД.	Успіх
14	Створення нового запису про «Академ. ресурс» в БД	POST api/works	tag:1-4; ua_title:string; en_title:string; ua_description:string; en_description:string; url:string;	200 OK, Створений об'єкт «Академ. ресурсу» з повним описом полів, збережених до БД.	Успіх

Кінець таблиці 4.1

1	2	3	4	5	6
15	Оновлення даних про «Академ. ресурс» в БД	PUT api/works/{id}	tag:1-4; ua_title:string; en_title:string; ua_description:string; en_description:string; url:string;	200 OK, Оновлений об'єкт «Академ. ресурсу» з повним описом полів, збережених у БД.	Успіх
16	Видалення запису про «Академ. ресурс» з БД	DELETE api/works/{id}	—	200 OK, «Work deleted successfully»	Успіх
17	Отримати об'єкт певного запису «Академ. ресурсу» а БД	GET /api/works/{id}	—	200 OK, Об'єкт «Академ. ресурсу» за ID з повним описом полів збережених у БД.	Успіх

Також результати тестування наведено у таблиці 4.2.

Таблиця 4.2 – Негативні тест-кейси для API до бекенду

№	Назва тесту	Метод / Маршрут	Вхідні дані	Очікуваний результат	Статус
Публічні API без необхідності аутентифікації (bearer токена)					
1	2	3	4	5	6
1	Звернення до неіснуючого API	GET /api/unknown	—	404 Not Found, "The route api/unknown could not be found.".	Успіх
2	Отримання верхнього рівню меню FAQ обраної локалізації з незапланованими даними	GET /api/faq ?locale=en &tag=tag &password=123	locale = en/uk; tag=<tag>; password=123.	200 OK, масив об'єктів кнопок з назвою, описом обраною мовою, та посиланням на джерело. Надлишкові дані в GET запиті не заважають роботі API.	Успіх
3	Отримання підменю FAQ за ID батьківської кнопки в обраній локалізації, але кнопки з вказаним ID не існує в БД	GET /api/faq/-1?locale=en	id = -1; locale = en/uk.	404 Not Found, "No query results for model [App\\Models\\Button] -1"	Успіх

Кінець таблиці 4.2

1	2	3	4	5	6
4	Отримання підменю FAQ за ID батьківської кнопки в непередбаченій локалізації	GET /api/faq/1?locale=fr	locale = fr.	422 Unprocessable Content, "The selected locale is invalid."	Успіх
5	Звернення до API помилковим методом	POST /api/tags	—	405 Method Not Allowed, "The POST method is not supported for route api/tags. Supported methods: GET, HEAD."	Успіх
6	Звернення до закритих API без авторизації у адмін панелі	DELETE api/buttons/{id}	—	401 Unauthorized, "Unauthenticated."	Успіх
	Звернення на POST маршрут вказавши в валідне поле невалідні дані	POST api/login	email:123; password:[]; remember:text.	422 Unprocessable Content, "The email must be a valid email address. (and 2 more errors)"	Успіх

Як показано у таблицях 4.1 та 4.2, усі оголошені API було протестовано як у позитивних, так і в негативних сценаріях. Бекенд коректно обробляє HTTP-запити, належно взаємодіє з базою даних, не допускаючи внутрішніх помилок при зверненнях до неї.

Функціональність API чітко поділено на публічну та адміністративну: публічні маршрути не вимагають автентифікації, тоді як усі дії з адміністрування (додавання, редагування, видалення контенту) доступні виключно після проходження автентифікації через Bearer токен. Бекенд також коректно обробляє помилки, повертаючи лише короткі інформативні повідомлення, без розкриття внутрішньої структури проєкту, що підвищує безпеку системи.

Крім того, передача авторизаційних та адміністративних даних здійснюється виключно через API-запити з захищеним каналом, що мінімізує

ризик їхнього перехоплення. Таким чином, можна зробити висновок, що основа проекту — серверна логіка — працює надійно, стабільно та безпечно.

Після успішного тестування фундаментальної частини системи можна переходити до перевірки зовнішніх інтерфейсів: користувацького інтерфейсу та адміністративної панелі. Наступним кроком є перевірка їхньої коректної роботи, зручності використання, реакції на дії користувача та взаємодії з API.

4.2 Тестування користувацького інтерфейсу

Після підтвердження стабільної та безпомилкової роботи серверної частини чатбота, наступним кроком є перевірка користувацького інтерфейсу. Основна мета цього етапу — впевнитися, що фронтенд правильно підключений до API, коректно відображає дані, зручно працює на різних пристроях і підтримує функції, зазначені в технічному завданні.

Користувацький інтерфейс чатбота реалізовано як інтерактивний віджет, вбудований у вебпортал університету. Він має вкладки «FAQ» та «Академічні ресурси», підтримує дві мови (українську та англійську), а також пошук за тегами та ключовими словами. Тестування охоплює як функціональні аспекти (відображення кнопок, переходи, результати пошуку), так і UX-фактори (відповідність дизайну, адаптивність, інтуїтивність навігації).

Оскільки бекенд пройшов повне тестування і забезпечує стабільну взаємодію з базою даних, інтерфейсна частина мала змогу будуватися на надійному фундаменті. Це дозволило сконцентруватися на перевірці саме візуальної логіки, коректності відображення інформації та взаємодії користувача з чатботом.

Нижче, в таблицях 4.3 та 4.4, наведено ключові тест-кейси, що були виконані для перевірки функціональності користувацького інтерфейсу.

Таблиця 4.3 – Позитивні тест-кейси для користувацького інтерфейсу

№	Назва тесту	Тип тестування	Вхідні умови / Дії користувача	Очікуваний результат	Статус
1	Відображення віджета на головній сторінці	UI	Завантаження головної сторінки університетського порталу	Віджет чатбота з'являється у правому нижньому куті екрана	Успіх
2	Перемикання мови інтерфейсу	UX / Функціональний	Натиснути кнопку перемикання мови	Усі тексти інтерфейсу (кнопки, описи) змінюються відповідно до вибраної мови	Успіх
3	Завантаження FAQ при відкритті вкладки	Функціональний	Клік по вкладці "FAQ"	Виводиться список категорій верхнього рівня з відповідними назвами й описами	Успіх
4	Завантаження підменю FAQ	Функціональний	Клік по кнопці з підкатегоріями	Виводиться список відповідних підпунктів, дані отримано з API	Успіх
5	Пошук за ключовими словами	UX / Функціональний	Увести слово в пошукове поле у вкладці "Академічні ресурси"	Виводиться список релевантних результатів згідно з запитом	Успіх
6	Відображення результатів фільтрації за тегом	Функціональний	Обрати тег у фільтрі	Виводяться лише ресурси, що відповідають вибраній категорії	Успіх
7	Відкриття посилання на першоджерело	Функціональний	Натиснути на посилання у картці ресурсу	Відкривається відповідна сторінка у новій вкладці браузера	Успіх
8	Адаптивність для мобільних пристроїв	UI / UX	Відкрити сайт на екрані смартфона	Віджет адаптовано до мобільного формату, елементи не виходять за межі екрана	Успіх
9	Обробка запиту без інтернет-з'єднання	UX / Функціональний	Відкрити чатбот і вимкнути інтернет перед кліком	Запити не надсилаються. Поля з результатами пусті	Успіх

Таблиця 4.4 – Негативні тест-кейси для користувацького інтерфейсу

№	Назва тесту	Тип тестування	Вхідні умови / Дії користувача	Очікуваний результат	Статус
1	Пошук за пустим рядком	Функціональний	Залишити поле пошуку порожнім	Виводяться усі ресурси обраної категорії	Успіх
2	Неправильне значення параметра locale	Функціональний	Передати параметр locale=de через URL	Поле з результатами пошуку пусте. В лозі виведено помилку про непідтримувану локалізацію	Успіх
3	Спроба доступу до ресурсу без інтернету	UX	Вимкнути мережу та клікнути на вкладку «Академічні ресурси»	Відкривається вкладка «Академічні ресурси» але пошук не працює	Успіх
4	Відсутність результатів пошуку	UX / Функціональний	Увести в поле пошуку слово, якого немає в базі	Виводиться повідомлення «Нічого не знайдено»	Успіх
5	Некоректна відповідь від API (імітація 500 помилки)	Функціональний	Імітувати внутрішню помилку сервера	Інтерфейс відображає загальне повідомлення про технічну помилку, без деталей реалізації бекенду	Успіх

Як показано в таблицях 4.3 та 4.4, функціональність користувацького інтерфейсу була всебічно протестована як у позитивних, так і в негативних сценаріях. Перевірено стабільність відображення віджета на різних пристроях, працездатність мовного перемикача, роботу пошуку, фільтрації, взаємодії з API та реакцію системи на типові помилки користувача або сервера. Інтерфейс виявився стабільним, адаптивним і стійким до збоїв.

4.3 Тестування адмінпанелі

Тепер, коли користувацький інтерфейс підтвердив свою надійність, можна перейти до тестування адміністративної частини, яка відповідає за редагування даних у базі. На цьому етапі важливо переконатися, що форма введення, API-запити та обробка відповіді працюють коректно, зміни

зберігаються без помилок, а сама система забезпечує захищений і зручний спосіб адміністрування вмісту чатбота.

У межах цього етапу було проведено як функціональне, так і негативне тестування адміністративної панелі для перевірки її коректної роботи. Основна увага приділялася стабільності авторизації, взаємодії з формами створення, редагування та видалення записів у базі даних, а також захищеності запитів до закритих маршрутів API.

Тестування охоплює як модуль керування пунктами меню FAQ, так і управління академічними ресурсами. Таблиця 4.5 демонструє основні позитивні сценарії взаємодії адміністратора з панеллю керування, включно з перевіркою створення, редагування та видалення записів.

Таблиця 4.5 – Позитивні тест-кейси для інтерфейсу адмінпанелі

№	Назва тесту	Елемент інтерфейсу	Дія адміністратора	Очікувана реакція інтерфейсу	Статус
1	2	3	4	5	6
1	Авторизація в адмінпанель	Форма входу	Ввести правильні email та пароль, натиснути "Увійти"	Переадресація на головну сторінку адмін-панелі, збереження токена	Успіх
2	Вихід з адмінпанелі	Кнопка "Вийти"	Натиснути кнопку	Повернення на сторінку входу, токен видалено з локального сховища	Успіх
3	Створення пункту FAQ	Форма додавання кнопки	Заповнити всі поля, натиснути "Створити"	Після відповіді 200 OK — оновлення списку, повідомлення "Успішно створено"	Успіх
4	Редагування пункту FAQ	Форма редагування запису	Змінити назву/опис, натиснути "Зберегти"	Дані оновлюються в таблиці, спливає повідомлення про успіх	Успіх
5	Видалення пункту FAQ	Кнопка "Видалити"	Натиснути на обраний запис	Запит підтвердження → після підтвердження — запис зникає, повідомлення "Видалено"	Успіх
6	Прегляд окремого запису FAQ	Кнопка/іконка перегляду	Натиснути на запис у таблиці	Відкривається модальне вікно з повною інформацією	Успіх

Кінець таблиці 4.5

1	2	3	4	5	6
7	Створення академічного ресурсу	Форма створення ресурсу	Заповнити дані, вибрати тег, натиснути "Додати"	Після відповіді API — ресурс з'являється у списку, повідомлення про успішну дію	Успіх
8	Редагування ресурсу	Форма редагування	Змінити опис/посилання, натиснути "Зберегти"	Оновлені дані відображаються без перезавантаження сторінки	Успіх
9	Видалення ресурсу	Кнопка "Видалити" біля запису	Підтвердити видалення	Запис видаляється з таблиці, виводиться повідомлення	Успіх
10	Відображення повідомлень про помилки	Будь-яка форма	Не заповнити обов'язкове поле або ввести некоректне значення	Виводиться інформативне повідомлення під відповідним полем	Успіх

У таблиці 4.6 наведено негативні сценарії, що дозволили переконатися у правильному обробленні помилкових введеннь, відсутності несанкціонованого доступу та стабільності поведінки системи при неправильному використанні.

Таблиця 4.6 – Негативні тест-кейси для інтерфейсу адмінпанелі

№	Назва тесту	Елемент інтерфейсу	Дія адміністратора	Очікувана реакція інтерфейсу	Статус
1	2	3	4	5	6
1	Помилковий логін	Форма входу	Ввести неправильний email або пароль	Виведення повідомлення про помилку авторизації, залишення на тій же сторінці	Успіх
2	Порожні обов'язкові поля	Форма створення або редагування	Залишити одне чи більше обов'язкових полів порожніми	Підсвічування порожніх полів, повідомлення про обов'язковість заповнення	Успіх
3	Невалідне посилання	Поле URL у формі	Ввести некоректний URL (наприклад, без http/https)	Виведення повідомлення "Некоректне посилання", запит не надсилається	Успіх

Кінець таблиці 4.6

1	2	3	4	5	6
4	Невалідний ID при редагуванні	Форма редагування	Спроба перейти на сторінку редагування неіснуючого запису (ID не знайдено)	Повідомлення "Запис не знайдено"	Успіх
5	Невалідне значення поля "parent_id"	Форма створення FAQ-кнопки	Ввести символи або ID, якого не існує в БД	API повертає помилку, інтерфейс відображає її, форма не зберігається	Успіх
6	Видалення неіснуючого запису	Кнопка "Видалити" (штучна ситуація)	Спроба видалити запис, якого вже немає в базі	API повертає помилку, інтерфейс показує повідомлення про невдачу	Успіх
7	Недоступність сервера	Будь-яка дія, що вимагає API-запиту	Вимкнено сервер або API недоступне	Виведення повідомлення "Немає зв'язку з сервером", без краху інтерфейсу	Успіх

Тестування адміністративної панелі показало, що інтерфейс ефективно виконує свої функції з редагування даних у базі, коректно формує та надсилає API-запити, а також обробляє відповіді без помилок. Система реагує на помилкові дії адміністратора зрозумілими повідомленнями, що забезпечує зручність та безпеку при роботі з контентом чатбота. Таким чином, адміністративна частина повністю відповідає вимогам до стабільної та захищеної системи керування даними.

4.4 Оцінка тестування

На основі проведених тестувань можна впевнено стверджувати, що розроблена система є повністю функціональною, стабільною та готовою до впровадження. Усі компоненти — серверна частина (бек-енд), користувацький інтерфейс, а також адміністративна панель — пройшли комплексне тестування як за позитивними, так і за негативними сценаріями.

Бек-енд забезпечує надійну обробку запитів, реалізує механізми автентифікації та захисту, не допускає витоку службової інформації та правильно обробляє помилки. API відповідають вимогам специфікації, повертають коректні дані у відповідному форматі та гарантують відсутність внутрішніх збоїв.

Користувацький інтерфейс демонструє стабільну роботу, правильно взаємодіє з API, відображає дані згідно з обраною мовою та фільтрами, а також реагує на нештатні ситуації зрозумілими повідомленнями. Адміністративна панель надає зручні засоби управління контентом без помилок, коректно формує запити до бек-енду та має адекватну систему валідації.

Таким чином, усі три компоненти системи відповідають функціональним та нефункціональним вимогам, реалізовані згідно з технічним завданням і готові до використання кінцевими користувачами та адміністраторами.

ВИСНОВКИ

У межах кваліфікаційної роботи було реалізовано повний цикл створення інформаційного модуля — чатбота, інтегрованого у вебпортал університету, що інформує про екологічну політику та ініціативи закладу. Поставлена мета — розробити інноваційний інструмент для покращення доступності екологічної інформації та формування сталого іміджу ЗВО — досягнута в повному обсязі.

У результаті проведеного дослідження було встановлено, що провідні університети з високими позиціями в рейтингу UI GreenMetric демонструють відкритість та системний підхід до екологічної комунікації, тоді як українські заклади часто не мають структурованої та візуально вираженої стратегії екологічної представленості. Було сформульовано перелік ключових інформаційних блоків, які мають бути інтегровані у вебпортал та чатбот, включаючи розділи про сталий розвиток, екологічні досягнення, позицію в рейтингах тощо.

Дослідження світових практик використання ШІ показало, що попри широку популярність чатботів у комерційному секторі, більшість університетів ще не використовують їх для екологічної комунікації. Це створює можливість позиціювання нашого університету як новатора, здатного інтегрувати сучасні IT-рішення у сфері сталого розвитку.

Для реалізації проєкту було обрано оптимальний технологічний стек: Laravel для бек-енду, MySQL для зберігання даних, React для побудови клієнтського інтерфейсу та Blade для шаблонізації в адмінпанелі. Такий підхід забезпечив ефективне розділення логіки, масштабованість, повторне використання коду та зручність подальшої підтримки системи. Реалізовано як фронтенд-інтерфейс чатбота, так і адміністративну панель для керування вмістом і мовними налаштуваннями.

Система була протестована за позитивними та негативними сценаріями, і результати тестування підтвердили її готовність до реального

використання. Усі компоненти — API, інтерфейс користувача та адмінпанель — продемонстрували стабільну роботу, відповідність технічному завданню та високу якість реалізації.

Таким чином, запропонований програмний модуль чатбота є ефективним, гнучким і сучасним інструментом, що сприятиме підвищенню рівня відкритості університету, популяризації екологічної політики та формуванню позитивного іміджу у глобальному інформаційному середовищі.

Результати роботи було представлено на Щорічній науково-практичній конференції викладачів, науковців, молодих учених, аспірантів та студентів Національного університету «Запорізька політехніка» – «Тиждень науки-2025».

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Green University Networks [Electronic resource]. – Access mode: <https://www.unep.org/explore-topics/education-environment/why-does-education-and-environment-matter/green-university>.
2. Штучний інтелект [Електрон. ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Штучний_інтелект.
3. UI GreenMetric [Electronic resource]. – Access mode: https://en.wikipedia.org/wiki/UI_GreenMetric.
4. Transforming our world: the 2030 Agenda for Sustainable Development [Electronic resource]. – Access mode: <https://sdgs.un.org/2030agenda>.
5. Витрачай зараз – заощаджуй у майбутньому. Кліматична політика ЄС [Електрон. ресурс]. – Режим доступу: <https://ecoaction.org.ua/zaoshchadzhuji-u-majbutnomu.html>.
6. European Green Deal [Electronic resource]. – Access mode: https://en.wikipedia.org/wiki/European_Green_Deal.
7. Потенціал фінансування ініціатив зеленого курсу в Україні [Електрон. ресурс]. – Режим доступу: <https://ir.kneu.edu.ua/server/api/core/bitstreams/a56a8215-3268-4d1a-9a76-a8e31e2fe238/content>.
8. Green Transition in Ukrainian Universities, NAWA project 2023 [Electronic resource]. – Access mode: <https://nure.ua/wp-content/uploads/gtua-nawa-project-introduction-.pdf>.
9. U.S. EPA Bipartisan Infrastructure Law & Inflation Reduction Act Awards & Selections [Electronic resource]. – Access mode: <https://www.epa.gov/invest/investments-epa-under-bipartisan-infrastructure-law-and-inflation-reduction-act>.
10. Website communications for campus sustainability: an analysis of Canadian universities [Electronic resource]. – Access mode: <https://www.emerald.com/insight/content/doi/10.1108/ijshe-04-2019->

0137/full/html.

11. Сторінка репозиторію НЛТУ на тему “Екологія” [Електрон. ресурс]. – Режим доступу: <https://www.oa.edu.ua/ua/science/sjournal/>.

12. Марушинець, В. В. Оцінка ландшафтної та екосистемної репрезентативності Карпатського біосферного заповідника / В. В. Марушинець, М. В. Чернявський. – Львів – 2024 – 58 с.

13. Wageningen University & Research [Electronic resource]. – Access mode: <https://www.wur.nl/>.

14. Міжнародна наукова конференція стратегії сталого розвитку в архітектурі та дизайні Національного університету "Львівська політехніка" ” [Електрон. ресурс]. – Режим доступу: <https://lpnu.ua/events/mizhnarodna-naukova-konferentsiia-stratehii-staloho-v-arkhitekturi-ta-dyzaini>.

15. Can Science Save California Tomatoes From Invasive Weeds? [Electronic resource] / University of California, Davis. – Access mode: <https://www.ucdavis.edu/news/research>.

16. University of Nottingham [Electronic resource]. – Access mode: <https://www.ntu.ac.uk/research>.

17. Biswas, J. K. Building a green image: best sustainability practices in universities: [Electronic resource] / J. K. Biswas, M. Dash. – Access mode: https://www.academia.edu/49009241/BUILDING_A_GREEN_IMAGE_BEST_SUSTAINABILITY_PRACTICES_IN_UNIVERSITIES.

18. Стратегія розвитку Національного університету «Запорізька політехніка» на період 2023 - 2027 років [Електрон. ресурс]. – Режим доступу: https://zp.edu.ua/uploads/pubdocs/2025/Nakaz_N61_vid_24.02.25.pdf.

19. White Paper for Universities on Navigating Artificial Intelligence Innovation Ecosystems in AI for Sustainable Development [Electronic resource]. – Access mode: https://asef.org/wp-content/uploads/2025/02/250219_White-Paper-AI-for-Sustainable-Development_FINAL.pdf.

20. Дячук, П. П. Використання геоінформаційних систем і штучного інтелекту для захисту особливо цінних лісів України: огляд можливостей,

виклики та правове забезпечення. / П. П. Дячук, І. В. Берднік // WWF-Україна. – Київ. 2024. – С. 1-36.

21. Як ШІ допомагає вам оптимізувати споживання енергії [Електрон. ресурс]. – Режим доступу: <https://www.ariston.com/ua-ua/the-comfort-way/zberihannia-enerhii/yak-shi-optymizovuie-spozhyvannia/>.

22. Суходоля, О. М. Штучний інтелект в енергетиці / О. М. Суходоля // Аналітична доповідь. – Київ. – 2022. – С.1-49.

23. Штучний інтелект як інструмент забезпечення екологічної безпеки та сталого розвитку / [А. В. Череп, В. Г. Воронкова, О. Г. Череп та ін.] // Ukrainian science and education in the conditions of European integration. – Київ. – 2023. – С. 1-16.

24. How artificial intelligence is helping tackle environmental challenges [Electronic resource]. – Access mode: <https://www.unep.org/news-and-stories/story/how-artificial-intelligence-helping-tackle-environmental-challenges>.

25. DeepMind and Google Train AI to Predict Energy Output of Wind Farms [Electronic resource]. – Access mode: <https://www.worldenergy.org.tr/deepmind-and-google-train-ai-to-predict-energy-output-of-wind-farms/>.

26. IBM Research Launches Project "Green Horizon" to Help China Deliver on Ambitious Energy and Environmental Goals [Electronic resource]. – Access mode: <https://www.prnewswire.com/news-releases/ibm-research-launches-project-green-horizon-to-help-china-deliver-on-ambitious-energy-and-environmental-goals-265983691.html>.

27. Google Environmental Insights Explorer [Electronic resource]. – Access mode: <https://www.cakex.org/tools/google-environmental-insights-explorer>.

28. Environmental Insights Explorer [Electronic resource]. – Access mode: <https://insights.sustainability.google/?hl=ru>.

29. Wildlife Insights by Conservation International [Electronic resource]. – Access mode: <https://www.vizzuality.com/project/wildlife-insights>.

30. Microsoft's AI for Earth Program [Electronic resource]. – Access mode: <https://www.triplepundit.com/story/2017/ai-earth-program-microsoft-support->

environmental-projects/732621.

31. New UC Berkeley initiative uses AI research to solve climate problems [Electronic resource]. – Access mode: <https://cdss.berkeley.edu/news/new-uc-berkeley-initiative-uses-ai-research-solve-climate-problems>.

32. Morell, N. Tackling the MIT campus's top energy consumers, building by building [Electronic resource]. – Access mode: <https://news.mit.edu/2023/tackling-mit-campus-top-energy-consumers-0525>.

33. Smart Garden Watering [Electronic resource]. – Access mode: <https://cis.unimelb.edu.au/hci/projects/sgw#info>.

34. AI predicts Earth's peak warming [Electronic resource]. – Access mode: <https://news.stanford.edu/stories/2024/12/ai-predicts-earths-peak-warming>.

35. Слупська, С. Ю. Аналіз особливостей побудови систем моніторингу стану довкілля з використанням методів штучного інтелекту / С. Ю. Слупська, // Атестаційна робота. – Харків. – 2020. – 77 с.

36. SaveEcoBot – Єдина в Україні екологічна система [Електрон. ресурс]. – Режим доступу: <https://www.saveecobot.com/#chatbot>.

37. ChatGPT Revolutionizes Conversations: Everything You Need to Know About OpenAI's Chatbot [Electronic resource]. – Access mode: <https://opentools.ai/news/chatgpt-revolutionizes-conversations-everything-you-need-to-know-about-openais-chatbot>.

38. Gemini – Features [Electronic resource]. – Access mode: <https://www.tutorialspoint.com/gemini/gemini-features.htm>.

39. Startduck agency [Electronic resource]. – Access mode: <https://startduck.com/ru/service/chat-bot-sistema-dlya-biznesa>.

40. Model–view–controller [Electronic resource]. – Access mode: <https://en.wikipedia.org/wiki/Model–view–controller>.

41. Кірієнко, А. С. Розробка вебпорталу для підтримки зеленої стратегії розвитку університету / А. С. Кірієнко, А. В. Пархоменко // Тиждень науки-2025: Щорічна НПК викладачів, науковців, молодих учених, аспірантів та студентів, Запоріжжя, 14-18 квітня 2025 р.: тези доповідей. – Запоріжжя:

HY3Π, 2025. – C. 370-371.

42. Model–view–presenter [Electronic resource]. – Access mode: <https://en.wikipedia.org/wiki/Model–view–presenter>.

43. Model–view–viewmodel [Electronic resource]. – Access mode: <https://en.wikipedia.org/wiki/Model–view–viewmodel>.

44. Microservices [Electronic resource]. – Access mode: <https://en.wikipedia.org/wiki/Microservices>.

45. Laravel [Electronic resource]. – Access mode: <https://en.wikipedia.org/wiki/Laravel>.

46. Django: The web framework for perfectionists with deadlines [Electronic resource]. – Access mode: <https://www.djangoproject.com>.

47. Express - Node.js web application framework [Electronic resource]. – Access mode: <https://expressjs.com>.

48. Spring Boot [Electronic resource]. – Access mode: <https://spring.io/projects/spring-boot>.

49. MySQL [Electronic resource]. – Access mode: <https://en.wikipedia.org/wiki/MySQL>.

50. PostgreSQL: The World's Most Advanced Open Source Relational Database [Electronic resource]. – Access mode: <https://www.postgresql.org>.

51. SQLite [Electronic resource]. – Access mode: <https://sqlite.org>.

52. MongoDB: Loved by developers. Built for AI [Electronic resource]. – Access mode: <https://www.mongodb.com>.

53. React: the library for web and native user interfaces [Electronic resource]. – Access mode: <https://ru.legacy.reactjs.org>.

54. Vue.js: The Progressive JavaScript Framework [Electronic resource]. – Access mode: <https://vuejs.org>.

55. Angular [Electronic resource]. – Access mode: <https://angular.dev>

56. jQuery: fast, small, and feature-rich JavaScript library. [Electronic resource]. – Access mode: <https://jquery.com>.

57. HTML, CSS, & JavaScript Explained with Analogies [Electronic

resource]. – Access mode: <https://tadabase.io/blog/html-css-javascript-explained>

58. Laravel Queues Documentation [Electronic resource]. – Access mode: <https://laravel.com/docs/12.x/queues>.

59. Laravel Horizon [Electronic resource]. – Access mode: <https://laravel.com/docs/11.x/horizon>.

ДОДАТОК А
Технічне завдання

Вступ

Метою роботи є розробка та впровадження чатбота, який забезпечує зручний доступ до екологічної інформації та допомагає формувати позитивний імідж університету як активного учасника зелених ініціатив.

На момент початку розробки вебпортал університету, вже містить інформаційні меню та стрічку новин, однак не має реалізованих швидких засобів доступу до звітної інформації, бази даних новин або матеріалів конференцій. На головній сторінці відсутній механізм навігації по структурованим даним або пошуку за темами. Пошук в репозитарії працює погано, та шукає матеріали тільки за повною назвою. Також наразі не використовується жодний сучасний інтерактивний засіб поширення новин, в тому числі чатбота. Опис поточного стану зроблено на основі домашньої сторінки вебпорталу НУЗП <https://zp.edu.ua> станом на 09.05.2025

Після інтеграції чатбот з'явиться у вигляді інтерактивної кнопки, розміщеної в правому нижньому куті сторінки. Натискання на неї відкриватиме інтерфейс із динамічними кнопками та пошуковим полем, за допомогою якого користувач зможе швидко отримати доступ до потрібної інформації – наприклад, переглянути останні екологічні новини, ознайомитись із розкладом подій, знайти звіти за ключовими словами тощо. Вся інформація буде підвантажуватись з бази даних, що оновлюється через окрему адміністративну панель.

A.1 Обсяг робіт

A.1.1 Вимоги до інтеграції

Інтеграція чатбота передбачає поєднання автономного модуля, з існуючим фронтендом екологічної вебсторінки університету, побудованим на React. Взаємодія між компонентами реалізується через REST API, що дозволяє

забезпечити чітке розділення клієнтської та серверної логіки. Основні вимоги до інтеграції включають:

- обсяг даних для передачі: текстова інформація (заголовки, описи, посилання, дати), що стосується екологічних новин, звітів, конференцій, ініціатив;
- формат даних: JSON (усі запити і відповіді між фронтендом і сервером);
- частота обміну: за запитом користувача (кожне натискання кнопки чи виконання пошуку ініціює запит до бекенду);
- методи API: отримання списку кнопок, отримання списку тегів, пошук за ключовим словом та тегом, запит детальної інформації за ідентифікатором;
- інтерфейс підключення: REST API-запити з React;
- розділення ролей: публічний доступ для користувачів, авторизований доступ до панелі адміністратора для наповнення та оновлення інформації.

Інтеграція має забезпечити швидке реагування інтерфейсу, підтримку запитів у реальному часі (без перезавантаження сторінки), а також легке оновлення серверної логіки без втручання в основний код порталу.

A.1.2 Функціональні вимоги

Система чатбота повинна реалізовувати функціональність, орієнтовану на спрощений доступ користувача до екологічної інформації, що зберігається в базі даних. Основні функціональні вимоги включають:

Користувацький інтерфейс:

- відображення інтерактивної кнопки чатбота на сторінці;
- відкриття вікна чатбота з динамічними кнопками: «FAQ», «Публікації»;
- пошуковий інтерфейс для введення запиту за ключовими словами;

- динамічне підвантаження результатів запитів із серверної частини;
- можливість редіректу до джерела інформації за кожною категорією (наприклад, окремий звіт або новину).

Серверна логіка:

- а) надання структурованих API-ендпоінтів для отримання:
 - 1) списку доступних категорій;
 - 2) списку публікацій за категорією;
 - 3) результатів пошуку за ключовими словами;
 - 4) повної інформації за ID;
- б) реалізація авторизації для адміністративного інтерфейсу;
- в) обробка запитів з боку фронтенду з валідацією та кешуванням даних за потреби.

Адміністративна панель:

- вхід під обліковим записом адміністратора;
- створення, редагування та видалення інформаційних матеріалів (новин, звітів, конференцій);
- створення, редагування кнопок;
- перегляд структури наявних записів;
- можливість швидкого оновлення вмісту без перезавантаження клієнтської частини.

Ролі та права доступу:

- користувачі (гості): мають доступ до відкритої інформації через інтерфейс чатбота;
- адміністратори: мають повний доступ до операцій редагування через захищену панель управління.

A.1.3 Технічні вимоги

Архітектура системи. Інтеграція чатбота реалізується на основі MVC-архітектури, де:

- model – відповідає за зберігання та обробку даних (новини, звіти, конференції);
- view – формує інтерфейс користувача, що взаємодіє з API;
- controller – обробляє запити клієнта, викликає сервіси та передає відповідь у форматі JSON.

Така структура забезпечує чітке розділення логіки, масштабованість та можливість швидкого доповнення функціоналу.

Вимоги до клієнтської частини для підключення до існуючої архітектури вебпорталу:

- JavaScript-бібліотека: React 19+;
- метод підключення до API: Axios.

Апаратні вимоги:

- CPU: 2 ядра;
- RAM: 2 ГБ;
- storage: 5+ ГБ SSD;
- ОС: Ubuntu 24.10 або сумісна.

Інтерфейси для обміну даними:

- а) HTTP/HTTPS-запити від React;
- б) публічні REST-ендпоінти:
 - 1) GET запит для отримання handbook масиву тегів;
 - 2) GET запит для отримання масиву кнопок;
 - 3) GET запити для пошуку екологічних новин та робіт;
- в) адміністративні API:
 - 1) CRUD запити для редагування та додавання робіт в БД;
 - 2) CRUD запити для редагування та додавання новин в БД;
 - 3) CRUD запити для редагування та додавання кнопок в БД.

А.2 Виконання робіт

Реалізація проєкту передбачає проходження кількох послідовних етапів, що охоплюють усі ключові фази життєвого циклу вебсистеми – від планування до супроводу:

А.2.1 Планування

Включає:

- аналіз потреб користувача та цілей інтеграції;
- формування технічного завдання;
- визначення технологічного стеку, структури БД та архітектури системи;
- розробку загального макету інтерфейсу чатбота та його поведінки.

А.2.2 Розробка серверної частини

Включає:

- створення структури проєкту відповідно до шаблону MVC;
- розробка моделей, міграцій та сервісів для роботи з новинами, звітами, конференціями тощо;
- реалізація REST API для публічного доступу та адмін-панелі;
- налаштування системи авторизації для адміністраторів.

А.2.3 Розробка клієнтської частини

Включає:

- інтеграція кнопки чатбота на екологічній сторінці;
- розробка компонентів для динамічних кнопок і пошукового поля;

- підключення до серверного API та обробка відповідей;
- забезпечення адаптивності інтерфейсу.

A.2.4 Адміністративна панель

Включає:

- створення авторизованого інтерфейсу керування контентом (додавання, редагування, видалення);
- перевірка валідації форм, повідомлень про помилки;
- тестування CRUD-операцій з базою даних.

A.2.5 Тестування

Включає:

- проведення функціонального, інтеграційного та UX-тестування;
- перевірка відповідності поведінки чатбота очікуваним сценаріям;
- автоматизоване тестування API через Postman;
- виправлення виявлених помилок.

A.2.6 Впровадження

Включає:

- розгортання серверної частини на хостингу або локальному сервері;
- інтеграція компонента чатбота у структуру існуючої вебсторінки;
- перевірка стабільності роботи після інтеграції.

A.2.7 Супровід

Включає:

- документування структури системи та інструкцій користування;
- підготовка середовища для подальшого розширення функціоналу.

A.3 Тестування

A.3.1 План тестування

Для забезпечення якості розробленої системи планується проведення таких видів тестування:

- функціональне тестування – перевірка відповідності реалізованого функціоналу вимогам: навігація, фільтрація, чаткомунікація, авторизація тощо;
- інтеграційне тестування – тестування взаємодії між компонентами системи: Інтерфейс ↔ API ↔ БД;
- навантажувальне тестування – моделювання одночасної активності кількох користувачів (2–10) для оцінки продуктивності системи у пікових режимах;
- тестування збоїв – перевірка поведінки системи при втраті з'єднання з сервером чи базою даних;
- кросбраузерне тестування – перевірка коректного відображення інтерфейсу у сучасних браузерах (Chrome, Firefox, Edge).

A.3.2 Критерії приймання

Інтеграція вважається успішною за виконання наступних умов:

- усі основні функції працюють без критичних помилок (авторизація, перегляд, пошук, чат);

- інтерфейс коректно відображається у підтримуваних браузерх і пристроях;
- реакція на події (нові повідомлення, навігація) не перевищує 1 секунди;
- дані передаються коректно між усіма компонентами (без втрати чи дублювання);
- система залишається стабільною при навантаженні не менше 10 одночасних активних сесій;
- всі відомі помилки після тестування зафіксовані та виправлені або документовані як некритичні.

A.4 Документація

У процесі розробки та впровадження системи повинна бути створена така документація:

- технічна документація – опис архітектури проєкту, використовуваних технологій, структури бази даних та взаємодії компонентів;
- інструкція адміністратора – опис адмін-панелі та сценарії взаємодії з БД та редагування боту;
- інструкція користувача – опис інтерфейсу та сценаріїв взаємодії з ботом для кінцевих користувачів.

A.5 Безпека

Враховуючи специфіку архітектури системи, до вимог безпеки належать:

- обмеження доступу до адміністративної панелі через авторизацію (email + пароль);
- захист GET-запитів від стороннього втручання – перевірка допустимих параметрів і обмеження доступу до службових маршрутів;

- інформація, що передається, не повинна містити персональних або чутливих даних користувачів;
- адміністративні дані (у панелі управління) мають бути захищені аутентифікацією і не передаватися у відкритому вигляді через URL.

ДОДАТОК Б
Інструкція користувача

Б.1 Загальна інформація

Чатбот-віджет призначений для забезпечення зручного доступу до екологічної інформації та сприяння формуванню позитивного іміджу університету як активного учасника зелених ініціатив.

Функціональні можливості чатбота:

- підтримка української та англійської мови;
- пошук за тегами та ключовими словами;
- дві основні вкладки: FAQ та Академічні ресурси;
- інтуїтивний інтерфейс з елементами навігації (включно з «хлібними крихтами»).

Б.2 Доступ до чатбота

Послідовність дій для відкриття віджету чатботу:

- відкрийте офіційний вебпортал університету <https://zp.edu.ua>;
- на головній сторінці, у правому нижньому куті, з'явиться зелена кругла кнопка запуску віджету чатботу;
- натисніть кнопку зі значком повідомлення або написом “Еко-чат”, щоб активувати чат.

Б.3 Інтерфейс чатбота

Після відкриття віджета користувач бачить інтерфейс, що містить:

- а) мовний перемикач (UA / EN);
- б) логотип університету (у лівому верхньому куті віджету);
- в) кнопку закриття (у правому верхньому куті віджету);
- г) дві вкладки:
 - 1) FAQ – відповіді на часті запитання про екополітику університету;

- 2) академічні ресурси – доступ до навчальних матеріалів, посилань, презентацій тощо.

Б.4 Робота з чатботом

Б.4.1 Перемикач мови

Щоб обрати мову локалізації виконайте дії:

- знайдіть іконку прапора у правому верхньому куті віджету;
- натисніть на іконку прапора;
- оберіть бажану мову інтерфейсу (українська або англійська).

Б.4.2 Меню FAQ

Послідовність дій для навігації по меню FAQ:

- а) перейдіть на вкладку FAQ;
- б) натисніть на цікавий елемент:
 - 1) якщо це інформаційний елемент – відобразиться коротка відповідь та посилання на джерело;
 - 2) якщо це категорія – відкриється підменю;
- в) повторюйте навігацію, доки не буде знайдено потрібну відповідь;
- г) для перегляду повної інформації натисніть кнопку «Подивитись»;
- д) перейдіть за наданим посиланням для детальнішої інформації.

Б.4.3 Хлібні крихти

Для повернення до попередньої категорії:

- знайдіть кнопку «Повернутися до {Назва розділу}» у лівій верхній частині меню;
- натисніть на неї, щоб перейти до попереднього рівня меню.

Б.4.4 Меню «Академ. ресурси»

Щоб знайти певний академ. ресурс виконайте дії:

- перейдіть на вкладку «Академ. ресурси»;
- (опціонально) оберіть категорію з випадаючого меню;
- (опціонально) введіть ключові слова в поле пошуку;
- серед результатів знайдіть потрібний та натисніть на його назву;
- ознайомтесь із коротким описом та натисніть кнопку «Детальніше»;
- перейдіть за посиланням для перегляду повного ресурсу.

Б.4.5 Закриття чатбота

Щоб закрити віджет чатботу знайдіть кнопку хрестика у правому верхньому куті чатбота, натисніть на неї для закриття віджету.

Б.5 Рекомендації з використання

Перелік рекомендацій при використанні чатботу:

- формулюйте запити чітко і лаконічно, використовуючи ключові слова (наприклад: "зелений кампус", "сортування відходів", "енергоефективність");
- використовуйте вкладки для кращої навігації та доступу до структурованої інформації;
- у разі відсутності потрібної відповіді – спробуйте інше формулювання або скористайтесь меню категорій;
- переключайте мову інтерфейсу, якщо потрібна англomовна версія відповідей або матеріалів;
- використовуйте пошук за тегами у вкладці «Академ. ресурси» для швидкого доступу до тематичних матеріалів.

ДОДАТОК В
Керівництво адміністратора

В.1 Вхід до системи

Щоб увійти до адмін панелі виконайте наступні дії:

- перейдіть за посиланням на адмінпанель (наприклад, {{host}}/admin);
- у формі авторизації введіть логін та пароль;
- натисніть кнопку "Увійти";
- у разі успішної авторизації ви потрапите до головного меню.
- Якщо введені дані некоректні, з'явиться повідомлення про помилку. Зверніться до системного адміністратора, якщо ви забули пароль.

В.2 Керування FAQ (Поширеними запитаннями)

Щоб переглянути всі запитання:

- у головному меню оберіть FAQ;
- відкриється таблиця з наявними записами.

Щоб додати нове запитання:

- а) натисніть кнопку "Додати";
- б) заповніть поля:
 - 1) питання (UA / EN);
 - 2) відповідь (UA / EN);
 - 3) посилання;
- в) натисніть "Зберегти".

Щоб змінити запитання:

- у таблиці знайдіть потрібний запис і натисніть "Редагувати";
- змініть потрібні поля;
- натисніть "Оновити".

Щоб видалити запитання:

- натисніть кнопку "Видалити" навпроти запису;
- підтвердіть дію у спливаючому вікні.

В.3 Керування академічними ресурсами

Щоб переглянути всі ресурси:

- у головному меню оберіть Академічні ресурси;
- відкриється таблиця з наявними ресурсами.

Щоб додати новий ресурс:

- а) натисніть кнопку "Додати";
- б) заповніть поля:
 - 1) назва ресурсу (UA / EN);
 - 2) опис (UA / EN);
 - 3) посилання;
 - 4) тег;
- в) натисніть "Зберегти".

Щоб редагувати ресурс:

- натисніть кнопку "Редагувати";
- внесіть зміни;
- натисніть "Оновити".

Щоб видалити ресурс:

- натисніть "Видалити";
- підтвердьте дію.

В.4 Захист і безпека

Список рекомендацій при користуванні адмін панелю:

- використовуйте надійний пароль;
- не передавайте облікові дані третім особам;
- регулярно виходьте з облікового запису після завершення роботи.

ДОДАТОК Д
Текст програми

Цей додаток містить повний текст реалізованого програмного коду, структурований за основними компонентами системи.

Програмна система складається з трьох незалежних частин:

- laravel back-end (розділ Д.1) — відповідає за обробку запитів, взаємодію з базою даних і реалізацію бізнес-логіки;
- react front-end (розділ Д.2) — реалізує інтерактивний користувацький інтерфейс для вбудовування чатбота у вебпортал;
- blade server rendering (розділ Д.3) — забезпечує рендеринг сторінок адміністративної панелі.

Кожен підрозділ містить окремі фрагменти коду, що відображають їх призначення, логіку роботи та взаємодію з іншими модулями системи. Текст програми подано для посилання у відповідних розділах пояснювальної записки.

Д.1 Laravel Back-end

Д.1.1 Міграції

Д.1.1.1 Create_users_table

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateUsersTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('users', function (Blueprint $table) {
            $table->id();

            $table->string('name');

            $table->string('email')->unique();
            $table->timestamp('email_verified_at')->nullable();

            $table->string('password');
```

```

        $table->rememberToken();
        $table->dateTime('last_login')->nullable();
        $table->timestamps();
    });
}

/**
 * Reverse the migrations.
 *
 * @return void
 */
public function down()
{
    Schema::dropIfExists('users');
}
}

```

Д.1.1.2 Create_password_resets

```

<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('password_resets', function (Blueprint $table) {
            $table->string('email')->index();
            $table->string('token');
            $table->timestamp('created_at')->nullable();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('password_resets');
    }
};

```

Д.1.1.3 Create_failed_jobs

```

<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;

```

```

use Illuminate\Support\Facades\Schema;

class CreateFailedJobsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('failed_jobs', function (Blueprint $table) {
            $table->id();
            $table->string('uuid')->unique();
            $table->text('connection');
            $table->text('queue');
            $table->longText('payload');
            $table->longText('exception');
            $table->timestamp('failed_at')->useCurrent();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('failed_jobs');
    }
}

```

Д.1.1.4 Create_personal_access_tokens

```

<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreatePersonalAccessTokensTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('personal_access_tokens', function (Blueprint $table) {
            $table->id();
            $table->morphs('tokenable');
            $table->string('name');
            $table->string('token', 64)->unique();
            $table->text('abilities')->nullable();
            $table->timestamp('last_used_at')->nullable();
            $table->timestamp('expires_at')->nullable();
            $table->timestamps();
        });
    }
}

```

```

    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('personal_access_tokens');
    }
}

```

Д.1.1.5 Create_works

```

<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('works', function (Blueprint $table) {
            $table->id();
            $table->string('tag');
            $table->string('ua_name');
            $table->string('en_name');
            $table->text('ua_description');
            $table->text('en_description');
            $table->string('url');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('works');
    }
};

```

Д.1.1.6 Create_buttons

```

<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

```

```

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('buttons', function (Blueprint $table) {
            $table->id();
            $table->string('ua_title');
            $table->string('en_title');
            $table->text('ua_description')->nullable();
            $table->text('en_description')->nullable();
            $table->string('url')->nullable();
            $table->foreignId('parent_id')->nullable()->constrained('buttons')->onDelete('cascade');
            $table->timestamps();
        });
    }

    /**
     * @return void
     */
    public function down(): void
    {
        Schema::dropIfExists('buttons');
    }
};

```

Д.1.1.7 Create_sessions

```

<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('sessions', function (Blueprint $table) {
            $table->string('id')->primary();
            $table->foreignId('user_id')->nullable()->index();
            $table->string('ip_address', 45)->nullable();
            $table->text('user_agent')->nullable();
            $table->longText('payload');
            $table->integer('last_activity')->index();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('sessions');
    }
};

```

Д.1.2 Seeders

Д.1.2.1 UserSeeder

```
<?php

namespace Database\Seeders;

use App\Enums\RolesEnum;
use App\Models\Language;
use App\Models\User;
use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;

class UserSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    public function run(): void
    {
        User::updateOrCreate(
            ['email' => 'root@root.ua'],
            [
                'name' => 'Root',
                'email_verified_at' => now(),
                'password' => bcrypt('password'),
            ]
        );
    }
}
```

Д.1.2.2 ButtonSeeder

```
<?php

namespace Database\Seeders;

use App\Models\Button;
use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;

class ButtonSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    public function run(): void
    {
        Button::create([
            'ua_title' => 'Що таке GreenMetric',
            'en_title' => 'What is GreenMetric',
            'ua_description' => 'UI GreenMetric — це щорічний міжнародний рейтинг ефективності університетів у сфері сталого розвитку. Університетам дають бали, що відображають їхні зусилля щодо зменшення екологічного сліду університету та підтримки освіти та досліджень.',
            'en_description' => 'The UI GreenMetric is an annual international ranking of the sustainability performance of universities. Universities are given a score reflecting their efforts in reducing the ecological footprint of the university and sustainability education and research.',
        ]);
    }
}
```

```

        'url' => 'https://en.wikipedia.org/wiki/UI_GreenMetric',
        'parent_id' => null,
    ));

    Button::create([
        'ua_title' => 'Позиція НУЗП в GreenMetric',
        'en_title' => 'ZNU GreenMetric position',
        'ua_description' => 'Наразі ми не приймаємо участь в рейтингу GreenMetric, але виконавши
план сталого розвитку до 2027, плануємо набрати необхідну кількість екологічних досягнень для прийняття
участі в конкурсі',
        'en_description' => 'We are not currently participating in the GreenMetric rating, but after
implementing our sustainable development plan by 2027, we plan to gain the necessary number of environmental
achievements to participate in the competition.',
        'url' => 'https://greenmetric.ui.ac.id/rankings/overall-rankings-2024',
        'parent_id' => null,
    ));

    Button::create([
        'ua_title' => 'Екологічні патенти та досягнення',
        'en_title' => 'Environmental patents and achievements',
        'ua_description' => 'Спосіб керування активним фільтром та енергозберігаюча система
електроживлення на його основі. Енергозберігаюче керування в системах автоматизованого електропривода
та удосконалення оплати електроспоживачів за реактивну енергію',
        'en_description' => 'Active filter control method and energy-saving power supply system based on
it. Energy-saving control in automated electric drive systems and improvement of payment of electricity consumers
for reactive energy',
        'url' => 'https://zp.edu.ua/uploads/reclam_srd.pdf',
        'parent_id' => null,
    ));

    Button::create([
        'ua_title' => 'Стратегія розвитку Національного університету "Запорізька політехніка" на
період 2023-2027 років',
        'en_title' => 'Development Strategy of the National University "Zaporizhzhia Polytechnic" for the
period 2023-2027',
        'ua_description' => 'Стратегія розвитку Національного університету «Запорізька політехніка»
визначає напрямки розвитку Університету на сучасному етапі розвитку суспільства та економіки регіону та
країни.',
        'en_description' => 'The Development Strategy of the National University "Zaporizhzhia
Polytechnic" defines the development areas of the University at the current stage of the society and economy
development of the region and country.',
        'url' => 'https://zp.edu.ua/uploads/pubdocs/2025/Nakaz_N61_vid_24.02.25.pdf',
        'parent_id' => null,
    ));

    Button::create([
        'ua_title' => 'Сертифікація з енергоефективності',
        'en_title' => 'Energy efficiency certification',
        'ua_description' => '',
        'en_description' => '',
        'url' => 'https://zp.edu.ua/sertyfikaciya-z-energoefektyvnosti',
        'parent_id' => null,
    ));

    Button::create([
        'ua_title' => 'Міжнародна науково-практична конференція «Бізнес-екосистеми: сталий
розвиток 2025»',
        'en_title' => 'International Scientific and Practical Conference Business Ecosystems: Sustainable
Development 2025 (BESD-2025)',
        'ua_description' => '',
        'en_description' => '',
        'url' => 'https://zp.edu.ua/conference-2025-1-1',
        'parent_id' => null,
    ));

```

```

    ));

    $allReports = Button::create([
        'ua_title' => 'Звіти зі сталого розвитку',
        'en_title' => 'Reports',
        'url' => null,
        'parent_id' => null,
    ));

    $reports2025 = Button::create([
        'ua_title' => 'Звіти зі сталого розвитку за 2025',
        'en_title' => 'Reports 2025',
        'url' => null,
        'parent_id' => $allReports->id,
    ));

    $reports2024 = Button::create([
        'ua_title' => 'Звіти зі сталого розвитку за 2024',
        'en_title' => 'Reports 2024',
        'url' => null,
        'parent_id' => $allReports->id,
    ));

    $reports2023 = Button::create([
        'ua_title' => 'Звіти зі сталого розвитку за 2023',
        'en_title' => 'Reports 2023',
        'url' => null,
        'parent_id' => $allReports->id,
    ));

    Button::create([
        'ua_title' => 'Звіт за весну 2023',
        'en_title' => 'Report spring 2023',
        'ua_description' => 'Звітуємо що все дуже добре, добре добре, повний звіт за кнопкою нижче',
        'en_description' => 'Reporting that everything is Ok, okey, fantastic. Click button to see details',
        'url' => '/report/2023/spring',
        'parent_id' => $reports2023->id,
    ));

    Button::create([
        'ua_title' => 'Звіт за літо 2023',
        'en_title' => 'Report summer 2023',
        'ua_description' => 'Звітуємо що все дуже добре, добре добре, повний звіт за кнопкою нижче',
        'en_description' => 'Reporting that everything is Ok, okey, fantastic. Click button to see details',
        'url' => '/report/2023/summer',
        'parent_id' => $reports2023->id,
    ));

    Button::create([
        'ua_title' => 'Звіт за осінь 2023',
        'en_title' => 'Report autumn 2023',
        'ua_description' => 'Звітуємо що все дуже добре, добре добре, повний звіт за кнопкою нижче',
        'en_description' => 'Reporting that everything is Ok, okey, fantastic. Click button to see details',
        'url' => '/report/2023/autumn',
        'parent_id' => $reports2023->id,
    ));

    Button::create([
        'ua_title' => 'Звіт за зиму 2023',
        'en_title' => 'Report winter 2023',
        'ua_description' => 'Звітуємо що все дуже добре, добре добре, повний звіт за кнопкою нижче',
        'en_description' => 'Reporting that everything is Ok, okey, fantastic. Click button to see details',
        'url' => '/report/2023/winter',
        'parent_id' => $reports2023->id,
    ));

```

```

        Button::create([
            'ua_title' => 'Звіт за весну 2024',
            'en_title' => 'Report spring 2024',
            'ua_description' => 'Звітуємо що все дуже добре, добре добре, повний звіт за кнопкою нижче',
            'en_description' => 'Reporting that everything is Ok, okey, fantastic. Click button to see details',
            'url' => '/report/2024/spring',
            'parent_id' => $reports2024->id,
        ]);
        Button::create([
            'ua_title' => 'Звіт за літо 2024',
            'en_title' => 'Report summer 2024',
            'ua_description' => 'Звітуємо що все дуже добре, добре добре, повний звіт за кнопкою нижче',
            'en_description' => 'Reporting that everything is Ok, okey, fantastic. Click button to see details',
            'url' => '/report/2024/summer',
            'parent_id' => $reports2024->id,
        ]);
        Button::create([
            'ua_title' => 'Звіт за осінь 2024',
            'en_title' => 'Report autumn 2024',
            'ua_description' => 'Звітуємо що все дуже добре, добре добре, повний звіт за кнопкою нижче',
            'en_description' => 'Reporting that everything is Ok, okey, fantastic. Click button to see details',
            'url' => '/report/2024/autumn',
            'parent_id' => $reports2024->id,
        ]);
        Button::create([
            'ua_title' => 'Звіт за зиму 2024',
            'en_title' => 'Report winter 2024',
            'ua_description' => 'Звітуємо що все дуже добре, добре добре, повний звіт за кнопкою нижче',
            'en_description' => 'Reporting that everything is Ok, okey, fantastic. Click button to see details',
            'url' => '/report/2024/winter',
            'parent_id' => $reports2024->id,
        ]);

        Button::create([
            'ua_title' => 'Звіт за весну 2025',
            'en_title' => 'Report spring 2025',
            'ua_description' => 'Звітуємо що все дуже добре, добре добре, повний звіт за кнопкою нижче',
            'en_description' => 'Reporting that everything is Ok, okey, fantastic. Click button to see details',
            'url' => '/report/2025/spring',
            'parent_id' => $reports2025->id,
        ]);
    }
}

```

Д.1.2.3 WorkSeeder

```

<?php

namespace Database\Seeders;

use App\Enums\TagEnum;
use App\Models\Work;
use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;

class WorkSeeder extends Seeder

```

```

{
  public function run(): void
  {
    $works = [
      [
        'tag' => TagEnum::WORK,
        'ua_name' => 'Вплив забруднення повітря на здоров'я населення',
        'en_name' => 'Impact of Air Pollution on Public Health',
        'ua_description' => 'Дослідження впливу забруднення повітря на захворюваність дихальної системи в великих містах України. Аналіз даних за останні 10 років.',
        'en_description' => 'Study of the impact of air pollution on respiratory diseases in large cities of Ukraine. Analysis of data over the last 10 years.',
        'url' => 'https://example.com/air-pollution-health'
      ],
      [
        'tag' => TagEnum::CONFERENCE,
        'ua_name' => 'Відновлювані джерела енергії в Україні: перспективи розвитку',
        'en_name' => 'Renewable Energy Sources in Ukraine: Development Prospects',
        'ua_description' => 'Аналіз потенціалу сонячної та вітрової енергетики в різних регіонах України. Економічні та екологічні переваги.',
        'en_description' => 'Analysis of the potential of solar and wind energy in different regions of Ukraine. Economic and environmental benefits.',
        'url' => 'https://example.com/renewable-energy-ukraine'
      ],
      [
        'tag' => TagEnum::WORK,
        'ua_name' => 'Ефективні методи переробки пластикових відходів',
        'en_name' => 'Effective Methods of Plastic Waste Recycling',
        'ua_description' => 'Огляд сучасних технологій переробки пластику та їх впровадження в Україні. Порівняльний аналіз з європейським досвідом.',
        'en_description' => 'Review of modern plastic recycling technologies and their implementation in Ukraine. Comparative analysis with European experience.',
        'url' => 'https://example.com/plastic-recycling-methods'
      ],
      [
        'tag' => TagEnum::NEWS,
        'ua_name' => 'Біорізноманіття Карпатського регіону: сучасні загрози',
        'en_name' => 'Biodiversity of the Carpathian Region: Current Threats',
        'ua_description' => 'Дослідження впливу змін клімату та антропогенного фактора на екосистеми Карпат. Рекомендації зі збереження біорізноманіття.',
        'en_description' => 'Study of the impact of climate change and anthropogenic factors on Carpathian ecosystems. Biodiversity conservation recommendations.',
        'url' => 'https://example.com/carpathian-biodiversity'
      ],
      [
        'tag' => TagEnum::REPORT,
        'ua_name' => 'Екологічні наслідки використання пестицидів в сільському господарстві',
        'en_name' => 'Environmental Consequences of Pesticide Use in Agriculture',
        'ua_description' => 'Дослідження впливу хімічних засобів захисту рослин на ґрунт та водні ресурси. Альтернативні методи захисту рослин.',
        'en_description' => 'Study of the impact of plant protection chemicals on soil and water resources. Alternative plant protection methods.',
        'url' => 'https://example.com/pesticides-environment'
      ]
    ];

    foreach ($works as $work) {
      Work::create($work);
    }
  }
}

```

Д.1.2.4 DatabaseSeeder

```
<?php

namespace Database\Seeders;

use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder
{
    /**
     * Seed the application's database.
     *
     * @return void
     */
    public function run()
    {
        // \App\Models\User::factory(10)->create();
        $this->call([
            UserSeeder::class,
            WorkSeeder::class,
            ButtonSeeder::class,
        ]);
    }
}
```

Д.1.3 Eloquent-моделі

Д.1.3.1 User

```
<?php

namespace App\Models;

use Illuminate\Contracts\Auth\MustVerifyEmail;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;
use Laravel\Sanctum\HasApiTokens;

class User extends Authenticatable
{
    use HasApiTokens, HasFactory, Notifiable;

    /**
     * The attributes that are mass assignable.
     *
     * @var array<int, string>
     */
    protected $fillable = [
        'name',
        'email',
        'password',
    ];

    /**
     * The attributes that should be hidden for serialization.
     *

```

```

    * @var array<int, string>
    */
    protected $hidden = [
        'password',
        'remember_token',
    ];

    /**
     * The attributes that should be cast.
     *
     * @var array<string, string>
     */
    protected $casts = [
        'email_verified_at' => 'datetime',
    ];
}

```

Д.1.3.2 Button

```

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\BelongsTo;
use Illuminate\Database\Eloquent\Relations\HasMany;

class Button extends Model
{
    protected $fillable = [
        'ua_title',
        'en_title',
        'ua_description',
        'en_description',
        'url',
        'parent_id',
    ];

    protected $hidden = ['created_at', 'updated_at'];

    /**
     * Рекурсивное отношение для получения всех потомков.
     */
    public function children(): HasMany
    {
        return $this->hasMany(Button::class, 'parent_id');
    }

    /**
     * Получение родительской кнопки.
     */
    public function parent(): BelongsTo
    {
        return $this->belongsTo(Button::class, 'parent_id');
    }
}

```

Д.1.3.3 Work

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Work extends Model
{
    use HasFactory;

    /**
     * The attributes that are mass assignable.
     *
     * @var array<string>
     */
    protected $fillable = [
        'tag',
        'ua_name',
        'en_name',
        'ua_description',
        'en_description',
        'url',
    ];
}
```

Д.1.4 Controllers

Д.1.4.1 AuthController

```
<?php

namespace App\Http\Controllers;

use App\Http\Requests\Auth\LoginRequest;
use App\Services\Auth\Interfaces\AuthServiceInterface;
use Illuminate\Http\JsonResponse;
use Illuminate\Support\Facades\Auth;

class AuthController extends Controller
{
    /**
     * @param AuthServiceInterface $authService
     */
    public function __construct(
        public readonly AuthServiceInterface $authService
    )
    {

    }

    /**
     * @param LoginRequest $request
     * @return JsonResponse
     */
}
```

```

public function login(LoginRequest $request): JsonResponse
{
    if ($this->authService->checkUserPassword($request->validated())) {
        return response()->json(
            $this->authService->login($request->validated()),
            202
        );
    }

    return response()->json([
        'status' => 'error',
        'message' => 'incorrect password'
    ], 401);
}

/**
 * @return JsonResponse
 */
public function logout(): JsonResponse
{
    Auth::user()?->currentAccessToken()->delete();

    return response()->json("Logged out",200);
}
}

```

Д.1.4.2 ButtonController

```

<?php

namespace App\Http\Controllers;

use App\Http\Requests\Faq\GetChildrenRequest;
use App\Http\Requests\Faq\GetMenusRequest;
use App\Http\Requests\Button\StoreButtonRequest;
use App\Http\Requests\Button\UpdateButtonRequest;
use App\Models\Button;
use Illuminate\Http\JsonResponse;

class ButtonController extends Controller
{
    /**
     * @return JsonResponse
     */
    public function getMenus(GetMenusRequest $request): JsonResponse
    {
        $locale = $request->get('locale', 'uk');

        $rootButtons = Button::whereNull('parent_id')->get();

        $localizedButtons['buttons'] = $rootButtons->map(function ($button) use ($locale) {
            return [
                'id' => $button->id,
                'name' => $locale === 'uk' ? $button->ua_title : $button->en_title,
                'description' => $locale === 'uk' ? $button->ua_description : $button->en_description,
                'url' => $button->url,
            ];
        });

        return response()->json($localizedButtons);
    }
}

```

```

}

public function getChildren(GetChildrenRequest $request, Button $button): JsonResponse
{
    $locale = $request->get('locale', 'uk');

    $children['buttons'] = $button->children->map(function ($child) use ($locale) {
        return [
            'id' => $child->id,
            'name' => $locale === 'uk' ? $child->ua_title : $child->en_title,
            'description' => $locale === 'uk' ? $child->ua_description : $child->en_description,
            'url' => $child->url,
        ];
    });
    $children['parent_id'] = $button->parent_id;

    return response()->json($children);
}

/**
 * Display a listing of the resource.
 */
public function index()
{
    //
}

/**

/**
 * Store a newly created resource in storage.
 */
public function store(StoreButtonRequest $request)
{
    $button = Button::create($request->validated());
    return response()->json($button);
}

/**
 * Display the specified resource.
 */
public function show(Button $button)
{
    return response()->json($button->load('children'));
}

/**
 * Update the specified resource in storage.
 */
public function update(UpdateButtonRequest $request, Button $button)
{
    $button->update($request->validated());
    return response()->json($button->refresh());
}

/**
 * Remove the specified resource from storage.
 */
public function destroy(Button $button)
{
    $button->delete();
    return response("Button deleted successfully");
}

```

```
    }
}
```

Д.1.4.3 WorkController

```
<?php

namespace App\Http\Controllers;

use App\Http\Requests\Work\FindWorksRequest;
use App\Http\Requests\Work\StoreWorkRequest;
use App\Http\Requests\Work\UpdateWorkRequest;
use App\Models\Work;
use App\Services\Work\Interfaces\WorkServiceInterface;

class WorkController extends Controller
{
    public function __construct(
        public readonly WorkServiceInterface $workService
    )
    {

    }

    /**
     * Display a listing of the resource.
     */
    public function index(FindWorksRequest $request)
    {
        return response()->json(
            $this->workService->showWorks($request->validated())
        );
    }

    /**
     * Store a newly created resource in storage.
     */
    public function store(StoreWorkRequest $request)
    {
        return response()->json(
            $this->workService->createWork($request->validated())
        );
    }

    /**
     * Display the specified resource.
     */
    public function show(Work $work)
    {
        return response()->json($work);
    }

    /**
     * Update the specified resource in storage.
     */
    public function update(UpdateWorkRequest $request, Work $work)
    {
        return response()->json(
            $this->workService->updateWork($work , $request->validated())
        );
    }
}
```

```

/**
 * Remove the specified resource from storage.
 */
public function destroy(Work $work)
{
    $work->delete();
    return response("Work deleted successfully");
}
}

```

Д.1.4.4 TagEnumController

```

<?php

namespace App\Http\Controllers\HandBook;

use App\Enums\TagEnum;

use App\Http\Controllers\Controller;
use Illuminate\Http\JsonResponse;

class TagEnumController extends Controller
{
    /**
     * @return \Illuminate\Http\JsonResponse
     */
    public function index(): JsonResponse
    {
        return response()->json(TagEnum::getCases());
    }
}

```

Д.1.5 Enums

Д.1.5.1 TagEnum

```

<?php

namespace App\Enums;

use App\Attributes\UaLabel;
use App\Attributes\EnLabel;
use App\Traits\AttributableEnumTrait;
use App\Traits\EnumHelperTrait;

enum TagEnum: int
{
    use AttributableEnumTrait;
    use EnumHelperTrait;

    #[EnLabel('Reports')]
    #[UaLabel('Звіти')]
    case REPORT = 1;

    #[EnLabel('Conferences')]
    #[UaLabel('Конференції')]
    case CONFERENCE = 2;
}

```

```

#[EnLabel('News')]
#[UaLabel('Новини')]
case NEWS = 3;

#[EnLabel('Science works')]
#[UaLabel('Наукові роботи')]
case WORK = 4;

/**
 * @param int $id
 * @return self|null
 */
public static function getById(int $id): ?self
{
    return self::tryFrom($id);
}
}

```

Д.1.5.2 AttributableEnumTrait

```

<?php

namespace App\Traits;

use App\Attributes\AttributeProperty;
use BadMethodCallException;
use Illuminate\Support\Str;
use ReflectionAttribute;
use ReflectionEnum;

trait AttributableEnumTrait
{
    /**
     * Call the given method on the enum case
     *
     * @param string $method
     * @param array $arguments
     * @return mixed
     * @throws \ReflectionException
     */
    public function __call(string $method, array $arguments): mixed
    {
        $reflection = new ReflectionEnum(static::class);
        $attributes = $reflection->getCase($this->name)->getAttributes();

        $filtered_attributes = array_filter($attributes, fn (ReflectionAttribute $attribute) => $attribute-
        >getName() === AttributeProperty::ATTRIBUTE_PATH . Str::ucfirst($method));

        if (empty($filtered_attributes)) {
            throw new BadMethodCallException(sprintf('Call to undefined method %s::%s()', static::class,
            $method));
        }

        return array_shift($filtered_attributes)->newInstance()->get();
    }
}

```

Д.1.5.3 EnumHelperTrait

```

<?php

namespace App\Traits;

use App\Attributes\Description;
use App\Attributes\EnLabel;
use App\Attributes\UaLabel;

trait EnumHelperTrait
{
    /**
     * @param \ReflectionEnumBackedCase $reflection
     * @param string $attributeClass
     * @return mixed|null
     */
    private static function getAttributeValue(\ReflectionEnumBackedCase $reflection, string
$attributeClass): mixed
    {
        $attributes = $reflection->getAttributes($attributeClass);
        return !empty($attributes) ? $attributes[0]->newInstance()->get() : null;
    }

    /**
     * @return array
     */
    public static function getCases(): array
    {
        $cases = [];

        foreach (self::cases() as $case) {
            $reflection = new \ReflectionEnumBackedCase(self::class, $case->name);

            $caseData = [
                'name' => $case->name,
                'value' => $case->value,
                'en_label' => self::getAttributeValue($reflection, EnLabel::class),
                'ua_label' => self::getAttributeValue($reflection, UaLabel::class),
                'description' => self::getAttributeValue($reflection, Description::class),
            ];

            $cases[] = array_filter($caseData, function ($value) {
                return $value !== null;
            });
        }

        return $cases;
    }

    /**
     * @return array
     */
    public static function getCasesForSelect(): array
    {
        return array_map(function ($case) {
            return [
                'value' => $case->value,
                'label' => $case->label(),
            ];
        }, self::cases());
    }
}

```

```
    }
}
```

Д.1.5.4 AttributeProperty

```
<?php

namespace App\Attributes;

use Attribute;

#[Attribute(Attribute::TARGET_CLASS_CONSTANT)]
class AttributeProperty
{
    public const ATTRIBUTE_PATH = 'App\Attributes\';

    public function __construct(
        private mixed $value,
    ) {
    }

    /**
     * Get the value of the attribute
     */
    public function get(): mixed
    {
        return $this->value;
    }
}
```

Д.1.5.6 EnLabel

```
<?php

namespace App\Attributes;

use Attribute;

#[Attribute(Attribute::TARGET_CLASS_CONSTANT)]
class EnLabel extends AttributeProperty {}
```

Д.1.5.7 UaLabel

```
<?php

namespace App\Attributes;

use Attribute;

#[Attribute(Attribute::TARGET_CLASS_CONSTANT)]
class UaLabel extends AttributeProperty {}
```

Д.1.6 Services

Д.1.6.1 AuthService

```
<?php
declare(strict_types=1);
declare(ticks=1);

namespace App\Services\Auth\Classes;

use App\Models\User;
use App\Services\Auth\Interfaces\AuthServiceInterface;
use Illuminate\Auth\Events\Registered;
use Illuminate\Support\Arr;
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Facades\Log;

class AuthService implements AuthServiceInterface
{
    /**
     * @param array $request
     * @return bool
     */
    public function checkUserPassword(array $request): bool
    {
        $user = User::where('email', $request['email'])->first();

        return Hash::check($request['password'], $user->password);
    }

    /**
     * @param array $request
     * @return array
     */
    public function login(array $request): array
    {
        $user = User::where('email', $request['email'])->first();

        $user->last_login = now();
        $user->save();

        $user->tokens()->where('name', 'web')->delete();

        $expiration = now()->addDay();
        if (Arr::get($request, 'remember', false)) {
            $expiration = now()->addYear();
        }

        return [
            'id' => $user->id,
            'token' => $user->createToken('web', ['*'], $expiration)->plainTextToken,
        ];
    }
}
```

Д.1.6.2 AuthServiceInterface

```
<?php
```

```

declare(strict_types=1);
declare(ticks=1);

namespace App\Services\Auth\Interfaces;

use App\Models\User;

interface AuthServiceInterface
{
    /**
     * @param array $request
     * @return bool
     */
    public function checkUserPassword(array $request): bool;

    /**
     * @param array $request
     * @return array
     */
    public function login(array $request): array;
}

```

Д.1.6.3 WorkService

```

<?php
declare(strict_types=1);
declare(ticks=1);

namespace App\Services\Work\Classes;

use App\Models\User;
use App\Models\Work;
use App\Services\Work\Interfaces\WorkServiceInterface;
use Illuminate\Auth\Events\Registered;
use Illuminate\Database\Eloquent\Collection;
use Illuminate\Pagination\LengthAwarePaginator;
use Illuminate\Support\Arr;
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Facades\Log;

class WorkService implements WorkServiceInterface
{
    /**
     * @param array $workData
     * @return Work
     */
    public function createWork(array $workData): Work
    {
        $work = Work::create($workData);
        return $work;
    }

    /**
     * @param Work $work
     * @param array $workData
     * @return Work
     */
    public function updateWork(Work $work, array $workData): Work
    {
        $work->update($workData);
    }
}

```

```

    return $work;
}

/**
 * @param array $filters
 * @return LengthAwarePaginator
 */
public function showWorks(array $filters): LengthAwarePaginator
{
    $locale = $filters['locale'] ?? '';
    $perPage = $filters['per_page'] ?? 15;

    $query = Work::query();

    if (!empty($filters['search'])) {
        $this->applySearch($query, $filters['search'], $locale);
    }

    if (!empty($filters['tag'])) {
        $query->where('tag', (int) $filters['tag']);
    }

    if ($locale) {
        $query->select([
            'id',
            $locale === 'uk' ? 'ua_name as name' : 'en_name as name',
            $locale === 'uk' ? 'ua_description as description' : 'en_description as description',
            'url',
            'created_at',
            'updated_at'
        ]);
    }

    return $query->paginate($perPage);
}

/**
 * @param $query
 * @param string $searchTerm
 * @param string $locale
 * @return void
 */
protected function applySearch($query, string $searchTerm, string $locale): void
{
    $query->where(function ($q) use ($searchTerm, $locale) {
        if ($locale === 'uk') {
            $q->where('ua_name', 'like', "%{$searchTerm}%")
                ->orWhere('ua_description', 'like', "%{$searchTerm}%");
        } elseif ($locale === 'en') {
            $q->where('en_name', 'like', "%{$searchTerm}%")
                ->orWhere('en_description', 'like', "%{$searchTerm}%");
        } else {
            $q->where('ua_name', 'like', "%{$searchTerm}%")
                ->orWhere('ua_description', 'like', "%{$searchTerm}%")
                ->orWhere('en_name', 'like', "%{$searchTerm}%")
                ->orWhere('en_description', 'like', "%{$searchTerm}%");
        }
    });
}

```

Д.1.6.4 WorkServiceInterface

```
<?php
declare(strict_types=1);
declare(ticks=1);

namespace App\Services\Work\Interfaces;

use App\Models\User;
use App\Models\Work;
use Illuminate\Database\Eloquent\Collection;
use Illuminate\Pagination\LengthAwarePaginator;

interface WorkServiceInterface
{
    /**
     * @param array $workData
     * @return Work
     */
    public function createWork(array $workData): Work;

    /**
     * @param Work $work
     * @param array $workData
     * @return Work
     */
    public function updateWork(Work $work, array $workData): Work;

    /**
     * @param array $filters
     * @return LengthAwarePaginator
     */
    public function showWorks(array $filters): LengthAwarePaginator;
}
```

Д.1.7 AppServiceProvider

```
<?php

namespace App\Providers;

use App\Services\Auth\Classes\AuthService;
use App\Services\Auth\Interfaces\AuthServiceInterface;
use App\Services\Work\Classes\WorkService;
use App\Services\Work\Interfaces\WorkServiceInterface;
use Illuminate\Support\ServiceProvider;

class AppServiceProvider extends ServiceProvider
{
    /**
     * Register any application services.
     *
     * @return void
     */
    public function register()
    {
        $this->app->bind(AuthServiceInterface::class, AuthService::class);
        $this->app->bind(WorkServiceInterface::class, WorkService::class);
    }
}
```

```

/**
 * Bootstrap any application services.
 *
 * @return void
 */
public function boot()
{
    //
}
}

```

Д.1.8 Requests (Валідатори)

Д.1.8.1 LoginRequest

```

<?php

namespace App\Http\Requests\Auth;

use Illuminate\Foundation\Http\FormRequest;

class LoginRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
     */
    public function authorize(): bool
    {
        return true;
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array<string, \Illuminate\Contracts\Validation\ValidationRule|array<mixed>|string>
     */
    public function rules(): array
    {
        return [
            'email' => 'required|email|exists:users,email',
            'password' => 'required',
            'remember' => 'bool'
        ];
    }
}

```

Д.1.8.2 StoreButtonRequest

```

<?php

namespace App\Http\Requests\Button;

use Illuminate\Foundation\Http\FormRequest;

class StoreButtonRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.

```

```

    */
    public function authorize(): bool
    {
        return true;
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array<string, \Illuminate\Contracts\Validation\ValidationRule|array<mixed>|string>
     */
    public function rules(): array
    {
        return [
            'ua_title' => 'required|string',
            'en_title' => 'required|string',
            'ua_description' => 'nullable|string',
            'en_description' => 'nullable|string',
            'url' => 'nullable|string',
            'parent_id' => 'nullable|integer|exists:buttons,id',
        ];
    }
}

```

Д.1.8.3 UpdateButtonRequest

```

<?php

namespace App\Http\Requests\Button;

use Illuminate\Foundation\Http\FormRequest;

class UpdateButtonRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
     */
    public function authorize(): bool
    {
        return true;
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array<string, \Illuminate\Contracts\Validation\ValidationRule|array<mixed>|string>
     */
    public function rules(): array
    {
        return [
            'ua_title' => 'nullable|string',
            'en_title' => 'nullable|string',
            'ua_description' => 'nullable|string',
            'en_description' => 'nullable|string',
            'url' => 'nullable|string',
            'parent_id' => 'nullable|integer|exists:buttons,id',
        ];
    }
}

```

Д.1.8.4 GetChildrenRequest

```
<?php

namespace App\Http\Requests\Faq;

use Illuminate\Foundation\Http\FormRequest;

class GetChildrenRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
     */
    public function authorize(): bool
    {
        return true;
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array<string, \Illuminate\Contracts\Validation\ValidationRule|array<mixed>|string>
     */
    public function rules(): array
    {
        return [
            'locale' => 'required|string|in:uk,en',
        ];
    }
}
```

Д.1.8.5 GetMenusRequest

```
<?php

namespace App\Http\Requests\Faq;

use App\Enums\TagEnum;
use Illuminate\Foundation\Http\FormRequest;
use Illuminate\Validation\Rule;

class GetMenusRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
     */
    public function authorize(): bool
    {
        return true;
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array<string, \Illuminate\Contracts\Validation\ValidationRule|array<mixed>|string>
     */
    public function rules(): array
    {
        return [
```

```

        'locale' => 'required|string|in:uk,en',
    ];
}
}

```

Д.1.8.6 FindWorksRequest

```

<?php

namespace App\Http\Requests\Work;

use App\Enums\TagEnum;
use Illuminate\Foundation\Http\FormRequest;
use Illuminate\Validation\Rule;

class FindWorksRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
     */
    public function authorize(): bool
    {
        return true;
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array<string, \Illuminate\Contracts\Validation\ValidationRule|array<mixed>|string>
     */
    public function rules(): array
    {
        return [
            'tag' => [
                'nullable',
                'integer',
                Rule::enum(TagEnum::class)
            ],
            'locale' => 'nullable|string',
            'per_page' => 'nullable|integer',
            'search' => 'nullable|string',
        ];
    }
}

```

Д.1.8.7 StoreWorksRequest

```

<?php

namespace App\Http\Requests\Work;

use App\Enums\TagEnum;
use Illuminate\Foundation\Http\FormRequest;
use Illuminate\Validation\Rule;

class StoreWorkRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.

```

```

    */
    public function authorize(): bool
    {
        return true;
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array<string, \Illuminate\Contracts\Validation\ValidationRule|array<mixed>|string>
     */
    public function rules(): array
    {
        return [
            'tag' => [
                'required',
                'integer',
                Rule::enum(TagEnum::class)
            ],
            'ua_name' => 'required|string',
            'en_name' => 'required|string',
            'ua_description' => 'required|string',
            'en_description' => 'required|string',
            'url' => 'required|string',
        ];
    }
}

```

Д.1.8.8 UpdateWorksRequest

```

<?php

namespace App\Http\Requests\Work;

use App\Enums\TagEnum;
use Illuminate\Foundation\Http\FormRequest;
use Illuminate\Validation\Rule;

class UpdateWorkRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
     */
    public function authorize(): bool
    {
        return true;
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array<string, \Illuminate\Contracts\Validation\ValidationRule|array<mixed>|string>
     */
    public function rules(): array
    {
        return [
            'tag' => [
                'nullable',
                'integer',
                Rule::enum(TagEnum::class)
            ],

```

```

        'ua_name' => 'nullable|string',
        'en_name' => 'nullable|string',
        'ua_description' => 'nullable|string',
        'en_description' => 'nullable|string',
        'url' => 'nullable|string',
    ];
}
}

```

Д.1.9 Routes (Маршрутизація)

Д.1.9.1 API

```

<?php

use App\Http\Controllers\AuthController;
use App\Http\Controllers\ButtonController;
use App\Http\Controllers\HandBook\TagEnumController;
use App\Http\Controllers\WorkController;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Route;

Route::post('/login', [AuthController::class, 'login']);
Route::post('/logout', [AuthController::class, 'logout']);

Route::get('/faq', [ButtonController::class, 'getMenus'])->name('getTopMenu');
Route::get('/faq/{button}', [ButtonController::class, 'getChildren'])->name('getChildMenu');

Route::get('/tags', [TagEnumController::class, 'index']);

Route::get('/works', [WorkController::class, 'index']);

Route::group(['middleware' => 'auth:sanctum', 'as' => 'api.', function () {
    Route::resource('works', WorkController::class)->except(['create', 'edit', 'index']);
    Route::resource('buttons', ButtonController::class)->except(['create', 'edit']);
}]);

```

Д.1.9.2 Web

```

<?php

use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Route;
use App\Http\Controllers\Admin\AdminController;
use App\Http\Controllers\Admin\PublicationController;
use App\Http\Controllers\Admin\WorkController;

Route::get('/', function () {
    return view('welcome');
});

Route::get('/admin', [AdminController::class, 'signIn'])->name('signin');

Route::group(['prefix' => 'admin', 'middleware' => 'auth', 'as' => 'admin.', function () {
    Route::resource('/pubs', PublicationController::class)->except(['show']);
    Route::resource('/work', WorkController::class)->except(['show']);
}]);

```

```
Auth::routes();
```

Д.2 React Front-end

Д.2.1 index.html

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <link rel="icon" type="image/svg+xml" href="/vite.svg" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Vite + React</title>
  </head>
  <body>
    <div id="background"></div>
    <div id="root"></div>
    <script type="module" src="/src/main.jsx"></script>
  </body>
</html>
```

Д.2.2 main.jsx

```
import { StrictMode } from 'react'
import { createRoot } from 'react-dom/client'
import './index.scss'
import App from './App.jsx'
import { BrowserRouter, MemoryRouter } from "react-router-dom";

createRoot(document.getElementById('root')).render(
  <StrictMode>
    <BrowserRouter>
      <App/>
    </BrowserRouter>
  </StrictMode>,
)
```

Д.2.3 App.jsx

```
import './App.css'
import Widget from "./components/Widget.jsx";
import { QueryClient, QueryClientProvider } from "@tanstack/react-query";
import { Route, Routes } from "react-router-dom";
import Faq from "./components/Faq/Faq.jsx";
import Resources from "./components/Resources/Resources.jsx";
import FaqDesc from "./components/Faq/faqDesc.jsx";

function App() {
  const queryClient = new QueryClient();

  return (
    <QueryClientProvider client={queryClient}>
      <Routes>
        <Route path="/" element={ <Widget/> }>
        <Route path="faq">
          <Route index element={ <Faq/> }/>
```

```

        <Route path=":id" element={<FaqDesc/>} />
      </Route>
      <Route path="resources">
        <Route index element={<Resources/>} />
        <Route path=":id" element={<Resources/>} />
      </Route>
    </Route>
  </Routes>
</QueryClientProvider>
)
}

export default App;

```

Д.2.4 Components

Д.2.4.1 WidgetHeader

```

import styles from "./styles.module.scss";
import logo from "../../assets/logo.png";
import { IconButton } from "@mui/material";
import CloseIcon from "../../assets/icons/closeIcon.jsx";
import { useNavigate, useLocation } from "react-router-dom";
import { useEffect, useState } from "react";
import FlagUkraine from "../../assets/icons/flagUkraine.jsx";
import FlagUk from "../../assets/icons/flagUk.jsx";

const tabLabels = {
  uk: ["Часті питань", "Академ. ресурси"],
  en: ["FAQ", "Academic resources"]
};

const WidgetHeader = ({ toggleModal, locale, toggleLanguage }) => {
  const navigate = useNavigate();
  const location = useLocation();
  const [activeButton, setActiveButton] = useState(1);

  useEffect(() => {
    if (location.pathname.startsWith("/resources")) {
      setActiveButton(2);
    } else {
      setActiveButton(1);
    }
  }, [location.pathname]);

  const handleTabChange = (tab) => {
    setActiveButton(tab);
    const base = tab === 1 ? "/faq" : "/resources";
    navigate(`${base}?locale=${locale}`);
  };

  return (
    <div className={styles.container}>
      <div className={styles.container__image}>
        <img src={logo} alt="logo" />
      </div>
      <div className={styles.container__btnGroup}>
        {tabLabels[locale].map((label, i) => {
          const tabIndex = i + 1;
          return (

```

```

        <button
          key={tabIndex}
          onClick={() => handleTabChange(tabIndex)}
          className={` ${styles.container__btn} ${
            activeButton === tabIndex ? styles.active : ""
          }`}
        >
          {label}
        </button>
      );
    })}
</div>
<IconButton onClick={toggleLanguage} sx={{
  mr: 1.5,
  border: '1px solid #fff',
  borderRadius: '8px',
  overflow: 'hidden',
  marginRight: '1.5rem',
  '& .MuiTouchRipple-root': {
    borderRadius: '8px',
  },
  '&:hover': {
    backgroundColor: 'rgba(255, 0, 0, 0.12)',
  },
  '&:active': {
    backgroundColor: 'rgba(255, 0, 0, 0.12)',
    borderColor: 'rgba(255, 213, 0, 1)',
  }
}}>
  {locale === "uk"
    ? <FlagUkraine/>
    : <FlagUk/>
  }
</IconButton>

<IconButton onClick={toggleModal} sx={{mr: 1.5}} className={styles.btnClose}>
  <CloseIcon/>
</IconButton>
</div>
);
};

export default WidgetHeader;

```

Д.2.4.2 Widget

```

import {Box, IconButton} from "@mui/material";
import WidgetIcon from "../assets/icons/widgetIcon.jsx";
import {useState} from "react";
import WidgetHeader from "../WidgetHeader/index.jsx";
import {Outlet, useNavigate} from "react-router-dom";

const Widget = () => {
  const [isOpen, setIsOpen] = useState(false);
  const [locale, setLocale] = useState("uk");

  const navigate = useNavigate();

  const toggleModal = () => {
    const opening = !isOpen;

```

```

    setIsOpen(opening);
    if (opening) {
      navigate(`/faq?locale=${locale}`);
    }
  };

  const toggleLanguage = () => {
    setLocale((prev) => (prev === "uk" ? "en" : "uk"));
    if (isOpen) {
      navigate(`/faq?locale=${locale === "uk" ? "en" : "uk"}`);
    }
  };

  return (
    <Box position="absolute" bottom={50} right={50}>
      {isOpen ? (
        <Box
          sx={{
            borderRadius: 6,
            width: 560,
            height: 540,
            overflow: "hidden",
            bgcolor: "#D7E3D7",
            display: "flex",
            flexDirection: "column",
          }}
        >
          <WidgetHeader
            toggleModal={toggleModal}
            locale={locale}
            toggleLanguage={toggleLanguage}
          />
          <Box
            sx={{
              flex: 1,
              overflowY: "auto",
              padding: "1rem",
              scrollbarWidth: "none",
              "&::-webkit-scrollbar": {display: "none"},
            }}
          >
            <Outlet context={{locale}}/>
          </Box>
        </Box>
      ) : (
        <IconButton onClick={toggleModal}>
          <WidgetIcon/>
        </IconButton>
      )}
    </Box>
  );
};

export default Widget;

```

Д.2.4.3 Faq

```

import React from "react";
import styles from "./styles.module.scss";
import CButton from "../Button";
import Accordion from "../Accordion/Accordion";

```

```

import {useQuery} from "@tanstack/react-query";
import axios from "axios";
import {useOutletContext, useNavigate} from "react-router-dom";

async function fetchTopMenus(locale) {
  const {data} = await axios.get("/api/faq", {params: {locale}});
  return data;
}

const Faq = () => {
  const navigate = useNavigate();
  const {locale} = useOutletContext();
  const {data, isLoading} = useQuery({
    queryKey: ["faqMenus", locale],
    queryFn: () => fetchTopMenus(locale),
  });

  if (isLoading) return <div>Завантаження...</div>;

  const categories = data.buttons.filter(btn => !btn.url);
  const leaves = data.buttons.filter(btn => !!btn.url);

  return (
    <div className={styles.container}>
      {categories.map((btn, idx) => (
        <CButton
          key={btn.id}
          index={idx}
          action={() => navigate(`/faq/${btn.id}?locale=${locale}`)}
        >
          <span className={styles.btnId}>{idx + 1}</span>
          {btn.name}
        </CButton>
      ))}
      {leaves.length > 0 && (
        <Accordion
          items={leaves.map(leaf => ({
            id: leaf.id,
            title: leaf.name,
            content: leaf.description || "",
            moreLink: leaf.url,
          }))}
          locale={locale}
          startIndex={categories.length}
        />
      )}
    </div>
  );
}

export default Faq;

```

Д.2.4.4 FaqDesc

```

import {useParams, useNavigate, useOutletContext} from "react-router-dom";
import Accordion from "../../components/Accordion/Accordion.jsx";
import CButton from "../../components/Button/index.jsx";
import styles from "./styles.module.scss";
import ArrowLeft from "../../assets/icons/arrowLeft.jsx";
import {useQuery} from "@tanstack/react-query";

```

```

import axios from "axios";

async function fetchChildMenus({parentId, locale}) {
  const url = parentId ? `/api/faq/${parentId}` : "/api/faq";
  const res = await axios.get(url, {params: {locale}});
  return res.data;
}

const FaqDesc = () => {
  const {id} = useParams();
  const navigate = useNavigate();
  const {locale} = useOutletContext();
  const parentId = id || null;

  const {data, isLoading} = useQuery({
    queryKey: ["faqSubmenus", parentId, locale],
    queryFn: () => fetchChildMenus({parentId, locale}),
    enabled: locale != null,
  });

  if (isLoading) return <div>Завантаження...</div>;

  const categories = data.buttons.filter(btn => btn.url == null);
  const leaves = data.buttons.filter(btn => btn.url != null);

  return (
    <div className={styles.wrapper}>
      <button
        className={styles.back}
        onClick={() => {
          if (data.parent_id) {
            navigate(`/faq/${data.parent_id}?locale=${locale}`);
          } else {
            navigate(`/faq?locale=${locale}`);
          }
        }}
      >
        <ArrowLeft/> Повернутись до категорій
      </button>

      <div className={styles.container}>
        {categories.map((cat, idx) => (
          <CButton
            key={cat.id}
            index={idx}
            action={() =>
              navigate(`/faq/${cat.id}?locale=${locale}`)
            }
          >
            {cat.name}
          </CButton>
        ))}

        {leaves.length > 0 && (
          <Accordion
            items={leaves.map((leaf) => ({
              id: leaf.id,
              title: leaf.name,
              content: leaf.description || "",
              moreLink: leaf.url,
            }))}
            locale={locale}
          />

```

```

    })
  </div>
</div>
);
};

export default FaqDesc;

```

Д.2.4.5 Resources

```

import {useState, useEffect, useMemo} from "react";
import {useSearchParams} from "react-router-dom";
import axios from "axios";
import Accordion from "../../components/Accordion/Accordion.jsx";
import SearchIcon from "../../assets/icons/searchIcon.jsx";
import styles from "../../styles.module.scss";

const Resources = () => {
  const [searchParams] = useSearchParams();
  const locale = searchParams.get("locale") || "uk";

  const [tags, setTags] = useState([]);
  const [selectedTag, setSelectedTag] = useState(null);
  const [works, setWorks] = useState([]);
  const [searchTerm, setSearchTerm] = useState("");

  useEffect(() => {
    axios.get("/api/tags")
      .then(res => {
        setTags(res.data);
        if (res.data.length > 0) {
          setSelectedTag(res.data[0].value);
        }
      })
      .catch(console.error);
  }, []);

  useEffect(() => {
    console.log("raw works from server:", works);
    if (selectedTag == null) return;
    axios
      .get("/api/works", {params: {tag: selectedTag}})
      .then(res => {
        setWorks(res.data.data);
      })
      .catch(console.error);
  }, [selectedTag]);

  const filteredWorks = useMemo(() => {
    const term = searchTerm.trim().toLowerCase();

    return (works || [])
      .map(work => ({
        id: work.id,
        title: locale === "en" ? work.en_name : work.ua_name,
        content: locale === "en"
          ? work.en_description
          : work.ua_description,
        moreLink: work.url || null,
      })))
      .filter(item => item.title.toLowerCase().includes(term));
  }, [works, searchTerm]);

```

```

    }, [works, searchTerm, locale]);

const L = {
  uk: {
    placeholder: "Пошук роботи",
    found: "Знайдено",
    one: "результат",
    many: "ів",
  },
  en: {
    placeholder: "Search work",
    found: "Found",
    one: "result",
    many: "s",
  }
}[locale];

return (
  <div className={styles.wrapper}>
    <div className={styles.header}>
      <select
        className={styles.dropdown}
        value={selectedTag ?? ""}
        onChange={e => setSelectedTag(Number(e.target.value))}
      >
        {tags.map(tag => (
          <option key={tag.value} value={tag.value}>
            {locale === "en" ? tag.en_label : tag.ua_label}
          </option>
        ))}
      </select>

      <div className={styles.searchWrapper}>
        <SearchIcon className={styles.searchIcon}/>
        <input
          type="text"
          className={styles.search}
          placeholder={L.placeholder}
          value={searchTerm}
          onChange={e => setSearchTerm(e.target.value)}
        />
      </div>
    </div>

    <div className={styles.count}>
      {L.found} {filteredWorks.length} {" "}
      {filteredWorks.length % 10 === 1 && filteredWorks.length !== 11
        ? L.one
        : L.one + L.many}
    </div>

    <Accordion items={filteredWorks} locale={locale}/>
  </div>
);
};

export default Resources;

```

Д.2.4.6 CButton

```
import styles from "./styles.module.scss";
```

```
import ArrowRight from "../../assets/icons/arrowRight.jsx";

const CButton = ({children, action, id}) => {
  return (
    <button className={styles.btn} onClick={() => action(id)}>

      <div className={styles.btn__container}>
        {children}
      </div>

      <ArrowRight/>
    </button>
  );
};

export default CButton;
```

Д.2.4.7 Accordion

```
import React, {useState} from "react";
import styles from "../styles.module.scss";
import CloseMinus from "../../assets/icons/closeMinus.jsx";
import ArrowDown from "../../assets/icons/arrowDown.jsx";

const Accordion = ({items, locale, startIndex = null}) => {
  const [openId, setOpenId] = useState(null);
  const toggle = (id) => setOpenId((prev) => (prev === id ? null : id));

  const L = {
    uk: {linkText: "Кнопка з посиланням"},
    en: {linkText: "View resource"},
  }[locale];

  const showNumbers = typeof startIndex === "number";

  return (
    <div className={styles.accordion}>
      {items.map((item, idx) => {
        const number = showNumbers ? startIndex + idx + 1 : null;

        return (
          <div
            key={item.id}
            className={` ${styles.item} ${
              openId === item.id ? styles.open : ""
            }`}
          >
            <button
              className={styles.header}
              onClick={() => toggle(item.id)}
            >
              <span className={styles.title}>
                {showNumbers && <span className={styles.number}>{number}</span>}
                {item.title}
              </span>
                {openId === item.id ? <CloseMinus/> : <ArrowDown/>}
            </button>

            {openId === item.id && (
              <div className={styles.content}>
                <p>{item.content}</p>
              </div>
            )}
          </div>
        );
      })}
    </div>
  );
};
```

```

        {item.moreLink && (
          <a href={item.moreLink} className={styles.more}>
            {L.linkText}
          </a>
        )}
      </div>
    )}
  </div>
);
}}
</div>
);
};

export default Accordion;

```

Д.2.4.8 CloseMinus

```

import React from 'react';

const CloseMinus = () => {
  return (
    <svg width="24" height="24" viewBox="0 0 24 24" fill="none"
      xmlns="http://www.w3.org/2000/svg">
      <path
        d="M18 12.998H6C5.73478 12.998 5.48043 12.8926 5.29289 12.7051C5.10536 12.5176 5
        12.2632 5 11.998C5 11.7328 5.10536 11.4784 5.29289 11.2909C5.48043 11.1034 5.73478 10.998 6
        10.998H18C18.2652 10.998 18.5196 11.1034 18.7071 11.2909C18.8946 11.4784 19 11.7328 19 11.998C19 12.2632
        18.8946 12.5176 18.7071 12.7051C18.5196 12.8926 18.2652 12.998 18 12.998Z"
        fill="currentColor"/>
      </svg>
    );
  };
};

export default CloseMinus;

```

Д.2.4.9 ArrowDown

```

const ArrowDown = () => {
  return (
    <svg width="24" height="24" viewBox="0 0 24 24" fill="none"
      xmlns="http://www.w3.org/2000/svg">
      <path d="M7 10L12 15L17 10" stroke="currentColor" strokeWidth="2.5" strokeLinecap="round"
        strokeLinejoin="round"/>
      </svg>
    );
  };
};

export default ArrowDown;

```

Д.2.4.10 SearchIcon

```

import React from 'react';

const SearchIcon = () => {
  return (

```

```

        <svg width="24" height="24" viewBox="0 0 24 24" fill="none"
xmlns="http://www.w3.org/2000/svg">
        <path d="M20.9999 21L16.6599 16.66" stroke="black" strokeWidth="2" strokeLinecap="round"
strokeLinejoin="round"/>
        <path
d="M11 19C15.4183 19 19 15.4183 19 11C19 6.58172 15.4183 3 11 3C6.58172 3 3 6.58172 3
11C3 15.4183 6.58172 19 11 19Z"
stroke="black" strokeWidth="2" strokeLinecap="round" strokeLinejoin="round"/>
        </svg>
    );
};

export default SearchIcon;

```

Д.2.4.11 ArrowLeft

```

const ArrowLeft = () => {
    return (
        <svg width="24" height="24" viewBox="0 0 24 24" fill="none"
xmlns="http://www.w3.org/2000/svg">
        <path d="M14 7L9 12L14 17" stroke="currentColor" strokeWidth="2.5" strokeLinecap="round"
strokeLinejoin="round"/>
        </svg>
    );
};

export default ArrowLeft;

```

Д.2.4.12 ArrowRight

```

const ArrowRight = () => {
    return (
        <svg width="24" height="24" viewBox="0 0 24 24" fill="none"
xmlns="http://www.w3.org/2000/svg">
        <path d="M10 17L15 12L10 7Z" fill="#00CF00"/>
        <path d="M10 17L15 12L10 7" stroke="currentColor" strokeWidth="2.5" strokeLinecap="round"
strokeLinejoin="round"/>
        </svg>
    );
};

export default ArrowRight;

```

Д.3 Blade Admin-panel

Д.3.1 App Blade

```

<!DOCTYPE html>
<html lang="{{ str_replace('_', '-', app()->getLocale()) }}">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,initial-scale=1">
    <!-- CSRF Token -->
    <meta name="csrf-token" content="{{ csrf_token() }}">

```

```

<title>Вхід в систему адміністрування</title>
<link rel="stylesheet" href="{{ mix('css/style.css') }}">
</head>
<body>
@include('layouts.header')

<main>
    @yield('content')
</main>

<script
    src="https://code.jquery.com/jquery-3.7.1.min.js"
    integrity="sha256-/JqT3SQfawRcv/BIHPThkBs0OEvtfFFmqqPF/IYI/Cxo="
    crossorigin="anonymous">
</script>
@yield('scripts')
</body>
</html>

```

Д.3.2 Header Blade

```

<header class="dashboard">
    <div class="dashboard-header">
        <div class="dashboard-header__logo">
            
        </div>
        <div class="dashboard-header__user">
            <span class="dashboard-header__name">Вийти з сторінки адміністратора</span>
            <button class="dashboard-header__logout" type="button" onclick="event.preventDefault();
document.getElementById('logout-form').submit();">
                <svg width="24" height="24" viewBox="0 0 24 24" fill="none"
xmlns="http://www.w3.org/2000/svg">
                    <path d="M5 5H12V3H5C3.9 3 3.9 3 5V19C3 20.1 3.9 21 5 21H12V19H5V5ZM21 12L17
8V11H9V13H17V16L21 12Z" fill="black"/>
                </svg>
            </button>

            <form id="logout-form" action="{{ route('logout') }}" method="POST" class="d-none">
                @csrf
            </form>
        </div>
    </div>
</header>

```

Д.3.3 SignIn Blade

```

<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,initial-scale=1">
    <title>Вхід в систему адміністрування</title>
    <link rel="stylesheet" href="{{ mix('css/style.css') }}">
</head>
<body>
<div class="login-container">

```

```

<h1 class="login-header">Вхід в систему адміністрування</h1>
<p class="login-description">
  Введіть свою електронну адресу та пароль, щоб отримати доступ до свого облікового запису.
</p>
<form action="/login" method="POST" class="login-form" novalidate>
  @csrf
  <div class="form-group" id="group-email">
    <label for="email">Електронна пошта</label>
    <input
      type="email"
      id="email"
      name="email"
      placeholder="vashaposhhta@gmail.com"
      required
      value="{{ old('email') }}"
    />

    @error('email')
    <span class="invalid-feedback" role="alert">
      <strong>{{ $message }}</strong>
    </span>
    @enderror
  </div>
  <div class="form-group" id="group-password">
    <label for="password">Пароль</label>
    <input
      type="password"
      id="password"
      name="password"
      placeholder="*****"
      required
      value="{{ old('password') }}"
    />

    @error('password')
    <span class="invalid-feedback" role="alert">
      <strong>{{ $message }}</strong>
    </span>
    @enderror
  </div>
  <button type="submit" class="button button--width button--100">Увійти</button>
</form>
</div>
<script src="{{ mix('/js/app.js') }}"></script>
</body>
</html>

```

Д.3.4 Buttons Table Blade

```

@extends('layouts.app')

@section('content')
  <main class="dashboard-content">
    <section class="publications">
      <div class="publications__top">
        <h2 class="publications__title">
          Всі публікації <span class="publications__count">{{ $totalPublications }}</span>
        </h2>
        <button class="button button--width button--0">
          <span class="button__icon">

```

```

        <svg width="24" height="24" viewBox="0 0 24 24" fill="none"
xmlns="http://www.w3.org/2000/svg">
        <path
            d="M18 12.998H13V17.998C13 18.2633 12.8946 18.5176 12.7071 18.7052C12.5196
18.8927 12.2652 18.998 12 18.998C11.7348 18.998 11.4804 18.8927 11.2929 18.7052C11.1054 18.5176 11 18.2633
11 17.998V12.998H6C5.73478 12.998 5.48043 12.8927 5.29289 12.7052C5.10536 12.5176 5 12.2633 5 11.998C5
11.7328 5.10536 11.4785 5.29289 11.2909C5.48043 11.1034 5.73478 10.998 6 10.998H11V5.99805C11 5.73283
11.1054 5.47848 11.2929 5.29094C11.4804 5.1034 11.7348 4.99805 12 4.99805C12.2652 4.99805 12.5196 5.1034
12.7071 5.29094C12.8946 5.47848 13 5.73283 13 5.99805V10.998H18C18.2652 10.998 18.5196 11.1034 18.7071
11.2909C18.8946 11.4785 19 11.7328 19 11.998C19 12.2633 18.8946 12.5176 18.7071 12.7052C18.5196 12.8927
18.2652 12.998 18 12.998Z"
            fill="white"/>
        </svg>
        </span>
        <a href="{{ route('admin.pubs.create') }}" class="button__text">Створити публікацію</a>
    </button>
</div>

@if (session('success'))
    <div class="alert-success">
        {{ session('success') }}
    </div>
@endif

<ul class="publications__tabs">
    <li class="publications__tab publications__tab--active" data-filter="Часті питання">Часті
питання</li>
    <li class="publications__tab" data-filter="Академ. ресурси"><a href="{{
route('admin.work.index') }}">Академ. ресурси</a></li>
</ul>

<div class="publications__list">
    @foreach($faqs as $button)
        <div class="publication-item" data-category="Часті питання">
            <div class="publication-item__wrap">
                <h3 class="publication-item__title">
                    {{ Str::limit($button->ua_title, 80, '...') }}
                </h3>
                <div class="publication-item__info">
                    <span class="publication-item__category">id: {{ $button->id }}</span>
                </div>
            </div>
            <div class="publication-item__actions">
                <!-- Кнопка редактирования -->
                <a href="{{ route('admin.pubs.edit', ['pub' => $button->id]) }}"
                    class="action-button action-button--edit">
                    <svg width="40" height="41" viewBox="0 0 40 41" fill="none"
                        xmlns="http://www.w3.org/2000/svg">
                        <rect x="0.5" y="1" width="39" height="39" rx="15.5" fill="white"/>
                        <rect x="0.5" y="1" width="39" height="39" rx="15.5" stroke="#C0C0C0"/>
                        <path
                            d="M14.414 24.39L24.556 14.248L23.142 12.834L13
22.976V24.39H14.414ZM15.243 26.39H11V22.147L22.435 10.712C22.6225 10.5245 22.8768 10.4192 23.142
10.4192C23.4072 10.4192 23.6615 10.5245 23.849 10.712L26.678 13.541C26.8655 13.7285 26.9708 13.9828
26.9708 14.248C26.9708 14.5131 26.8655 14.7674 26.678 14.955L15.243 26.39ZM11
28.39H29V30.39H11V28.39Z"
                            fill="black"/>
                        </svg>
                    </a>

                <!-- Кнопка удаления -->
                <button class="action-button action-button--delete" data-id="{{ $button->id }}"

```

```

        onclick="event.preventDefault();        document.getElementById('delete-form-{{
$button->id }}').submit();">
        <svg width="40" height="41" viewBox="0 0 40 41" fill="none"
        xmlns="http://www.w3.org/2000/svg">
        <rect x="0.5" y="1" width="39" height="39" rx="15.5" fill="white"/>
        <rect x="0.5" y="1" width="39" height="39" rx="15.5" stroke="#C0C0C0"/>
        <path
        d="M15 29.5C14.45 29.5 13.9793 29.3043 13.588 28.913C13.1967 28.5217
13.0007 28.0507 13 27.5V14.5H12V12.5H17V11.5H23V12.5H28V14.5H27V27.5C27 28.05 26.8043 28.521 26.413
28.913C26.0217 29.305 25.5507 29.5007 25 29.5H15ZM25 14.5H15V27.5H25V14.5ZM17
25.5H19V16.5H17V25.5ZM21 25.5H23V16.5H21V25.5Z"
        fill="black"/>
        </svg>
    </button>
    <form id="delete-form-{{ $button->id }}" action="{{ route('admin.pubs.destroy', ['pub'
=> $button->id]) }}" method="POST" style="display: none;">
        @csrf
        @method('DELETE')
    </form>
</div>
</div>
@endforeach

<div class="pagination">
    {{ $faqs->links() }}
</div>
</div>
</section>
</main>
@endsection

@section('scripts')

@endsection

```

Д.3.5 Buttons Create Blade

```

@extends('layouts.app')

@section('content')
    <main>
        <section class="container">
            <h2>Створити Публікацію</h2>

            <form class="create-form" method="post" action="{{ route('admin.pubs.store') }}">
                @csrf
                <fieldset class="form-section">
                    <div class="form-section__wrapper">
                        <h3>Контент публікації</h3>
                    </div>

                    <div class="form-group">
                        <label class="form-group__label" for="ua_title">Заголовок (українською)</label>
                        <input
                            type="text"
                            id="ua_title"
                            name="ua_title"
                            placeholder="Введіть заголовок"
                            required
                            value="{{ old('ua_title') }}"
                        />

```

```

    @error('ua_title')
    <span class="invalid-feedback" role="alert">
      <strong>{{ $message }}</strong>
    </span>
  @enderror
</div>

<div class="form-group">
  <label class="form-group__label" for="en_title">Заголовок (англійською)</label>
  <input
    type="text"
    id="en_title"
    name="en_title"
    placeholder="Enter a title"
    required
    value="{{ old('en_title') }}"
  />
  @error('en_title')
  <span class="invalid-feedback" role="alert">
    <strong>{{ $message }}</strong>
  </span>
  @enderror
</div>

<!-- тепер сам input для URL (спочатку прихований) -->
<div class="form-group" id="link-url-group">
  <label class="form-group__label" for="url">Посилання</label>
  <input
    type="text"
    id="url"
    name="url"
    placeholder="Введіть URL"
    value="{{ old('url') }}"
  />
  @error('url')
  <span class="invalid-feedback" role="alert">
    <strong>{{ $message }}</strong>
  </span>
  @enderror
</div>

<!-- і нове поле під ним — ID батьківської публікації -->
<div class="form-group" id="parent-id-group">
  <label class="form-group__label" for="parent_id">ID батьківської публікації</label>
  <input
    type="text"
    id="parent_id"
    name="parent_id"
    placeholder="Введіть ID батьківської публікації"
    value="{{ old('parent_id') }}"
  />
  @error('parent_id')
  <span class="invalid-feedback" role="alert">
    <strong>{{ $message }}</strong>
  </span>
  @enderror
</div>

<div class="form-group" id="description-group-uk">
  <label class="form-group__label" for="ua_description">Опис (українською)</label>
  <textarea id="ua_description" name="ua_description" rows="5" placeholder="Введіть
опис">{{ old('ua_description') }}</textarea>

```

```

        @error('ua_description')
        <span class="invalid-feedback" role="alert">
            <strong>{{ $message }}</strong>
        </span>
    @enderror
</div>
<div class="form-group" id="description-group-en">
    <label class="form-group__label" for="en_description">Опис (англійською)</label>
    <textarea id="en_description" name="en_description" rows="5" placeholder="Enter a
description">{{ old('en_description') }}</textarea>
    @error('en_description')
    <span class="invalid-feedback" role="alert">
        <strong>{{ $message }}</strong>
    </span>
    @enderror
</div>
</fieldset>

<div>
    <button type="submit" class="button button--width button--2">
        Створити публікацію
    </button>
</div>
</form>
</section>
</main>
@endsection

```

Д.3.6 Buttons Edit Blade

```

@extends('layouts.app')

@section('content')
    <main>
        <section class="container">
            <h2>Редагувати Публікацію</h2>

            <form class="create-form" method="post" action="{{ route('admin.pubs.update', ['pub' => $pub-
>id]) }}">

                @csrf
                @method('PUT')
                <fieldset class="form-section">
                    <div class="form-section__wrapper">
                        <h3>Контент публікації</h3>
                    </div>

                    <div class="form-group">
                        <label class="form-group__label" for="ua_title">Заголовок (українською)</label>
                        <input
                            type="text"
                            id="ua_title"
                            name="ua_title"
                            placeholder="Введіть заголовок"
                            required
                            value="{{ $pub->ua_title ?: old('ua_title') }}"
                        />

                        @error('ua_title')
                        <span class="invalid-feedback" role="alert">
                            <strong>{{ $message }}</strong>
                        </span>
                    </div>
                </fieldset>
            </form>
        </section>
    </main>
@endsection

```

```

    @enderror
  </div>

  <div class="form-group">
    <label class="form-group__label" for="en_title">Заголовок (англійською)</label>
    <input
      type="text"
      id="en_title"
      name="en_title"
      placeholder="Enter a title"
      required
      value="{{ $pub->en_title ?: old('en_title') }}"
    />
    @error('en_title')
    <span class="invalid-feedback" role="alert">
      <strong>{{ $message }}</strong>
    </span>
    @enderror
  </div>

  <!-- тепер сам input для URL (спочатку прихований) -->
  <div class="form-group" id="link-url-group">
    <label class="form-group__label" for="url">Посилання</label>
    <input
      type="text"
      id="url"
      name="url"
      placeholder="Введіть URL"
      value="{{ $pub->url ?: old('url') }}"
    />
    @error('url')
    <span class="invalid-feedback" role="alert">
      <strong>{{ $message }}</strong>
    </span>
    @enderror
  </div>

  <!-- і нове поле під ним — ID батьківської публікації -->
  <div class="form-group" id="parent-id-group">
    <label class="form-group__label" for="parent_id">ID батьківської публікації</label>
    <input
      type="text"
      id="parent_id"
      name="parent_id"
      placeholder="Введіть ID батьківської публікації"
      value="{{ $pub->parent_id ?: old('parent_id') }}"
    />
    @error('parent_id')
    <span class="invalid-feedback" role="alert">
      <strong>{{ $message }}</strong>
    </span>
    @enderror
  </div>

  <div class="form-group" id="description-group-uk">
    <label class="form-group__label" for="ua_description">Опис (українською)</label>
    <textarea id="ua_description" name="ua_description" rows="5" placeholder="Введіть
опис">{{ $pub->ua_description ?: old('ua_description')}}</textarea>
    @error('ua_description')
    <span class="invalid-feedback" role="alert">
      <strong>{{ $message }}</strong>
    </span>
    @enderror
  </div>

```

```

</div>
<div class="form-group" id="description-group-en">
  <label class="form-group__label" for="en_description">Опис (англійською)</label>
  <textarea id="en_description" name="en_description" rows="5" placeholder="Enter a
description">{{ $pub->en_description ?: old('en_description') }}</textarea>
  @error('en_description')
  <span class="invalid-feedback" role="alert">
    <strong>{{ $message }}</strong>
  </span>
  @enderror
</div>
</fieldset>

<div>
  <button type="submit" class="button button--width button--2">
    Оновити публікацію
  </button>
</div>
</form>
</section>
</main>
@endsection

```

Д.3.7 Work Table Blade

```

@extends('layouts.app')

@section('content')
  <main class="dashboard-content">
    <section class="publications">
      <div class="publications__top">
        <h2 class="publications__title">
          Всі публікації <span class="publications__count">{{ $totalPublications }}</span>
        </h2>
        <button class="button button--width button--0">
          <span class="button__icon">
            <svg width="24" height="24" viewBox="0 0 24 24" fill="none"
xmlns="http://www.w3.org/2000/svg">
              <path
                d="M18 12.998H13V17.998C13 18.2633 12.8946 18.5176 12.7071 18.7052C12.5196
18.8927 12.2652 18.998 12 18.998C11.7348 18.998 11.4804 18.8927 11.2929 18.7052C11.1054 18.5176 11 18.2633
11 17.998V12.998H6C5.73478 12.998 5.48043 12.8927 5.29289 12.7052C5.10536 12.5176 5 12.2633 5 11.998C5
11.7328 5.10536 11.4785 5.29289 11.2909C5.48043 11.1034 5.73478 10.998 6 10.998H11V5.99805C11 5.73283
11.1054 5.47848 11.2929 5.29094C11.4804 5.1034 11.7348 4.99805 12 4.99805C12.2652 4.99805 12.5196 5.1034
12.7071 5.29094C12.8946 5.47848 13 5.73283 13 5.99805V10.998H18C18.2652 10.998 18.5196 11.1034 18.7071
11.2909C18.8946 11.4785 19 11.7328 19 11.998C19 12.2633 18.8946 12.5176 18.7071 12.7052C18.5196 12.8927
18.2652 12.998 18 12.998Z"
                fill="white"/>
            </svg>
          </span>
          <a href="{{ route('admin.work.create') }}" class="button__text">Створити публікацію</a>
        </button>
      </div>

      @if (session('success'))
        <div class="alert-success">
          {{ session('success') }}
        </div>
      @endif

      <ul class="publications__tabs">

```

```

        <li class="publications__tab" data-filter="Часті
route('admin.pubs.index') }}">Часті питания</a></li>
        <li class="publications__tab publications__tab--active" data-filter="Академ.
ресурси">Академ. ресурси</li>
    </ul>

    <div class="publications__filters-wrapper">
        <div class="publications__filters">
            <label for="tag-filter" class="visually-hidden"></label>
            <select id="tag-filter" class="filter-select">
                <option value="all">Всі теги</option>

                @foreach(App\Enums\TagEnum::getCases() as $tag)
                    <option value="{{ $tag['value'] }}" {{ request('tag') == $tag['value'] ? 'selected' : " }}"
                        {{ $tag['ua_label'] }}
                    </option>
                @endforeach
            </select>
            <label for="search-filter" class="visually-hidden"></label>
            <div class="search-filter">
                
                <input
                    type="text"
                    id="search-filter"
                    class="search-filter__input"
                    placeholder="Пошук за назвою..."
                    value="{{ request('search') }}"
                />
            </div>
        </div>
    </div>

    <div class="publications__list">
        @foreach($works as $button)
            <div class="publication-item" data-category="Часті питания">
                <div class="publication-item__wrap">
                    <h3 class="publication-item__title">
                        {{ Str::limit($button->ua_name, 80, '...') }}
                    </h3>
                    <div class="publication-item__info">
                        <span class="publication-item__category">id: {{ $button->id }}</span>
                        <span
                            class="publication-item__category">tag:
App\Enums\TagEnum::getById($button->tag)->UaLabel() }}</span>
                        </div>
                    </div>
                    <div class="publication-item__actions">
                        <a href="{{ route('admin.work.edit', ['work' => $button->id]) }}"
                            class="action-button action-button--edit">
                            <svg width="40" height="41" viewBox="0 0 40 41" fill="none"
                                xmlns="http://www.w3.org/2000/svg">
                                <rect x="0.5" y="1" width="39" height="39" rx="15.5" fill="white"/>
                                <rect x="0.5" y="1" width="39" height="39" rx="15.5" stroke="#C0C0C0"/>
                                <path
                                    d="M14.414 24.39L24.556 14.248L23.142 12.834L13
22.976V24.39H14.414ZM15.243 26.39H11V22.147L22.435 10.712C22.6225 10.5245 22.8768 10.4192 23.142
10.4192C23.4072 10.4192 23.6615 10.5245 23.849 10.712L26.678 13.541C26.8655 13.7285 26.9708 13.9828
26.9708 14.248C26.9708 14.5131 26.8655 14.7674 26.678 14.955L15.243 26.39ZM11
28.39H29V30.39H11V28.39Z"
                                    fill="black"/>
                                </svg>
                            </a>

                        <button class="action-button action-button--delete" data-id="{{ $button->id }}"

```

```

        onclick="event.preventDefault();        document.getElementById('delete-form-{{
$button->id }}').submit();">
        <svg width="40" height="41" viewBox="0 0 40 41" fill="none"
            xmlns="http://www.w3.org/2000/svg">
            <rect x="0.5" y="1" width="39" height="39" rx="15.5" fill="white"/>
            <rect x="0.5" y="1" width="39" height="39" rx="15.5" stroke="#C0C0C0"/>
            <path
                d="M15 29.5C14.45 29.5 13.9793 29.3043 13.588 28.913C13.1967 28.5217
13.0007 28.0507 13 27.5V14.5H12V12.5H17V11.5H23V12.5H28V14.5H27V27.5C27 28.05 26.8043 28.521 26.413
28.913C26.0217 29.305 25.5507 29.5007 25 29.5H15ZM25 14.5H15V27.5H25V14.5ZM17
25.5H19V16.5H17V25.5ZM21 25.5H23V16.5H21V25.5Z"
                fill="black"/>
            </svg>
        </button>
        <form id="delete-form-{{ $button->id }}" action="{{ route('admin.work.destroy',
['work' => $button->id]) }}" method="POST" style="display: none;">
            @csrf
            @method('DELETE')
        </form>
    </div>
</div>
@endforeach

<div class="pagination">
    {{ $works->links() }}
</div>
</div>
</section>
</main>
@endsection

@section('scripts')
<script>
    $('#filter-select').on('change', function() {
        var selectedTag = $(this).val();
        if (selectedTag === 'all') {
            window.location.href = '{{ route('admin.work.index') }}';
        } else {
            window.location.href = '{{ route('admin.work.index') }}?tag=' + selectedTag;
        }
    });

    $('#search-filter').on('keydown', function() {
        if (event.key === 'Enter' || event.keyCode === 13) {
            var searchQuery = $(this).val();
            var url = new URL('{{ route('admin.work.index') }}');
            if (searchQuery) {
                url.searchParams.set('search', searchQuery);
            } else {
                url.searchParams.delete('search');
            }
            window.location.href = url.toString();
        }
    });
</script>
@endsection

```

Д.3.8 Work Create Blade

```
@extends('layouts.app')
```

```

@section('content')
<main>
  <section class="container">
    <h2>Створити Публікацію</h2>

    <form class="create-form" method="post" action="{{ route('admin.work.store') }}">
      @csrf
      <fieldset class="form-section">
        <div class="form-section__wrapper">
          <h3>Контент публікації</h3>
        </div>

        <div class="form-group">
          <label class="form-group__label" for="ua_name">Заголовок (українською)</label>
          <input
            type="text"
            id="ua_name"
            name="ua_name"
            placeholder="Введіть заголовок"
            required
            value="{{ old('ua_name') }}"
          />

          @error('ua_name')
          <span class="invalid-feedback" role="alert">
            <strong>{{ $message }}</strong>
          </span>
          @enderror
        </div>

        <div class="form-group">
          <label class="form-group__label" for="en_name">Заголовок (англійською)</label>
          <input
            type="text"
            id="en_name"
            name="en_name"
            placeholder="Enter a title"
            required
            value="{{ old('en_name') }}"
          />

          @error('en_name')
          <span class="invalid-feedback" role="alert">
            <strong>{{ $message }}</strong>
          </span>
          @enderror
        </div>

        <div class="form-group" id="link-url-group">
          <label class="form-group__label" for="url">Посилання</label>
          <input
            type="text"
            id="url"
            name="url"
            placeholder="Введіть URL"
            value="{{ old('url') }}"
          />

          @error('url')
          <span class="invalid-feedback" role="alert">
            <strong>{{ $message }}</strong>
          </span>
          @enderror
        </div>

```

```

<div class="form-group" id="tag-group">
  <label class="form-group__label" for="tag">Оберіть тег</label>
  <div class="select-wrapper">
    <select id="tag" name="tag">
      @foreach(App\Enums\TagEnum::getCases() as $tag)
        <option value="{{ $tag['value'] }}" {{ old('tag') == $tag['value'] ? 'selected' : '' }}>
          {{ $tag['ua_label'] }}
        </option>
      @endforeach
    </select>

    @error('tag')
      <span class="invalid-feedback" role="alert">
        <strong>{{ $message }}</strong>
      </span>
    @enderror
  </div>
</div>

<div class="form-group" id="description-group-uk">
  <label class="form-group__label" for="ua_description">Опис (українською)</label>
  <textarea id="ua_description" name="ua_description" rows="5" placeholder="Введіть
опис">{{ old('ua_description') }}</textarea>
  @error('ua_description')
    <span class="invalid-feedback" role="alert">
      <strong>{{ $message }}</strong>
    </span>
  @enderror
</div>
<div class="form-group" id="description-group-en">
  <label class="form-group__label" for="en_description">Опис (англійською)</label>
  <textarea id="en_description" name="en_description" rows="5" placeholder="Enter a
description">{{ old('en_description') }}</textarea>
  @error('en_description')
    <span class="invalid-feedback" role="alert">
      <strong>{{ $message }}</strong>
    </span>
  @enderror
</div>
</fieldset>

<div>
  <button type="submit" class="button button--width button--2">
    Створити публікацію
  </button>
</div>
</form>
</section>
</main>
@endsection

```

Д.3.9 Work Edit Blade

```

@extends('layouts.app')

@section('content')
  <main>
    <section class="container">
      <h2>Редагувати Публікацію</h2>

```

```

$work->id)) }}">
    <form class="create-form" method="post" action="{ route('admin.work.update', ['work' =>
    @csrf
    @method('PUT')
    <fieldset class="form-section">
        <div class="form-section__wrapper">
            <h3>Контент публікації</h3>
        </div>

        <div class="form-group">
            <label class="form-group__label" for="ua_name">Заголовок (українською)</label>
            <input
                type="text"
                id="ua_name"
                name="ua_name"
                placeholder="Введіть заголовок"
                required
                value="{{ $work->ua_name ?: old('ua_name') }}"
            />

            @error('ua_name')
            <span class="invalid-feedback" role="alert">
                <strong>{{ $message }}</strong>
            </span>
            @enderror
        </div>

        <div class="form-group">
            <label class="form-group__label" for="en_name">Заголовок (англійською)</label>
            <input
                type="text"
                id="en_name"
                name="en_name"
                placeholder="Enter a title"
                required
                value="{{ $work->en_name ?: old('en_name') }}"
            />

            @error('en_name')
            <span class="invalid-feedback" role="alert">
                <strong>{{ $message }}</strong>
            </span>
            @enderror
        </div>

        <!-- тепер сам input для URL (спочатку прихований) -->
        <div class="form-group" id="link-url-group">
            <label class="form-group__label" for="url">Посилання</label>
            <input
                type="text"
                id="url"
                name="url"
                placeholder="Введіть URL"
                value="{{ $work->url ?: old('url') }}"
            />

            @error('url')
            <span class="invalid-feedback" role="alert">
                <strong>{{ $message }}</strong>
            </span>
            @enderror
        </div>

        <div class="form-group" id="tag-group">
            <label class="form-group__label" for="tag">Оберіть тег</label>

```

```

    <div class="select-wrapper">
      <select id="tag" name="tag">
        @foreach(App\Enums\TagEnum::getCases() as $tag)
          <option value="{{ $tag['value'] }}" {{ $work->tag == $tag['value'] ? 'selected' : '' }}>
            {{ $tag['ua_label'] }}
          </option>
        @endforeach
      </select>

      @error('tag')
      <span class="invalid-feedback" role="alert">
        <strong>{{ $message }}</strong>
      </span>
      @enderror
    </div>
  </div>

  <div class="form-group" id="description-group-uk">
    <label class="form-group__label" for="ua_description">Опис (українською)</label>
    <textarea id="ua_description" name="ua_description" rows="5" placeholder="Введіть опис">{{ $work->ua_description ?: old('ua_description') }}</textarea>
    @error('ua_description')
    <span class="invalid-feedback" role="alert">
      <strong>{{ $message }}</strong>
    </span>
    @enderror
  </div>

  <div class="form-group" id="description-group-en">
    <label class="form-group__label" for="en_description">Опис (англійською)</label>
    <textarea id="en_description" name="en_description" rows="5" placeholder="Enter a description">{{ $work->en_description ?: old('en_description') }}</textarea>
    @error('en_description')
    <span class="invalid-feedback" role="alert">
      <strong>{{ $message }}</strong>
    </span>
    @enderror
  </div>
</fieldset>

<div>
  <button type="submit" class="button button--width button--2">
    Оновити публікацію
  </button>
</div>
</form>
</section>
</main>
@endsection

```

ДОДАТОК Е
Слайди презентації

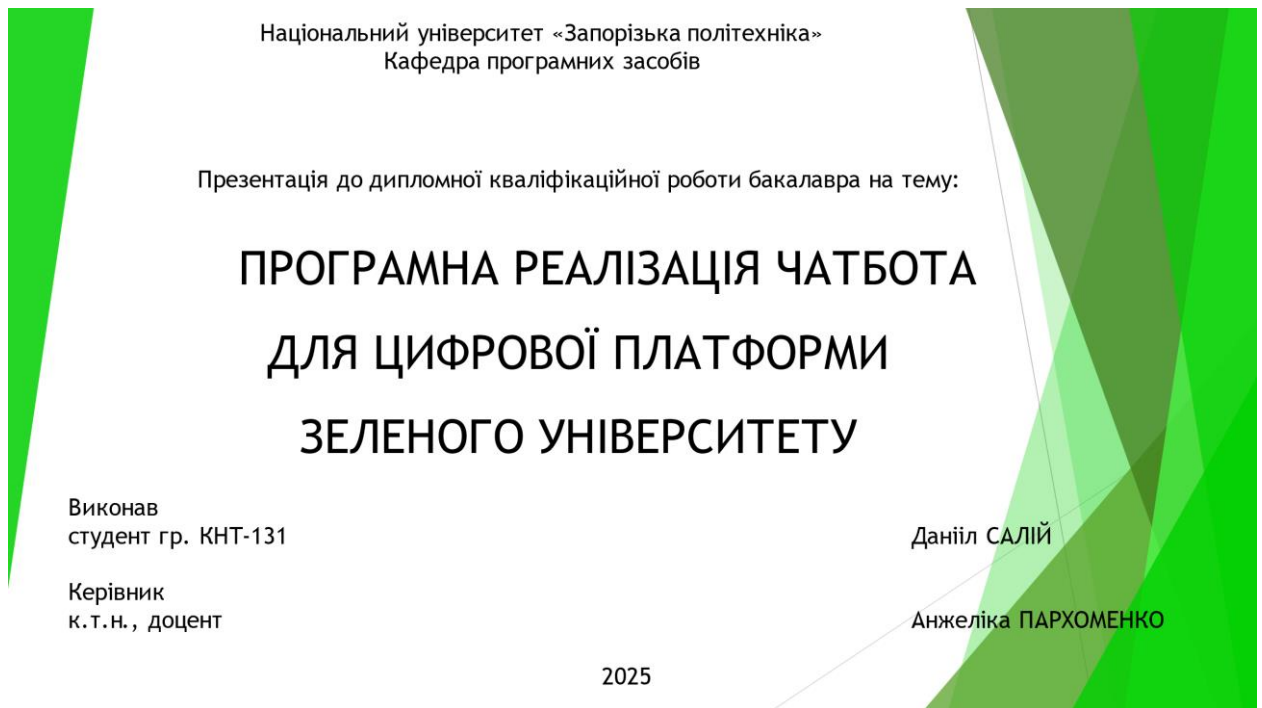


Рисунок Е.1 – Слайд 1 «Титульна сторінка презентації»

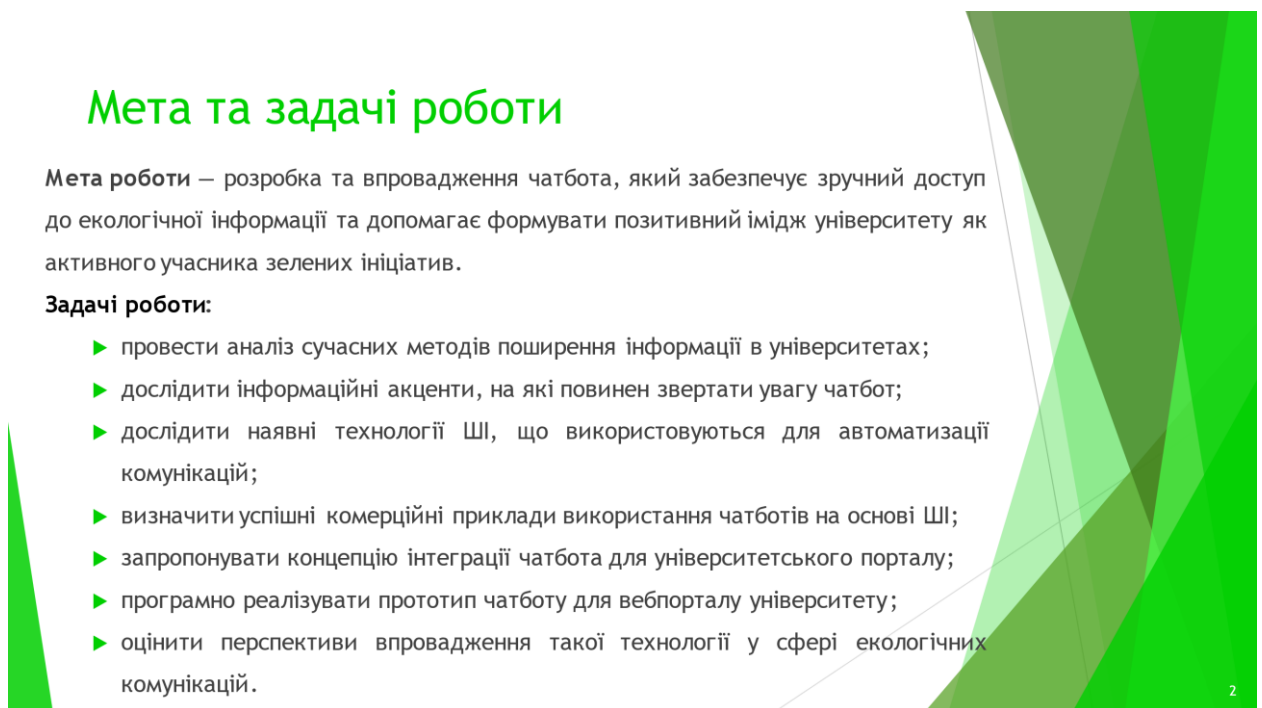
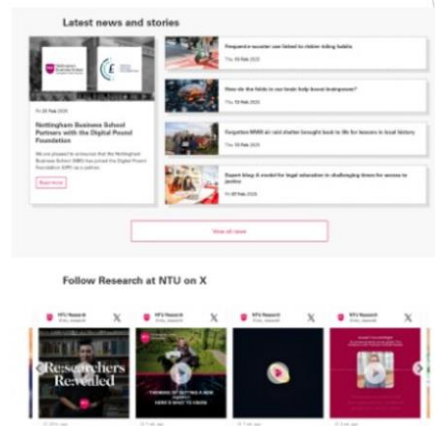


Рисунок Е.2 – Слайд 2 «Реферат презентації»

Аналіз предметної області



Приклад новини на порталі
Національного університету "Львівська
політехніка"



Приклад порталу University of Nottingham
з посиланнями на соц. мережі

Рисунок Е.3 – Слайд 3 «Аналіз предметної області»

Проектування програмного забезпечення

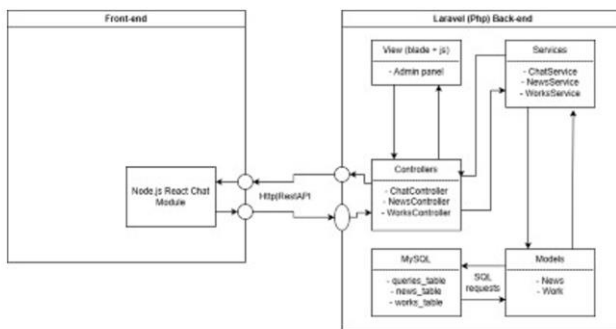
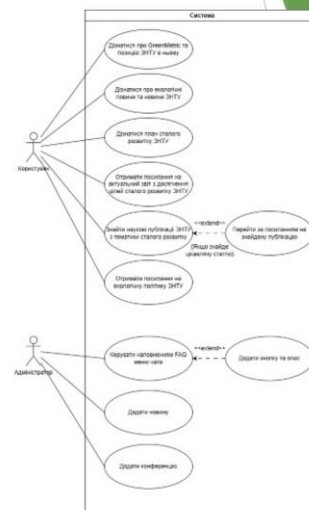


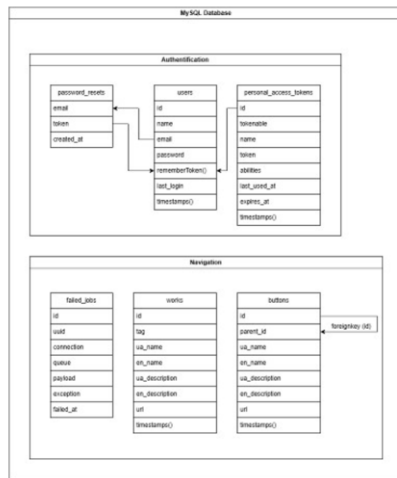
Схема технічної реалізації обраної архітектури



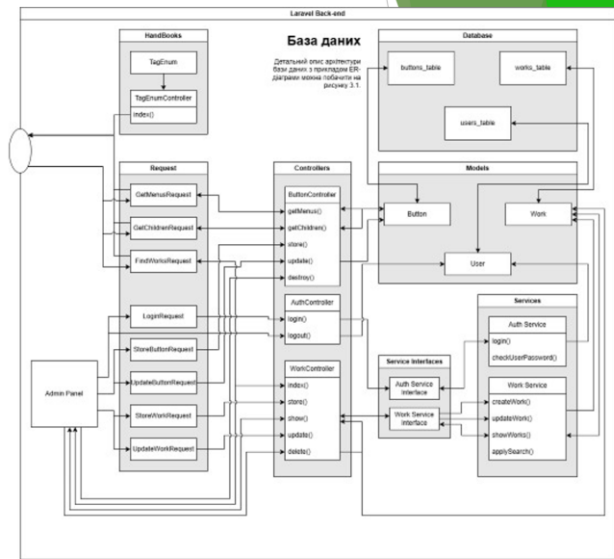
Діаграма прецедентів

Рисунок Е.4 – Слайд 4 «Проектування програмного забезпечення»

Розробка програми



ER-діаграма бази даних

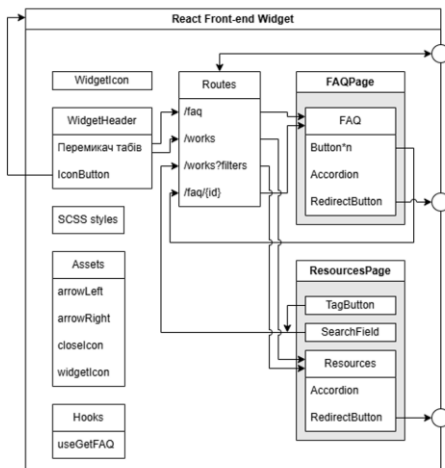


Діаграма класів back-end частини (Laravel)

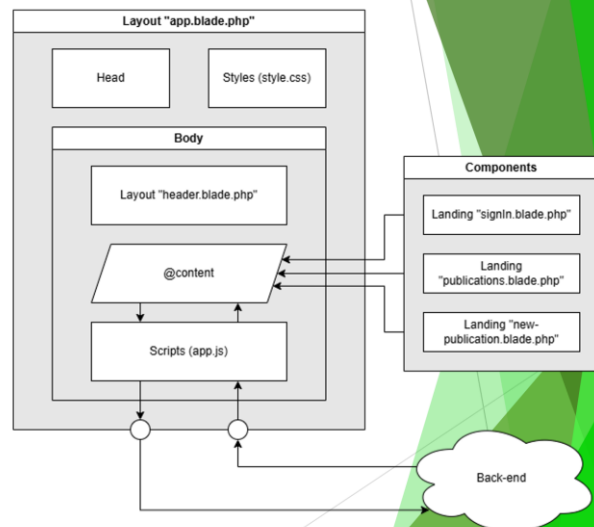
5

Рисунок Е.5 – Слайд 5 «Розробка програми»

Розробка програми



Діаграма компонентів React віджету користувача інтерфейсу



Діаграма компонентів адмінпанелі

6

Рисунок Е.6 – Слайд 6 «Розробка програми»

Розробка програми

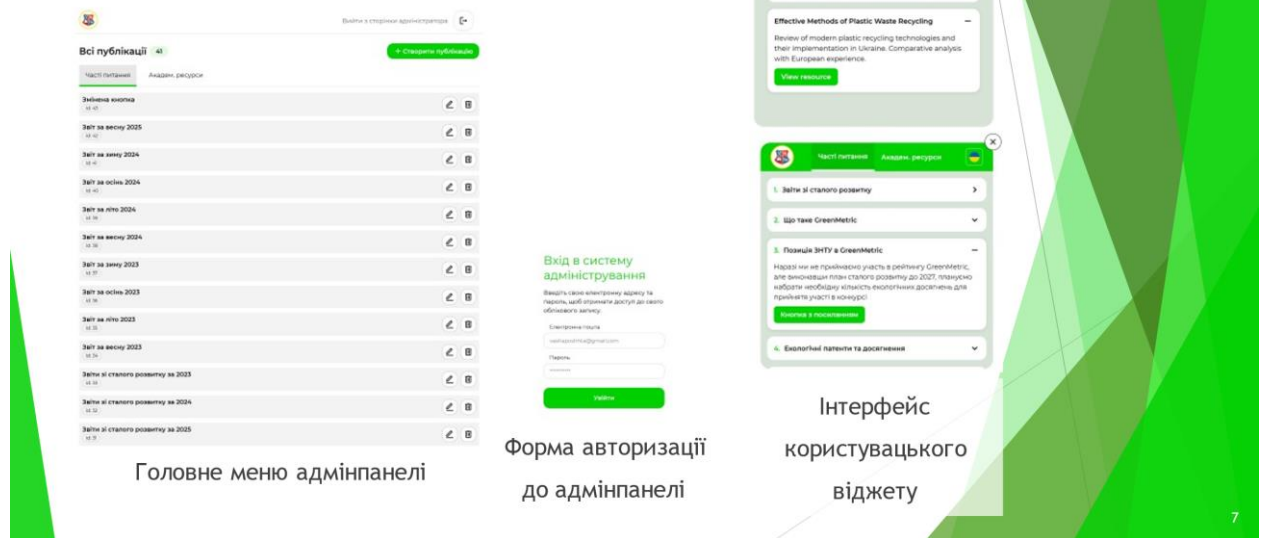


Рисунок Е.7 – Слайд 7 «Розробка програми»

Результат інтеграції чатбота з цифровою платформою Зеленого університету

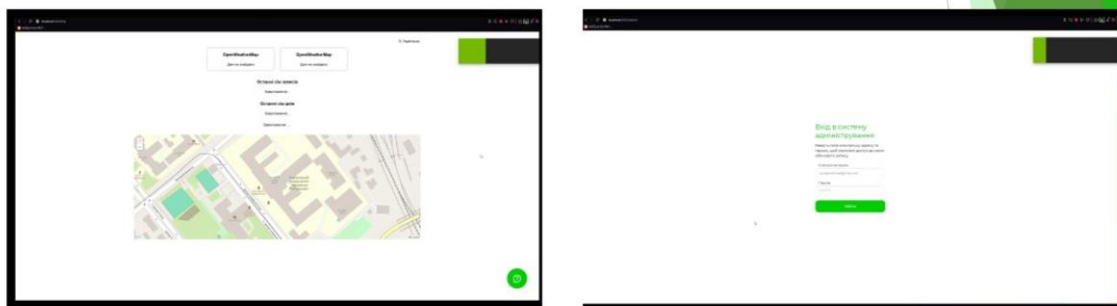


Рисунок Е.8 – Слайд 8 «Розробка програми»

Висновки

- ▶ В результаті виконання роботи створено веборієнтованого чатбота для інформування про екологічну політику університету. Реалізовано інтерактивний віджет на React для кінцевих користувачів і зручну адмінпанель на Blade для керування контентом.
- ▶ Практична цінність полягає в універсальності рішення, його простоті розгортання та можливості масштабування для потреб інших навчальних закладів.
- ▶ Результати роботи було представлено на Щорічній науково-практичній конференції викладачів, науковців, молодих учених, аспірантів та студентів Національного університету «Запорізька політехніка» - «Тиждень науки-2025».

Рисунок Е.9 – Слайд 9 «Висновки»