

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій
(повне найменування інституту, назва факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему РОЗРОБЛЕННЯ ВЕБЗАСТОСУНКУ ДЛЯ ЗБОРУ ТА
АНАЛІЗУ ВІДГУКІВ СТУДЕНТІВ ПРО ВИКЛАДАЧІВ
WEB APPLICATION FOR COLLECTING AND ANALYZING
STUDENT FEEDBACK ABOUT TEACHERS

Виконав: студент 4 курсу, групи КНТ-113сп
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

ДОРОФЄЄВ П.В.

(ПРИЗВИЩЕ та ініціали)

Керівник КОЛПАКОВА Т.О.

(ПРИЗВИЩЕ та ініціали)

Рецензент КОЦУР М.І.

(ПРИЗВИЩЕ та ініціали)

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

Інститут, 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.
С.О. СУББОТІН
“ ” 2026 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ДОРОФЄЄВА Павла Володимировича

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Розроблення вебзастосунку для збору та аналізу відгуків студентів про викладачів.
Web Application for Collecting and Analyzing Student Feedback About Teachers
керівник проєкту (роботи) к.т.н., доцент, КОЛПАКОВА Тетяна Олексіївна,
(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)
затверджені наказом закладу вищої освіти від “07” квітня 2026 року № 139
2. Строк подання студентом проєкту (роботи) 10 червня 2026 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. Аналіз та вибір засобів розробки програмного продукту. 3. Опис програми. 4. Експлуатація, тестування та експериментальне дослідження програми.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	КОЛПАКОВА Т.О., доцент		
Нормоконтроль	КАЛІНІНА М.В., асистент		

7. Дата видачі завдання «20» квітня 2026 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 3
5	Розробка програми.	3–4 тижні	Розділ 3,4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)

(підпис) Павло ДОРОФЄЄВ
(Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

(підпис) Тетяна КОЛПАКОВА
(Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 101 с., 9 табл., 39 рис., 2 дод., 12 джерел.

ВЕБДОДАТОК, ЯКІСТЬ ОСВІТИ, EXPRESS, NODE.JS, MONGODB.

Об'єкт дослідження – процес розроблення програмного забезпечення для збору та аналізу відгуків студентів про викладачів.

Предмет дослідження – вебзастосунки для збору та аналізу відгуків студентів про викладачів

Мета роботи – підвищення швидкості та ефективності збору, обробки й аналізу відгуків студентів про викладачів шляхом розроблення вебзастосунку.

Матеріали, методи та технічні засоби: структурне та об'єктно-орієнтоване програмування, мова гіпертекстової розмітки HTML, мова стилів вебсторінок CSS, база даних MongoDB, мова програмування JS, середовище виконання Node.js, фреймворк Express.

Результати. Розроблено програмний вебзастосунок для збору та аналізу відгуків студентів про викладачів із реалізацією функцій збереження, обробки та перегляду даних користувачами.

Висновки. Розроблено вебзастосунок для збору відгуків студентів про викладачів, який автоматизує та прискорює процес аналізу якості освіти.

Галузь використання – збір та аналіз відгуків студентів про викладачів з метою аналізу якості освіти в Національному університеті «Запорізька політехніка».

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 101 pages, 9 tables, 39 figures, 2 appendixes, 12 sources.

WEB APPLICATION, QUALITY OF EDUCATION, EXPRESS, NODE.JS, MONGODB.

Object of study is the process of developing software for collecting and analyzing student feedback about teachers.

Subject of study are web applications for collecting and analyzing student feedback about teachers.

The purpose of the work is to increase the speed and efficiency of collecting, processing, and analyzing student feedback about teachers by developing a web application.

Materials, methods, and tools: structured and object-oriented programming, hypertext markup language HTML, web page style language CSS, MongoDB database, JS programming language, Node.js runtime environment, Express framework.

Results. The software web application was developed to collect and analyze student feedback about teachers, implementing functions for storing, processing, and viewing data by users.

Conclusions A web application has been developed to collect student feedback on teachers, which automates and accelerates the process of analyzing the quality of education.

The field of use is the process of collection and analysis of student feedback about teachers in order to analyze the quality of education at the National University «Zaporizhzhia Polytechnic».

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	8
Вступ.....	9
1 Аналіз предметної області.....	10
1.1 Огляд сучасного стану питання.....	10
1.2 Аналіз існуючих прикладів.....	11
1.2.1 Rate my proffesors	11
1.2.2 The student room	12
1.2.3 Stud search	13
1.2.4 Порівняння існуючих аналогів	14
1.3 Висновки за розділом 1	15
2 Аналіз та вибір засобів розробки програмного продукту.....	16
2.1 Обґрунтування вибору середовища розробки системи.....	16
2.2 Обґрунтування вибору мов та технологій розробки	16
2.3 Обґрунтування вибору СКБД.....	23
2.4 Висновки за розділом 2	25
3 Опис програми.....	32
3.1 Використані моделі даних.....	27
3.2 Структурна схема програми.....	30
3.3 Розробка модулів системи.....	30
3.4 Висновки за розділом 3	32
4 Експлуатація, тестування та експериментальне дослідження програми.	33
4.1 Інструкція по експлуатації програми.....	33
4.1.1 Характеристики програми.....	33
4.1.2 Звертання до програми	33
4.1.3 Призначення і умови використання програми.....	34
4.1.4 Виконання програми.....	34
4.2 Методика та результати тестування.....	46

4.3 Експериментальне дослідження програми.....	49
4.4 Висновки за розділом 4	51
Висновки	52
Перелік джерел посилання	53
Додаток А Текст програми.....	54
Додаток Б Слайди презентації	95

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

BB	– Black Book;
CPAN	– Comprehensive Perl Archive Network;
CSS	– Cascading Style Sheets;
ER-model	– Entity-Relationship model;
HTML	– HyperText Markup Language;
ID	– ідентифікатор;
JS	– JavaScript;
MIT	– Массачусетський технологічний інститут;
MPA	– Multi Page Application;
npm	– Node Package Manager;
PEAR	– PHP Extension and Application Repository;
PWA	– Progressive Web App;
SPA	– Single Page Application;
WB	– White Book;
W3C	– World Wide Web Consortium;
БД	– база даних;
ЗНО	– зовнішнє незалежне оцінювання;
ПІБ	– прізвище, ім'я, по-батькові;
ПК	– персональний комп'ютер;
СКБД	– система керування базами даних.

ВСТУП

На сучасному етапі розвитку інформаційних технологій вебтехнології стали одним із ключових інструментів створення програмних систем, що забезпечують ефективний доступ до даних, зручну взаємодію користувачів та централізоване керування інформаційними ресурсами.

Розвиток вебтехнологій суттєво вплинув на процеси організації та обробки інформації в установах різного призначення. Завдяки використанню сучасних програмних засобів стало можливим створення інтерактивних інформаційних систем, які забезпечують постійний доступ до необхідних даних через браузер без потреби встановлення спеціалізованого програмного забезпечення на кожному комп'ютері користувача. Це відкриває широкі перспективи для впровадження інформаційних систем у сфері освіти.

Заклади вищої освіти активно застосовують вебсистеми для забезпечення ефективної взаємодії між студентами, викладачами та адміністрацією. Особливого значення набувають вебзастосунки, які дозволяють збирати, зберігати та аналізувати відгуки студентів про якість викладання. Такі системи сприяють підвищенню прозорості освітнього процесу, забезпечують оперативне отримання зворотного зв'язку та створюють умови для постійного вдосконалення навчального процесу.

Метою даної дипломної кваліфікаційної роботи є розробка вебзастосунку для збору, зберігання та публікації відгуків студентів про викладачів університету. Для досягнення поставленої мети було виконано аналіз існуючих аналогів, визначено функціональні та нефункціональні вимоги до системи, розроблено архітектуру програмного забезпечення, а також здійснено програмну реалізацію вебзастосунку із використанням сучасних вебтехнологій. Результатом роботи є програмний продукт, що забезпечує зручне введення коментарів, їх модерацію та подальший перегляд користувачами.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд сучасного стану проблеми

У сучасних закладах вищої освіти важливу роль відіграє систематичний збір зворотного зв'язку від студентів щодо якості викладання. Отримані відгуки дозволяють оцінити рівень педагогічної майстерності викладачів, виявити сильні сторони освітнього процесу та своєчасно виявляти проблеми. Традиційні методи анкетування, що базуються на паперових формах або електронних таблицях, поступово замінюються спеціалізованими вебзастосунками.

Вебзастосунки для збору відгуків забезпечують централізоване зберігання інформації. Такі системи дозволяють зменшити адміністративне навантаження на працівників навчальних закладів і значно прискорюють процес аналізу даних. Завдяки доступу через браузер користувачі можуть працювати із системою з будь-якого пристрою, підключеного до мережі Інтернет.

Однією з ключових вимог до таких систем є забезпечення анонімності студентів. Анонімний формат збору відгуків підвищує об'єктивність оцінювання та сприяє отриманню більш щирих коментарів. Крім того, важливо забезпечити захист персональних даних, обмеження доступу до адміністративних функцій та резервне копіювання інформації [1].

Для реалізації вебзастосунків широко використовуються сучасні технології, серед яких HyperText Markup Language (HTML), Cascading Style Sheets (CSS), JavaScript (JS), Node.js та Express. Для побудови користувацького інтерфейсу можуть застосовуватися шаблонізатори, наприклад EJS, а також фронтенд-фреймворки. Для зберігання інформації використовуються бази даних, такі як MongoDB.

Основними перевагами вебзастосунків є:

- доступність з будь-якого пристрою;
- централізоване зберігання інформації;

- зручність оновлення функціоналу;
- підтримка багатокористувацької роботи;
- можливість інтеграції з іншими інформаційними системами.

1.2 Аналіз існуючих прикладів

1.2.1 Rate my professors

Rate my professors є одним із найвідоміших вебресурсів для оцінювання викладачів закладів вищої освіти. Сервіс функціонує з 1999 року та містить мільйони відгуків від студентів різних університетів.

Сервіс дозволяє студентам залишати відгуки, виставляти оцінки та аналізувати складність курсів.

Аналіз архітектури сайту, інтерфейсу користувача показав, що окремі функціональні рішення не є оптимальними для зручності і зрозумілості вмісту для відвідувачів сайту.

До переваг належать великий обсяг даних, простий інтерфейс та зручний пошук.

Недоліками є орієнтація на закордонні університети, а також відсутність адаптації до структури конкретного університету.

Система не передбачає розділення позитивних і негативних відгуків на окремі категорії, що ускладнює аналіз інформації.

Вебресурс Rate my professors зображено на рисунку 1.1.

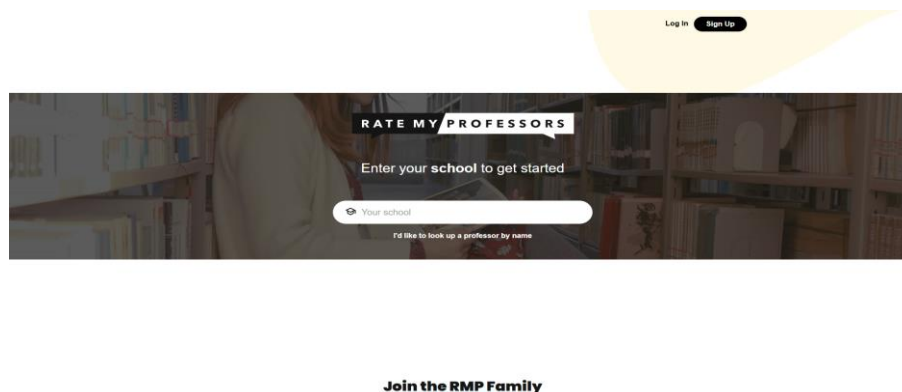


Рисунок 1.1 – Головне вікно Rate my professors [2]

1.2.2 The student room

The student room є однією з найбільших освітніх вебресурсів Великої Британії. Платформа поєднує форум та тематичні розділи для студентів.

Обговорення викладачів відбувається у форматі форумних тем, де користувачі можуть вільно ділитися враженнями та рекомендаціями.

Перевагою ресурсу є велика кількість активних учасників, завдяки чому можна знайти багато інформації щодо різних освітніх компонентів, наприклад навчальних дисциплін та викладачів.

Платформа дозволяє ставити уточнювальні запитання та отримувати відповіді від інших студентів, що сприяє більш повному висвітленню проблем.

Недоліком є відсутність чіткої структури. Інформація часто дублюється або розташовується в різних розділах. Це створює відчуття хаосу для нового користувача. Пошук релевантних відгуків може потребувати значного часу. Сторінка є перевантаженою для зору. Також візуал і структурна сайту не є інтуїтивно зрозумілими, а функціональні частини не мають додаткового пояснення свого призначення, що робить користування сайтом не зручним.

Вебресурс The student room зображено на рисунку 1.2.

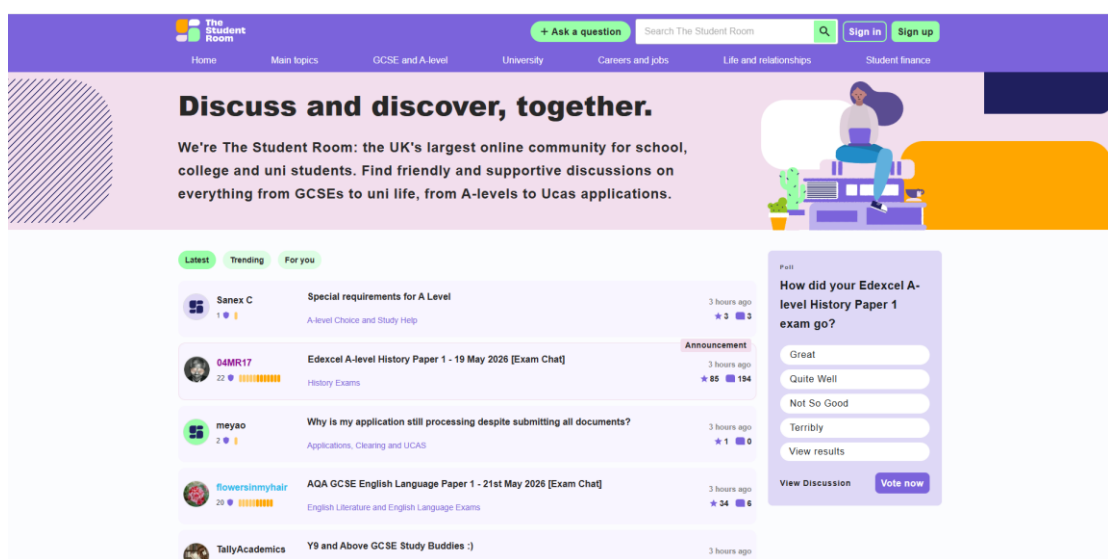


Рисунок 1.2 – Головне вікно The student room [3]

1.2.3 Stud search

Stud search є спеціалізованим вебресурсом для пошуку інформації про заклади освіти та ознайомлення з відгуками студентів.

Основним функціоналом платформи є можливість спілкування між абітурієнтом і студентом. Для цього на сайті є можливість реєстрації і налаштування акаунту. Також платформа надає можливість переглядати рейтинги університетів за кількістю вступників і середнім балом ЗНО (зовнішнього незалежного оцінювання) серед вступників.

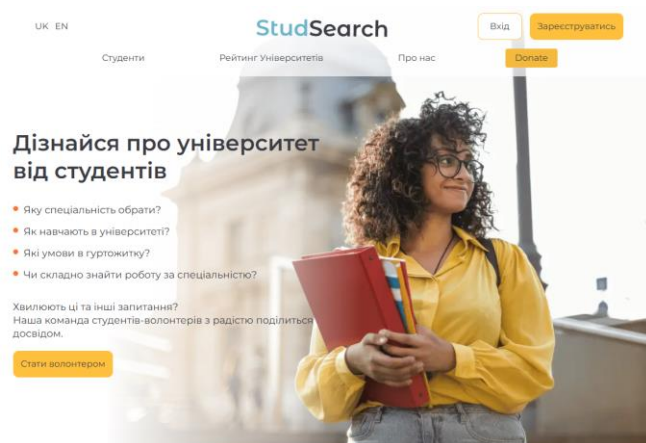
Сайт має багато фільтрів для пошуку потрібного закладу освіти і конкретного студента.

До переваг належать можливість прямого спілкування зі студентом, який готовий відповісти на конкретні запитання, структуроване подання інформації та простота використання.

Недоліками є відсутність великої кількості користувачів, обмежені функції для оцінки якості освіти та необхідність прямого спілкування між людьми, тобто гарантії своєчасного отримання інформації.

Система не реалізує зберігання інформації, що робить неможливим релевантний аналіз якості освіти.

Вебресурс Stud search зображено на рисунку 1.3.



Як працює StudSearch?

Рисунок 1.3 – Головне вікно Stud search [4]

1.2.4 Порівняння існуючих аналогів

Проведений аналіз показав, що розглянуті системи реалізують різні підходи до збору та представлення інформації.

Rate my professors забезпечує структуровану систему оцінювання та зручний інтерфейс, але недостатньо адаптований до потреб конкретних університетів.

The student room надає широкі можливості для обговорення, проте не має конкретної структури даних і ускладнює швидкий пошук інформації.

Stud search пропонує функціонал лише для абітурієнтів. Також не підтримує зберігання відгуків, що робить не можливим релевантний аналіз якості освіти.

Жоден із розглянутих ресурсів не містить одночасно поділ відгуків на позитивні та негативні, гнучку систему модерації та адаптацію сервісу до структури конкретного закладу освіти. Наглядне порівняння наведено у таблиці 1.1.

Таблиця 1.1 – Таблиця порівняння існуючих аналогів

Критерій	Rate My Professors	The Student Room	Stud Search
Структурованість даних	Висока	Низька	Низька
Зручність пошуку інформації	Висока	Низька	Низька
Поділ відгуків на позитивні та негативні	Ні	Ні	Ні
Можливість адаптації під конкретний університет	Обмежена	Відсутня	Відсутня
Система модерації контенту	Базова	Обмежена	Відсутня
Можливість швидкого аналізу інформації	Так	Ні	Ні
Наглядність для кожного зацікавленого у навчальному процесі конкретного закладу освіти	Частково	Ні	Ні

Отже, доцільною є розробка спеціалізованого вебзастосунку, який враховує виявлені недоліки та реалізує функції, необхідні для ефективного збору і аналізу студентських відгуків.

1.3 Висновки за розділом 1

Проведений аналіз існуючих аналогів показав, що жодне з розглянутих рішень не забезпечує повного набору функцій, необхідних для ефективного збору та аналізу відгуків студентів про викладачів у межах конкретного університету. Основними недоліками відсутність чіткого поділу позитивних і негативних коментарів, обмежені можливості модерації та недостатня структурованість інформації. Створення нового вебзастосунку є доцільним, оскільки дозволить реалізувати систему, яка враховує специфіку освітнього процесу, забезпечує зручний інтерфейс для студентів та адміністратора, ефективне адміністрування та надійне зберігання даних.

2 АНАЛІЗ ТА ВИБІР ЗАСОБІВ РОЗРОБКИ ПРОГРАМНОГО ПРОДУКТУ

Основною метою дипломної роботи є створення вебзастосунку, який дозволяє студентам залишати та переглядати відгуки про викладачів, а адміністраторам керувати системою. Вебдодаток повинен бути простим у використанні, безпечним та придатним до подальшого розширення. Для досягнення цієї мети варто використовувати сучасні вебтехнології.

2.1 Обґрунтування вибору середовища розробки системи

Для розробки програмного продукту було обрано середовище розробки Microsoft Visual Studio Code.

Програма Visual Studio Code — сучасний редактор програмного коду, призначений для створення, редагування та налагодження вебзастосунків, серверних програм і хмарних рішень. Середовище розробки поширюється безкоштовно та підтримує основні операційні системи, зокрема Windows, Linux і macOS. Завдяки кросплатформовості та невисоким системним вимогам Visual Studio Code набув широкого поширення серед розробників програмного забезпечення.

Редактор був представлений компанією Microsoft у 2015 році та став першим кросплатформовим продуктом у родині Visual Studio. Програмний продукт побудований на основі технологій Electron, Chromium та Node.js, що забезпечує поєднання вебтехнологій із можливостями настільних застосунків. Такий підхід дозволив створити гнучке та розширюване середовище розробки, яке підтримує велику кількість мов програмування та сучасних фреймворків.

Visual Studio Code містить вбудований налагоджувач, засоби інтеграції з системою контролю версій Git, інструменти рефакторингу, автодоповнення коду, контекстні підказки та розвинену систему розширень. Завдяки підтримці розширень користувач може адаптувати середовище до потреб конкретного

проєкту, додаючи підтримку нових мов програмування, шаблонів, засобів тестування та аналізу коду.

Середовище широко використовується для розробки застосунків на основі Node.js, Express, MongoDB та інших технологій сучасного веброзроблення. Воно підтримує роботу з мовами JavaScript, TypeScript, HTML, CSS, JSON, PHP, Python, Java, C#, C++, Markdown та багатьма іншими. Поєднання зручного інтерфейсу, високої продуктивності та широких можливостей налаштування робить Visual Studio Code одним із найпопулярніших інструментів для створення веборієнтованих інформаційних систем [5].

2.2 Обґрунтування вибору мов та технологій розробки

Мова HTML (HyperText Markup Language) є стандартизованою мовою гіпертекстової розмітки, призначеною для створення та відображення вебсторінок у браузері. HTML-документи передаються від вебсервера до клієнтського браузера за протоколами HTTP або HTTPS, а також можуть відкриватися безпосередньо з локального носія. Після завантаження браузер аналізує структуру документа, інтерпретує теги розмітки та формує графічний інтерфейс сторінки, який відображається на екрані користувача.

Мову HTML розробив британський учений Tim Berners-Lee у період з 1989 по 1991 рік під час роботи в CERN у Geneva. Первинною метою створення HTML було забезпечення зручного обміну науковою та технічною документацією між дослідниками. Мова базувалася на принципах SGML, однак була значно спрощена завдяки використанню обмеженого набору структурних і семантичних елементів, відомих як теги. Такий підхід дозволив створювати документи з чіткою логічною структурою та підтримкою гіпертекстових посилань.

Основною перевагою HTML є простота використання та універсальність. За допомогою цієї мови можна описувати заголовки, абзаци,

списки, таблиці, форми, зображення та гіперпосилання. Подальший розвиток стандарту забезпечив підтримку мультимедійних елементів, таких як аудіо, відео та графічна анімація. У поєднанні з CSS і JavaScript мова HTML є фундаментом сучасних вебтехнологій і використовується для створення як простих інформаційних сайтів, так і складних вебзастосунків [6].

Мова CSS (Cascading Style Sheets) є формальною мовою опису зовнішнього вигляду документів, створених за допомогою мов розмітки, насамперед HTML та XHTML. Вона використовується для визначення параметрів відображення елементів вебсторінки, таких як кольори, шрифти, відступи, позиціонування та анімація. CSS може застосовуватися також до інших XML-документів, зокрема SVG та XUL. Завдяки використанню таблиць стилів забезпечується відокремлення структури документа від його візуального представлення, що значно спрощує розробку та супровід вебзастосунків.

До появи CSS оформлення вебсторінок здійснювалося безпосередньо засобами HTML, що призводило до дублювання коду та ускладнювало зміну дизайну. Запровадження CSS дало можливість централізовано визначати стилі для великої кількості сторінок, забезпечуючи єдиний підхід до оформлення та швидке оновлення зовнішнього вигляду вебресурсу. Такий підхід підвищує повторне використання коду, покращує читабельність документів і сприяє дотриманню принципів структурованого вебпроектування.

CSS належить до групи вебстандартів, розроблених та підтримуваних World Wide Web Consortium (W3C). Ідею каскадних таблиць стилів запропонував Håkon Wium Lie у 1994 році, а подальший розвиток мови відбувався спільно з Bert Bos. Однією з ключових особливостей CSS є механізм каскадування та успадкування, за допомогою якого дочірні елементи можуть автоматично отримувати властивості батьківських елементів. Це дозволяє створювати гнучкі та масштабовані системи стилів для складних вебінтерфейсів.

Сучасні версії CSS підтримують адаптивний дизайн, анімацію, трансформації, використання змінних, Flexbox, Grid Layout та інші потужні засоби компоновання елементів. У поєднанні з HTML та JavaScript мова CSS є однією з базових технологій розробки інтерфейсів користувача і широко застосовується для створення інформаційних систем, корпоративних порталів та інтерактивних вебзастосунків [6].

Мова JavaScript є динамічною, об'єктно-орієнтованою мовою програмування, що реалізує специфікацію ECMAScript і широко використовується для створення інтерактивних вебзастосунків. Основне призначення JavaScript полягає у виконанні сценаріїв на стороні клієнта, що дозволяє забезпечувати взаємодію з користувачем, змінювати структуру та зовнішній вигляд вебсторінок, керувати поведінкою браузера та здійснювати асинхронний обмін даними з сервером без перезавантаження сторінки. Завдяки цим можливостям JavaScript є однією з ключових технологій сучасного веброзроблення поряд з HTML і CSS.

JavaScript належить до прототипно-орієнтованих мов програмування та підтримує декілька парадигм, зокрема імперативне, функціональне та об'єктно-орієнтоване програмування. Серед основних характеристик мови слід виділити динамічну та слабку типізацію, автоматичне керування пам'яттю, прототипне наслідування та використання функцій як об'єктів першого класу. Такі особливості забезпечують високу гнучкість розробки та дозволяють створювати масштабовані програмні рішення різного рівня складності.

Сфера застосування JavaScript значно розширилася порівняно з початковим призначенням. Мова використовується для написання сценаріїв вебсторінок, створення односторінкових Single Page Application (SPA) та прогресивних вебзастосунків Progressive Web App (PWA), серверного програмування за допомогою платформи Node.js, розробки настільних застосунків на базі Electron, мобільних програм із використанням React Native, а також для автоматизації вбудованих сценаріїв у різних програмних

продуктах. Завдяки універсальності JavaScript став основою повноцінної екосистеми для розробки інформаційних систем.

Упродовж останніх років JavaScript залишається однією з найпопулярніших мов програмування у світі. Значний вплив на її розвиток справила поява технології AJAX, яка забезпечила асинхронну взаємодію клієнта з сервером і стала основою для створення динамічних вебінтерфейсів. Подальший розвиток стандарту ECMAScript, особливо починаючи з версії ES6, додав класи, модулі, стрілкові функції, проміси та інші засоби, що суттєво підвищили продуктивність і зручність програмування. На базі JavaScript було створено численні бібліотеки та фреймворки, серед яких React, Angular, Vue.js та Express.js, які широко використовуються для створення сучасних веборієнтованих систем [7].

Платформа Node.js є середовищем виконання з відкритим вихідним кодом, призначеним для створення високопродуктивних мережових і серверних застосунків мовою JavaScript. Засновником платформи є Ryan Dahl, який представив її у 2009 році. До появи Node.js мова JavaScript використовувалася переважно для виконання сценаріїв у веббраузері на стороні клієнта. Запровадження Node.js надало можливість запускати JavaScript-код на сервері, що перетворило цю мову на універсальний інструмент для розробки як клієнтських, так і серверних компонентів вебзастосунків.

Основою Node.js є високопродуктивний рушій Google V8, який компілює JavaScript безпосередньо у машинний код. Завдяки цьому забезпечується висока швидкість виконання програм та ефективне використання системних ресурсів. Платформа підтримує модульну архітектуру та надає широкий набір вбудованих бібліотек для роботи з файловою системою, мережею, потоками даних, криптографією та іншими системними компонентами.

Для обробки великої кількості одночасних запитів у Node.js використовується асинхронна подієво-орієнтована модель виконання.

Неблокуючі операції введення-виведення дозволяють одному процесу ефективно обслуговувати тисячі клієнтських підключень без створення окремого потоку для кожного з них. Такий підхід особливо ефективний для вебсерверів, REST API, чатів, систем реального часу та інших мережевих застосунків із високим рівнем паралелізму.

Для керування залежностями та встановлення модулів використовується пакетний менеджер `npm` (Node Package Manager), який є однією з найбільших екосистем програмних бібліотек у світі. За допомогою `npm` розробники можуть швидко інтегрувати готові модулі, автоматизувати збірку проєкту та підтримувати актуальність використовуваних компонентів. У поєднанні з фреймворками, такими як `Express.js`, та базами даних, зокрема `MongoDB`, `Node.js` широко застосовується для створення масштабованих інформаційних систем і сучасних вебзастосунків [8].

`Node Package Manager (npm)` є стандартним менеджером пакунків для середовища виконання `Node.js` та одним із ключових компонентів сучасної екосистеми `JavaScript`. `npm` забезпечує встановлення, оновлення, видалення та керування бібліотеками й інструментами, необхідними для розробки програмного забезпечення. Система складається з клієнта командного рядка та централізованого реєстру пакетів, який містить мільйони відкритих і приватних модулів, доступних для використання в різноманітних проєктах.

`npm` був створений `Isaac Z. Schlueter` як засіб ефективного управління залежностями у `JavaScript`-проєктах. Основною метою розробки було спрощення повторного використання програмних компонентів і автоматизація процесу їх інтеграції. За концепцією `npm` подібний до таких систем, як `PHP Extension and Application Repository (PEAR)` для `PHP` та `Comprehensive Perl Archive Network (CPAN)` для `Perl`, однак орієнтований на специфіку `JavaScript` і платформу `Node.js`. Завдяки `npm` розробники отримали зручний механізм пошуку, інсталяції та оновлення сторонніх модулів.

Менеджер пакетів підтримує роботу як із локальними залежностями конкретного проєкту, так і з глобально встановленими інструментами

командного рядка. Конфігурація залежностей зберігається у файлі `package.json`, де визначаються назва проєкту, його параметри та перелік необхідних пакетів. За допомогою однієї команди `npm install` можна автоматично встановити всі залежності, необхідні для запуску застосунку. Це суттєво спрощує перенесення проєкту між різними середовищами розробки та забезпечує відтворюваність процесу складання.

У `npm` використовується принцип семантичного версонування (Semantic Versioning), який дозволяє задавати допустимі діапазони версій для кожної залежності. Такий підхід дає змогу автоматично отримувати сумісні оновлення, зменшуючи ризик виникнення помилок через несумісність між компонентами. Завдяки широкому вибору бібліотек та високому рівню автоматизації `npm` є невід'ємним інструментом розробки сучасних вебзастосунків, побудованих на основі `Node.js`, `Express` та інших технологій `JavaScript` [9].

`Express.js`, або скорочено `Express`, є мінімалістичним програмним каркасом для розробки серверної частини вебзастосунків на платформі `Node.js`. Фреймворк поширюється як вільне програмне забезпечення з відкритим вихідним кодом за ліцензією Массачусетського технологічного інституту (MIT) та призначений для створення вебсайтів, `RESTful API` та інших мережових застосунків. Завдяки простоті використання, високій продуктивності та гнучкості `Express.js` фактично став стандартним рішенням для серверної розробки в екосистемі `Node.js`.

Автором `Express.js` є TJ Holowaychuk, який створив цей фреймворк, надихаючись архітектурними принципами `Ruby`-фреймворка `Sinatra`. Основною ідеєю було створення легкого та мінімалістичного інструменту, який надає базові механізми маршрутизації, обробки `HTTP`-запитів і формування відповідей, але при цьому легко розширюється за допомогою додаткових модулів і проміжного програмного забезпечення (`middleware`).

`Express.js` забезпечує зручні засоби для визначення маршрутів, роботи з параметрами запиту, обробки форм, взаємодії з шаблонізаторами та

організації логіки застосунку. Однією з ключових особливостей фреймворка є підтримка middleware — функцій, які послідовно обробляють запити та відповіді, реалізуючи аутентифікацію, логування, перевірку даних, обробку помилок та інші службові задачі. Така архітектура сприяє модульності, повторному використанню коду та зручному масштабуванню системи [10].

Порівняння існуючих технологій для розробки вебсистем наведено в таблиці 2.1.

Таблиця 2.1 – Таблиця порівняння мов програмування

Критерій	JavaScript	PHP	Python	Java	C#
Типізація	Динамічна	Динамічна	Динамічна	Статична	Статична
Виконання	Клієнт/сервер	Сервер	Сервер	Сервер	Сервер
Продуктивність	Висока	Середня	Висока	Дуже висока	Дуже висока
Масштабованість	Висока	Середня	Висока	Дуже висока	Дуже висока
Сфера використання	Вебінтерфейси, API	Сайти та CMS	Вебсистеми, AI	Корпоративні системи	Корпоративні системи
Ефективність у Frontend	Дуже висока	Низька	Низька	Низька	Низька
Ефективність у Backend	Висока	Висока	Висока	Дуже висока	Дуже висока

2.3 Обґрунтування вибору СКБД

MongoDB — це документо-орієнтована система керування базами даних з відкритим вихідним кодом, яка належить до класу NoSQL-рішень і

призначена для зберігання та обробки великих обсягів структурованих і напівструктурованих даних. На відміну від традиційних реляційних систем керування базами даних (СКБД), MongoDB не потребує жорсткого визначення схеми таблиць, що забезпечує високу гнучкість під час розробки програмних систем. База даних займає проміжне положення між простими сховищами типу «ключ–значення» та повнофункціональними реляційними системами, поєднуючи високу продуктивність, масштабованість і потужні засоби формування запитів.

MongoDB була створена компанією MongoDB Inc., а її програмний код реалізовано мовою C++. Дані зберігаються у форматі BSON (Binary JSON), який є бінарним представленням JSON-документів. Такий підхід забезпечує природну інтеграцію з мовами програмування, що працюють із об'єктами та структурами даних, зокрема JavaScript та Node.js. Кожен документ у колекції може мати власну структуру, що дозволяє легко модифікувати модель даних без складних процедур міграції схеми.

Система підтримує широкий набір функціональних можливостей, включаючи створення індексів, складні запити, агрегацію даних, зберігання великих бінарних об'єктів, журналювання операцій, реплікацію та резервування. Для обробки великих обсягів інформації MongoDB використовує механізми горизонтального масштабування за допомогою шардінгу, що дозволяє розподіляти дані між кількома серверами. У поєднанні з реплікацією це забезпечує високу доступність, автоматичне відновлення після збоїв і відсутність єдиної точки відмови [11].

Mongoose — це бібліотека об'єктно-документного моделювання (ODM, Object Data Modeling) для середовища Node.js, яка забезпечує зручний механізм взаємодії між застосунком на JavaScript та базою даних MongoDB. Бібліотека надає схему-орієнтований підхід до опису структури документів, що дозволяє формалізувати модель даних, визначати типи полів, встановлювати обмеження та реалізовувати бізнес-логіку на рівні моделей.

Основною перевагою Mongoose є використання схем (Schema), за допомогою яких описується структура документів у колекціях MongoDB. На основі схем створюються моделі (Model), що надають об'єктно-орієнтований інтерфейс для виконання операцій створення, пошуку, оновлення та видалення даних. Такий підхід спрощує організацію коду, підвищує його читабельність і забезпечує централізоване керування логікою доступу до даних [12].

Порівняння існуючих СКБД для наведено в таблиці 2.2.

Таблиця 2.2 – Таблиця порівняння СКБД

Критерій	MySQL	PostgreSQL	MongoDB	Oracle Database
Тип БД	Реляційна	Реляційна	NoSQL (документна)	Реляційна
Ліцензія	Open Source	Open Source	Open Source/Commercial	Commercial
Мова запитів	SQL	SQL	MQL	PL/SQL
Шардінг	Обмежено	Через розширення	Вбудований	Підтримується
Складність адміністрування	Низька	Середня	Середня	Висока
Основна перевага	Простота	Надійність	Гнучкість схеми	Максимальна функціональність
Основний недолік	Менше можливостей	Складніше налаштування	Менш жорстка структура	Дуже висока вартість

2.4 Висновки за розділом 2

JavaScript є найкращою мовою для розробки вебзастосунка для збору відгуків студентів про викладачів. Тому що це єдина мова, яка ефективно використовується як на стороні клієнта, так і на стороні сервера, що дозволяє застосовувати єдиний стек технологій під час розробки всієї системи. Використання платформи Node.js дає змогу створювати високопродуктивні серверні застосунки, а велика екосистема бібліотек npm значно прискорює процес розробки. Крім того, JavaScript характеризується високою продуктивністю, широкими можливостями масштабування та значною популярністю у сфері веброботки.

Базою даних для розробленої системи було обрано MongoDB за її високу продуктивність, можливість горизонтального масштабування та гнучкість документо-орієнтованої моделі бази даних. На відміну від реляційних СКБД, MongoDB не вимагає жорстко визначеної структури таблиць, що дозволяє легко модифікувати дані під час розвитку проєкту. Важливою перевагою є використання формату BSON, який є розширенням JSON та добре інтегрується з мовою JavaScript і платформою Node.js, що використовуються для реалізації вебзастосунку. Це спрощує обмін інформацією між клієнтською частиною, сервером і базою даних. Сукупність цих характеристик забезпечує високу швидкодію, простоту розробки та масштабованість системи, що робить MongoDB оптимальним вибором для зберігання даних розробленого вебзастосунку.

3 ОПИС ПРОГРАМИ

3.1 Використані моделі даних

Для розробки даного програмного продукту використано документо-орієнтовану систему керування базами даних MongoDB. Модель даних було побудовано шляхом перетворення моделі «сутність-зв'язок» бази даних.

Одним з найбільш зручних способів представлення даних, незалежно від програмного забезпечення за допомогою якого буде реалізоване зберігання даних є модель «сутність-зв'язок» (entity-relationship model (ER-model), яка використовується для аналізу предметної області та забезпечує наочне та зрозуміле представлення структури даних, що значно спрощує процес проєктування інформаційної системи.

Модель «сутність-зв'язок» ґрунтується на деякій інформації про реальний світ і призначена для логічного представлення даних. Вона визначає значення даних їх взаємозв'язок з іншими даними, використовуваними в системі. Модель «сутність-зв'язок» представляється в графічному вигляді і називається діаграмою «сутність-зв'язок». На ній представлені усі виявлені сутності предметної області і вказані взаємозв'язки між ними. Такий підхід дозволяє виявити ключові елементи системи ще до початку програмної реалізації та забезпечити цілісність майбутньої структури даних.

Спроектована ER-діаграма документо-орієнтованої бази даних зображена на рисунку 3.1.

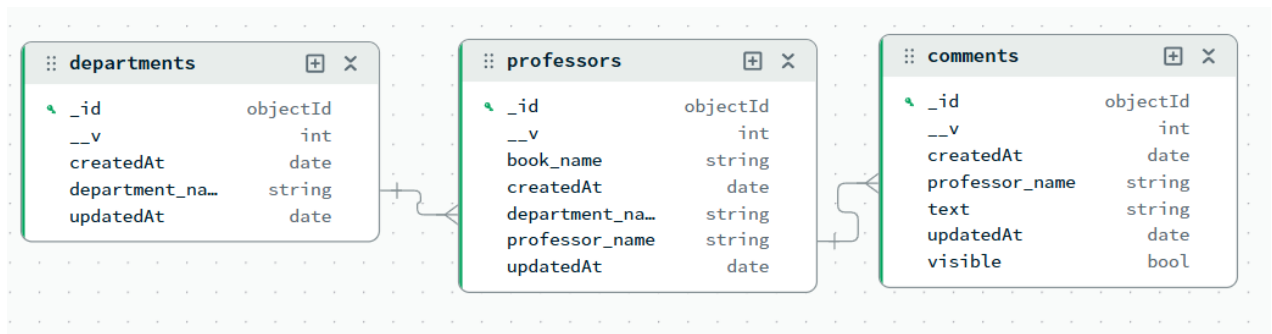


Рисунок 3.1 – ER-діаграма бази даних

В процесі проектування даного програмного продукту для побудови документо-орієнтованої моделі було виділено наступні об'єктні множини: Факультети, Професори, Коментарі.

Базу даних розроблено в форматі NOSQL за допомогою документо-орієнтованої системи керування базами даних MongoDB. База даних вміщує три колекції.

Сутність колекції «departments» (ідентифікатор (ID), createdAt, department_name, updatedAt). Сутність таблиці призначена для зберігання даних про факультети університету. У реалізації бази даних сутність представлена колекцією «departments» (табл. 3.1).

Таблиця 3.1 – Структура колекції «departments»

Назва поля	Тип поля	Опис
ID	ObjectId	Призначено для унікальності записів у полі
createdAt	Time	Призначено для зберігання часу створення документа
department_name	String	Призначено для зберігання назв факультетів
updatedAt	Time	Призначено для зберігання часу останнього редагування документа

Сутність таблиці «professors» (ID, createdAt, department_name, professor_name, updatedAt). Сутність таблиці «professors» призначена для зберігання даних про викладачів університету. У реалізації бази даних сутність представлена колекцією «professors» (таблиця 3.2).

Таблиця 3.2 – Структура колекції «professors»

Назва поля	Тип поля	Опис
ID	ObjectId	Призначено для унікальності записів у полі
createdAt	Time	Призначено для зберігання часу створення документа
department_name	String	Призначено для зберігання назви факультету, до якого належить викладач
professor_name	String	Призначено для зберігання прізвища, ім'я, по-батькові (ПІБ) викладачів
updatedAt	Time	Призначено для зберігання часу останнього редагування документа

Сутність таблиці «comments» (ID, createdAt, professor_name, text, updatedAt, visible). Сутність таблиці «comments» призначена для зберігання даних про коментарі до викладачів університету. У реалізації бази даних сутність представлена колекцією «comments» (таблиця 3.3).

Таблиця 3.3 – Структура колекції «comments»

Назва поля	Тип поля	Опис
ID	ObjectId	Призначено для унікальності записів у полі
createdAt	Time	Призначено для зберігання часу створення документа
professor_name	String	Призначено для зберігання ПІБ викладачів
text	String	Призначено для зберігання тексту коментаря
updatedAt	Time	Призначено для зберігання часу останнього редагування документа
visible	Boolean	Призначено для відмітки про проходження коментарем модерації

3.2 Структурна схема програми

Структурна схема програми представлена на рисунках 3.2 і 3.3.



Рисунок 3.2 – Структурна схема програми



Рисунок 3.3 – Структурна схема програми

3.3 Розробка модулів системи

Розроблена програма складається з наступних модулів, призначення кожного з модулів представлено у таблиці 3.4.

Таблиця 3.4 – Опис модулів програми

Назва файлу	Опис модулю
index	Головна сторінка програми
wb	Сторінка керування білої книги
wbform	Форма для залишення білих(позитивних) коментарів
wblist	Сторінка вибору факультета у білій книзі
wbdepartment	Сторінка вибору викладача у білій книзі
wbprofessor	Сторінка перегляду коментарів до обраного викладача у білій книзі
bb	Сторінка керування чорної книги
bbform	Форма для залишення чорних(негативних) коментарів
bblist	Сторінка вибору факультета у чорній книзі
bbdepartment	Сторінка вибору викладача у чорній книзі
bbprofessor	Сторінка перегляду коментарів до обраного викладача у чорній книзі
admin	Сторінка авторизації до адмін панелі
departments	Сторінка перегляду списку факультетів
department	Сторінка перегляду складу факультету
add-department	Сторінка додавання нового факультету
edit-department	Сторінка редагування факультету
professors	Сторінка перегляду списку викладачів
professor	Сторінка перегляду коментарів до викладача
add-professor	Сторінка додавання нового викладача
edit-professor	Сторінка редагування викладача
comments	Сторінка перегляду списку коментарів
add-comment	Сторінка додавання нового коментаря
edit-comments	Сторінка редагування коментаря

3.4 Висновки за розділом 3

Проведений процес проєктування системи показав, що для вирішення поставленої мети, а саме для ефективного збору та аналізу відгуків студентів про викладачів у межах конкретного університету, потрібна база даних з 3 колекціями. Програма має 23 модулі кожен з яких є важливою частиною структурної схеми програми. Використання такого рішення дозволяє нам вирішити такі недоліки існуючих аналогів: відсутність чіткого поділу позитивних і негативних коментарів, обмежені можливості модерації та недостатня структурованість інформації. Проведене проєктування системи дозволяє реалізувати систему, яка враховує специфіку освітнього процесу, забезпечує зручний інтерфейс для студентів та адміністратора, ефективно адміністрування та надійне зберігання даних.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Інструкція по експлуатації програми

4.1.1 Характеристики програми

Метою створення програмного продукту є розроблення сучасного вебзастосунку, призначеного для збору, зберігання та аналізу відгуків студентів про викладачів закладу вищої освіти. Розроблений сайт є швидким, зручним і простим у використанні, що дозволяє користувачам без зайвих труднощів знаходити потрібних викладачів за факультетами, кафедрами та прізвищем, переглядати позитивні та негативні відгуки, а також формувати власну думку щодо якості викладання. Інтерфейс вебзастосунку побудовано у вигляді багатосторінкового додатка – Multi Page Application (MPA), що забезпечує логічне розділення функціональних модулів та зручну навігацію між сторінками. Для розробки використано технології HTML, CSS та JavaScript, а серверну частину реалізовано на основі Node.js, Express.js, MongoDB та Mongoose. Система забезпечує повноцінну роботу з базою даних, включаючи додавання, редагування, перегляд і видалення інформації про факультети, викладачів та студентські відгуки, що робить її ефективним інструментом для оцінювання якості освітнього процесу.

4.1.2 Звертання до програми

Інтерфейс завантажується завдяки локальному серверу. Для початку роботи необхідно ввести у пошуковий рядок браузеру `http://localhost:3000/`. База даних має назву BlackAndWhite.

Програміст може змінити програму, відкривши програми за допомогою Microsoft Studio Code, або відкрити файли з розширеннями .js, .ejs, .css у будь-якому текстовому редакторі та змінити програмний код.

4.1.3 Призначення і умови використання програми

Метою створення програмного продукту є розроблення вебзастосунку для збору та аналізу відгуків студентів про викладачів.

Програма повинна реалізовувати такі функції:

- доступ до колекцій бази даних;
- додавання інформації в колекції;
- редагування інформації в колекціях;
- видалення інформації з колекцій;

Для роботи програми на персональному комп'ютері користувача повинен бути встановлений Google Chrome або інший сучасний браузер. Для роботи Google Chrome потрібний персональний комп'ютер з мікропроцесором Intel Pentium 4 або пізнішої версії з підтримкою SSE3, оперативна пам'ять 512 Мб і вище, вільного простору на жорсткому диску 350 Мб.

Для встановлення програмного продукту необхідно мати встановленою кореневу папку програми з усім вмістом.

4.1.4 Виконання програми

Для запуску додатку потрібно відкрити браузер за адресою “localhost:3000”, після чого на екрані з'явиться головне вікно програми, яке представлено на рисунку 4.1.

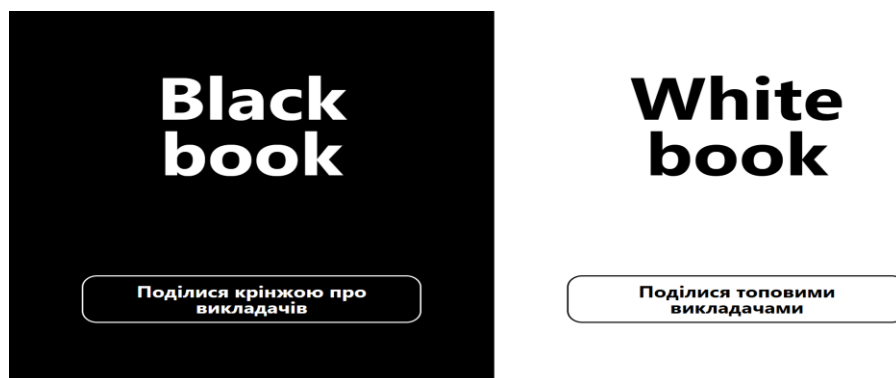


Рисунок 4.1 – Вигляд головного вікна програми

На цьому вікні користувач обирає до якої з двох книг білої(позитивної) і чорної(негативної) він хоче увійти. Якщо він обрав білу книгу то йому необхідно натиснути на кнопку «Поділися топовими викладачами» – він перейде на сторінку, зображену на рисунку 4.2.

White book

Заповнити форму

Переглянути свій факультет

Якщо твій викладач створює неймовірну атмосферу під час навчання - він/вона мусить бути в White book!

Рисунок 4.2 – Вигляд сторінки керування білої книги

На цій сторінці користувач обирає заповнити форму або перейти до перегляду коментарів. Якщо він обрав заповнити форму то йому необхідно натиснути на кнопку «Заповнити форму» – користувач перейде на сторінку, зображену на рисунку 4.3.

Викладач:

Тягунова Марія Юріївна

Коментар:

Залишити коментар

Рисунок 4.3 – Вигляд форми для залишення коментарів

На цьому вікні користувач обирає викладача та пише позитивний коментар. Після цього, для того, щоб коментар був направлений на перевірку адміном йому необхідно натиснути на кнопку «Залишити коментар» – на верху сторінки з’явиться повідомлення про підтвердження дії на сторінці з кнопкою «ОК», після натискання на яку він повернеться до головної сторінки, як це зображено на рисунку 4.4.

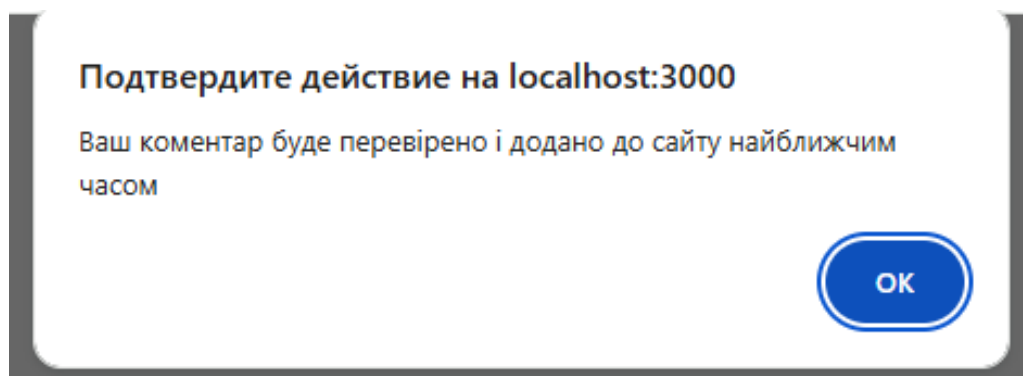


Рисунок 4.4 – Вигляд повідомлення про підтвердження готовності залишити коментар

Якщо на сторінці керування білою книгою користувач обрав перейти до перегляду коментарів то йому необхідно натиснути на кнопку «Переглянути свій факультет» – він перейде на сторінку, зображену на рисунку 4.5.

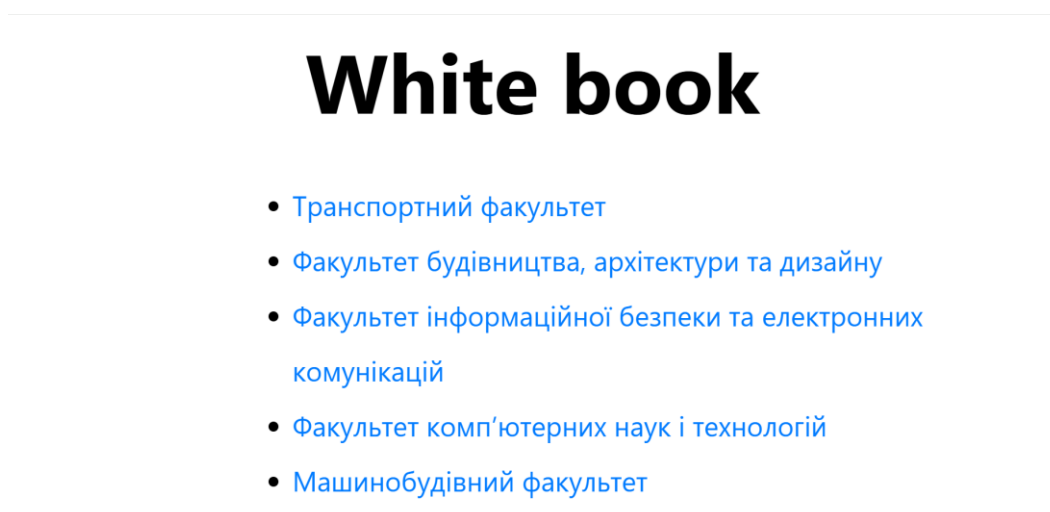
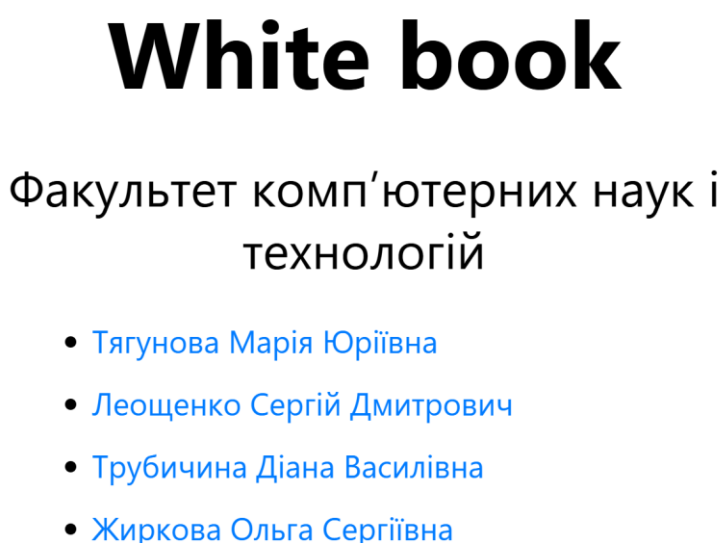


Рисунок 4.5 – Вигляд сторінки зі списком всіх факультетів

На цій сторінці користувач обирає факультет, до якого відноситься цікавий для нього викладач. Для того щоб обрати треба натиснути на назву факультету – користувач перейде на сторінку з усіма викладачами, що відносяться до цього факультету і знаходяться у білій книзі, зображену на рисунку 4.6. Унизу сторінки знаходиться кнопка «Назад», натискання на яку поверне користувача до головної сторінки програми.



White book

Факультет комп'ютерних наук і технологій

- Тягунова Марія Юріївна
- Леощенко Сергій Дмитрович
- Трубочина Діана Василівна
- Жиркова Ольга Сергіївна

Рисунок 4.6 – Вигляд сторінки зі списком всіх викладачів, що відносяться до обраного факультету і знаходяться у білій книзі

На цій сторінці користувач обирає викладача, коментарі до якого його цікавлять. Для того щоб обрати треба натиснути на ім'я викладача – користувач перейде на сторінку, зображену на рисунку 4.7. Унизу сторінки знаходиться кнопка «Назад», натискання на яку поверне користувача до головної сторінки програми.

White book

Колпакова Тетяна Олексіївна

- Завжди ввічливо спілкується зі студентами та надає зрозумілі відповіді на всі запитання.

10.05.2026

- Іноді в години великого навантаження доводиться трохи довше чекати на відповідь або оформлення документів.

10.05.2026

Рисунок 4.7 – Вигляд сторінки зі списком усіх коментарів до обраного викладача

На цьому вікні користувач бачить список усіх коментарів до обраного викладача з датою, коли цей коментар був залишений на сайті. Для повернення до головної сторінки потрібно натиснути на кнопку «Назад», яка знаходиться унизу списку.

Якщо на головній сторінці сайту користувач обрав чорну книгу то йому необхідно натиснути на кнопку «Поділися крінжою про викладачів» – він перейде на сторінку, зображену на рисунку 4.8.

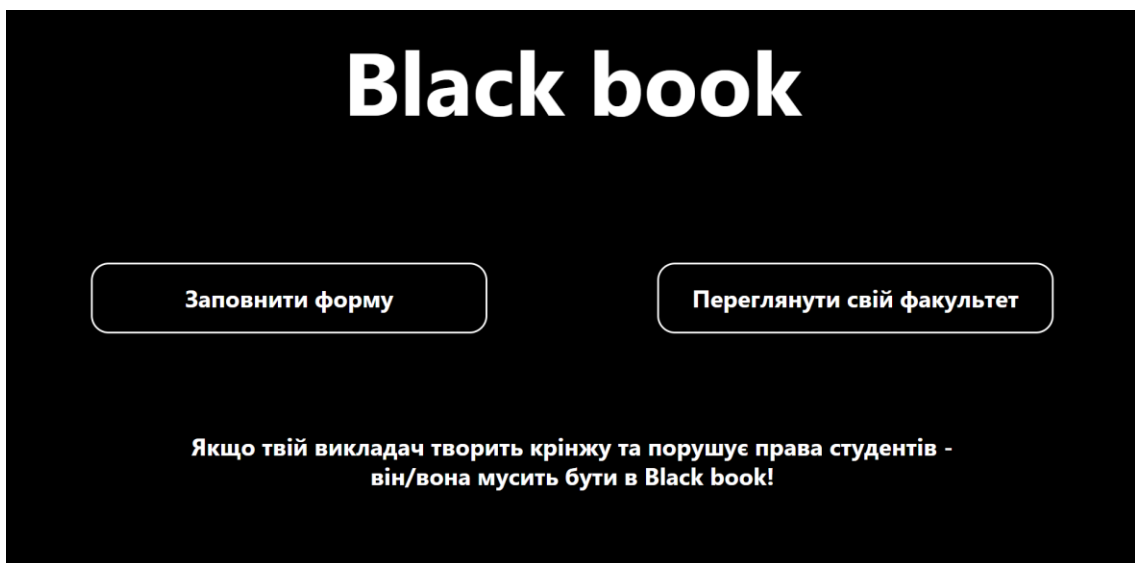
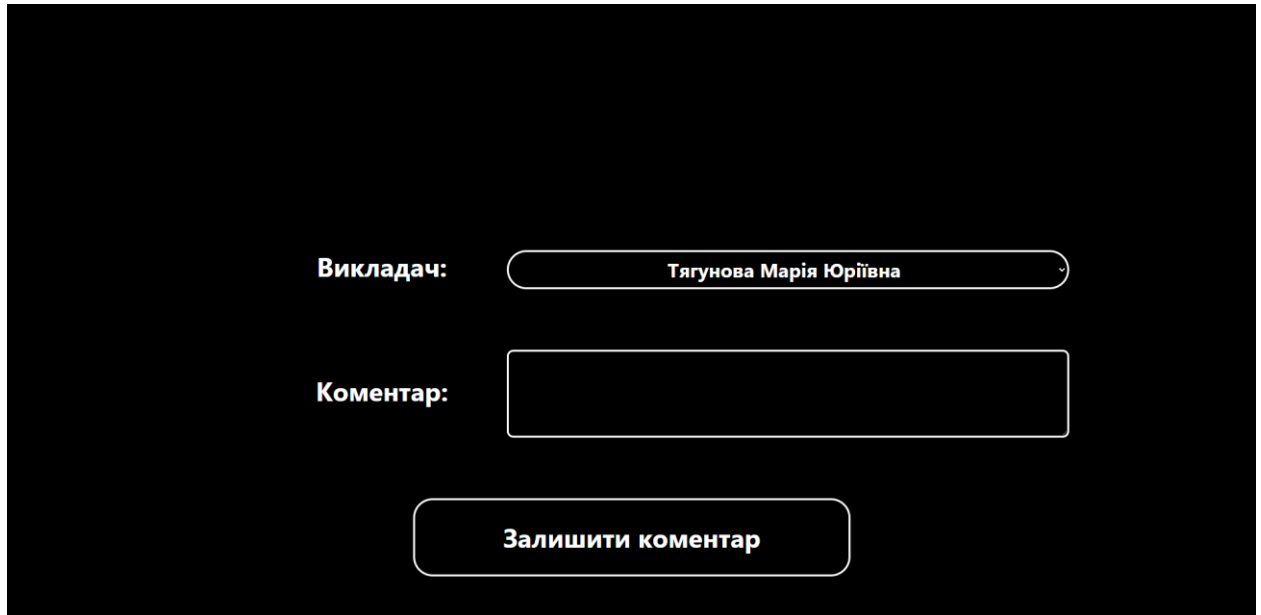


Рисунок 4.8 – Вигляд сторінки керування чорної книги

На цій сторінці користувач обирає заповнити форму або перейти до перегляду коментарів. Якщо він бажає заповнити форму, то йому необхідно натиснути на кнопку «Заповнити форму» – він перейде на сторінку, зображену на рисунку 4.9.



Викладач: Тягунова Марія Юріївна

Коментар:

Залишити коментар

Рисунок 4.9 – Вигляд форми для залишення коментарів

На цьому вікні користувач обирає викладача та пише негативний коментар. Після цього, для того, щоб коментар був направлений на перевірку адміном, користувачу необхідно натиснути на кнопку «Залишити коментар» – на верху сторінки з’явиться повідомлення про підтвердження дії на сторінці з кнопкою «ОК», після натискання на яку він повернеться до головної сторінки, як було зображено на рисунку 4.4.

Якщо на сторінці керування чорної книгою користувач обрав перейти до перегляду коментарів то йому необхідно натиснути на кнопку «Переглянути свій факультет» – він перейде на сторінку, зображену на рисунку 4.10.

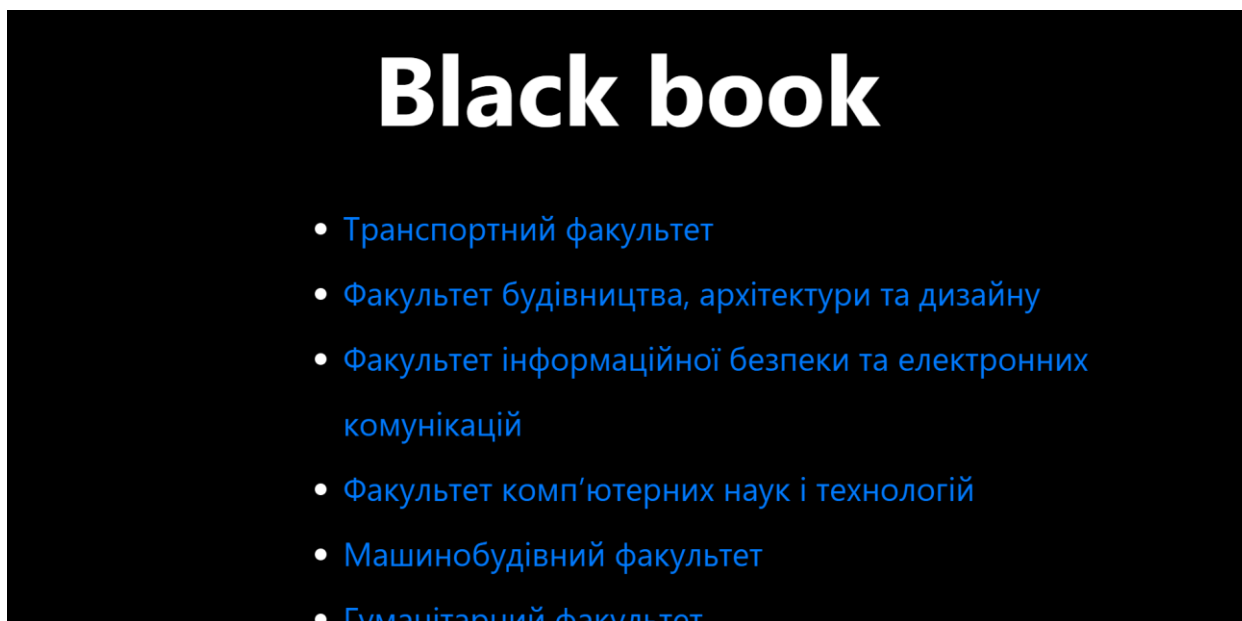


Рисунок 4.10 – Вигляд сторінки зі списком всіх факультетів

На цій сторінці користувач обирає факультет, до якого відноситься цікавий для нього викладач. Для того, щоб обрати, треба натиснути на назву факультету – користувач перейде на сторінку з усіма викладачами, що відносяться до цього факультету і знаходяться у чорній книзі, зображену на рисунку 4.11. Унизу сторінки знаходиться кнопка «Назад», натискання на яку поверне користувача до головної сторінки програми.

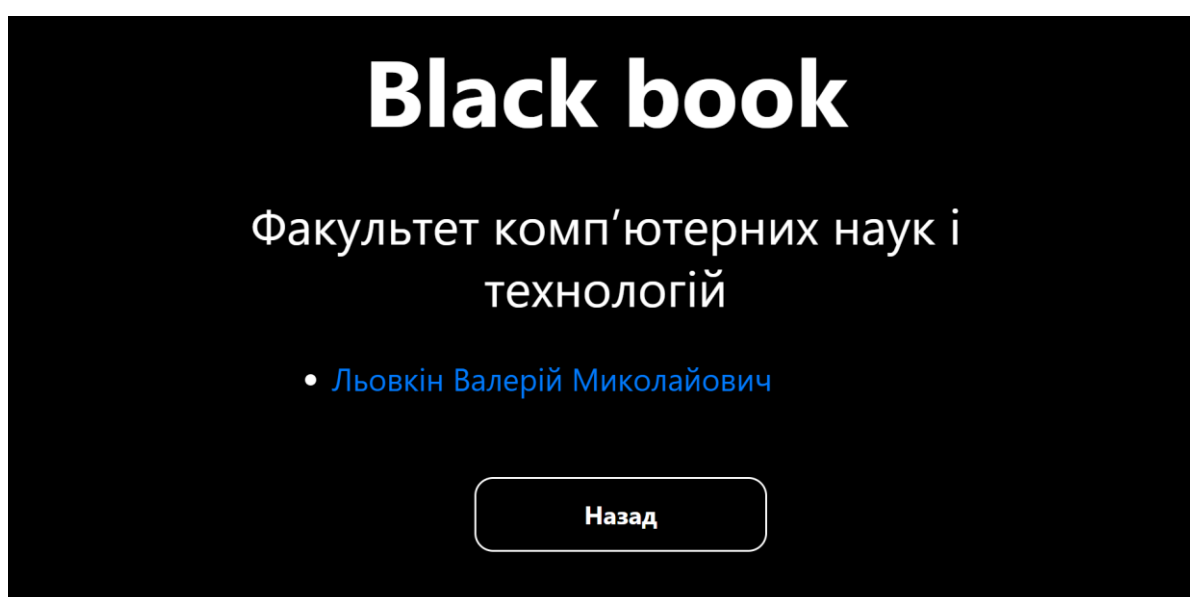


Рисунок 4.11 – Вигляд сторінки зі списком всіх викладачів, що відносяться до обраного факультету і знаходяться у чорній книзі

На цій сторінці користувач обирає викладача, коментарі до якого його цікавлять. Для того щоб обрати треба натиснути на ім'я викладача – користувач перейде на сторінку, зображену на рисунку 4.12. Унизу сторінки знаходиться кнопка «Назад», натискання на яку поверне користувача до головної сторінки програми.



Рисунок 4.12 – Вигляд сторінки зі списком усіх коментарів до обраного викладача

На цьому вікні користувач бачить список усіх коментарів до обраного викладача з датою, коли цей коментар був залишений на сайті. Для повернення до головної сторінки потрібно натиснути на кнопку «Назад», яка знаходиться унизу списку.

На головній сторінці також знаходиться прихована кнопка, яка зображена на рисунку 4.13.

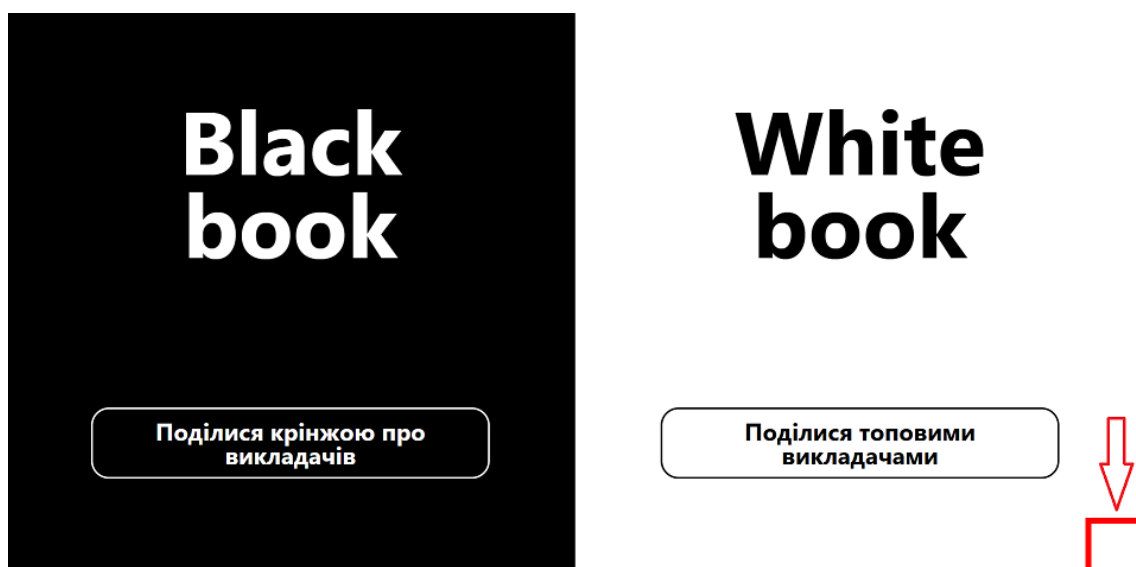


Рисунок 4.13 – Вигляд головного вікна програми з вказівкою на приховану кнопку

Натискання на приховану кнопку перенесе користувача на сторінку, зображену на рисунку 4.14.



Рисунок 4.14 – Вигляд сторінки авторизації до адмін панелі

На цій сторінці потрібно ввести правильні логін та пароль. Після цього необхідно натиснути на кнопку «Login» – адмін перейде на сторінку, зображену на рисунку 4.15.

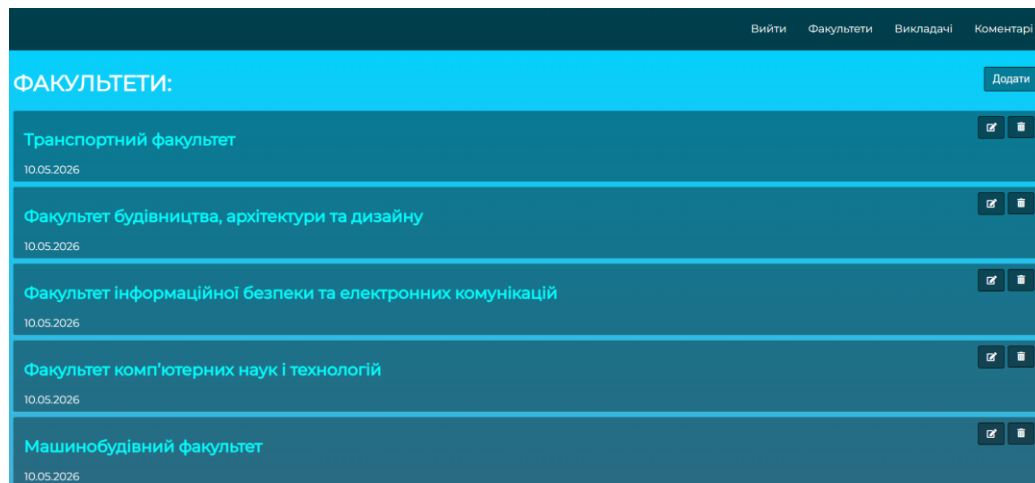


Рисунок 4.15 – Вигляд сторінки з факультетами

На цьому вікні адміні бачить список усіх факультетів. На горі знаходиться панель навігації, яка є єдиною для всіх сторінок адмін панелі. На панелі навігації знаходяться кнопки «Вийти» – адмін перейде на головну сторінку, зображену на рисунку 4.1, «Факультети» – адмін перейде на сторінку з факультетами, зображену на рисунку 4.15, «Викладачі» – адмін перейде на сторінку з усіма викладачами, зображену на рисунку 4.16, «Коментарі» – адмін перейде на сторінку з усіма коментарями, зображену на рисунку 4.17. Одразу під панеллю навігації на кожній з зазначених вище у цьому абзаці сторінок знаходиться кнопка «Додати» – адмін перейде до вікна додавання нового документу до однієї з колекцій бази даних, зображено на рисунку 4.18.

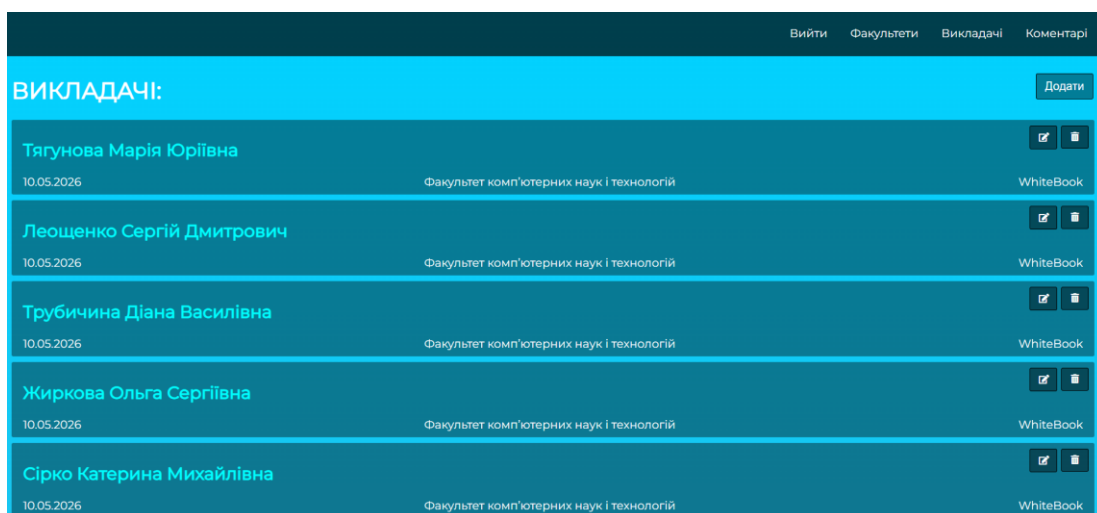


Рисунок 4.16 – Вигляд сторінки з викладачами

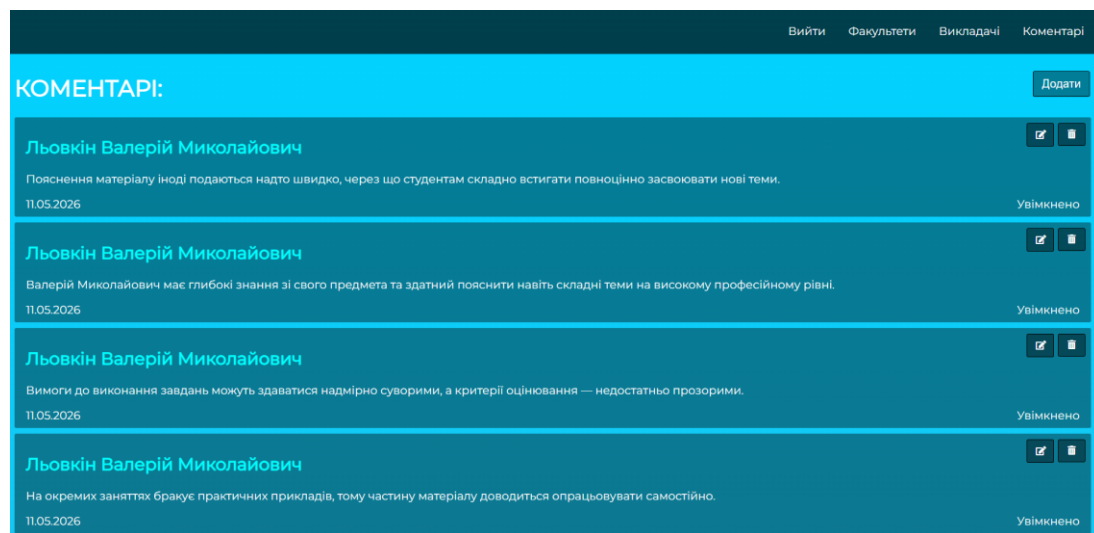


Рисунок 4.17 – Вигляд сторінки з коментарями

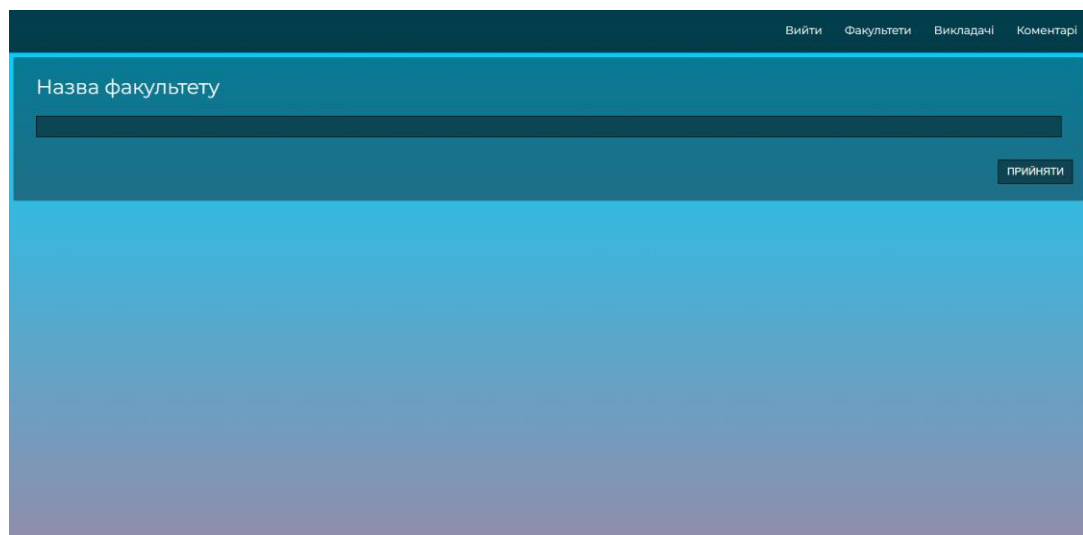


Рисунок 4.18 – Вигляд сторінки додавання

На цій сторінці після заповнення всіх необхідних полів, потрібно натиснути кнопку «Прийняти» – адмін повернеться до попередньої сторінки, де вже буде знаходитись документ, який він додав.

До кожного запису зі списку документів на сторінках «Факультети», «Викладачі», «Коментарі» справа додані кнопки редагування та видалення з відповідними іконками. Натиснувши кнопку редагування – адмін перейде до вікна редагування документу, зображено на рисунку 4.19.

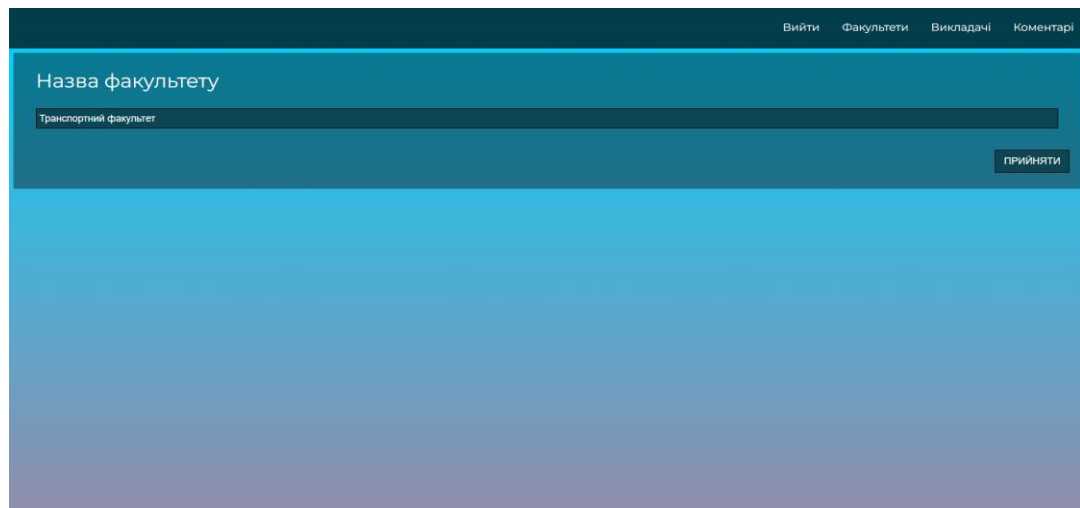


Рисунок 4.19 – Вигляд сторінки редагування

На цій сторінці після редагування всіх необхідних полів, потрібно натиснути кнопку «Прийняти» – адмін повернеться до попередньої сторінки, де вже буде знаходитись відредагований документ.

На вікні з факультетами, зображеному на рисунку 4.15, кожна назва факультету є посиланням на сторінку з викладачами, які відносяться до цього факультету, зображену на рисунку 4.20.

Вийти

Факультети

Викладачі

Коментарі

10.05.2026

ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК І ТЕХНОЛОГІЙ

Тягунова Марія Юріївна

10.05.2026

Факультет комп'ютерних наук і технологій

WhiteBook

Леощенко Сергій Дмитрович

10.05.2026

Факультет комп'ютерних наук і технологій

WhiteBook

Трубичина Діана Василівна

10.05.2026

Факультет комп'ютерних наук і технологій

WhiteBook

Жиркова Ольга Сергіївна

10.05.2026

Факультет комп'ютерних наук і технологій

WhiteBook

Сірко Катерина Михайлівна

10.05.2026

Факультет комп'ютерних наук і технологій

WhiteBook

Рисунок 4.20 – Вигляд сторінки з викладачами, які відносяться до обраного факультету

На цій сторінці кожне ім'я викладача є посиланням на сторінку з коментарями, які відносяться до обраного викладача, зображену на рисунку 4.21.

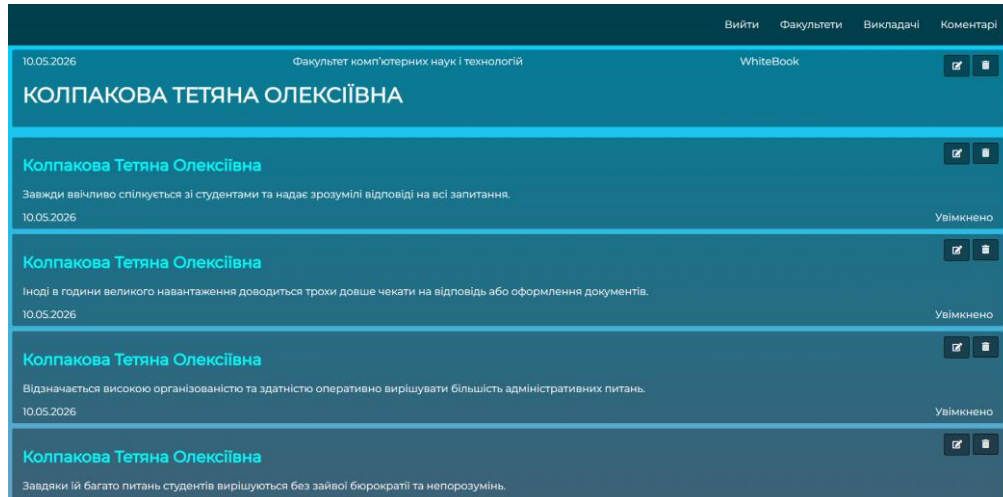


Рисунок 4.21 – Вигляд сторінки з коментарями, які відносяться до обраного викладача

4.2 Методика та результати тестування

Тестування програмного забезпечення проводилося відповідно до принципів функціонального, інтеграційного та регресійного тестування. Основною метою тестування було підтвердження коректності реалізації всіх функцій вебзастосунку та перевірка стійкості системи до некоректних дій користувачів.

Функціональне тестування передбачало перевірку кожного модуля окремо. Зокрема, були протестовані форми введення коментарів, механізми перегляду списків факультетів і викладачів, система авторизації адміністратора та модулі редагування даних.

Інтеграційне тестування використовувалося для перевірки взаємодії між вебінтерфейсом, серверною логікою та базою даних MongoDB. Перевірялася правильність передачі даних між компонентами системи та коректність відображення результатів у браузері.

Регресійне тестування проводилося після внесення змін до програмного коду для перевірки того, що нові зміни не порушили раніше реалізовану функціональність. Результати регресійного тестування наведено в таблиці 4.1.

Таблиця 4.1 – Результати функціонального тестування

Тестовий сценарій	Очікуваний результат	Фактичний результат
1	2	3
Запуск головної сторінки	Показ головної сторінки	Успішно
Відображення списку факультетів	Список факультетів	Успішно
Відображення списку викладачів	Список викладачів	Успішно
Відображення списку коментарів	Список коментарів	Успішно
Додавання позитивного коментаря	Збереження в базі даних (БД)	Успішно
Додавання негативного коментаря	Збереження в БД	Успішно
Введення порожнього коментаря	Повідомлення про помилку	Успішно
Відображення підтвердження	Повідомлення про успіх	Успішно
Вхід до адмін панелі з валідними даними	Доступ дозволено	Успішно

Продовження таблиці 4.1

1	2	3
Вхід до адмін панелі з неправильним паролем	Відмова у доступі	Успішно
Вхід до адмін панелі з неправильним логіном	Відмова у доступі	Успішно
Додавання факультету	Створення документа	Успішно
Редагування факультету	Оновлення документа	Успішно
Видалення факультету	Видалення з БД	Успішно
Додавання викладача	Створення документа	Успішно
Редагування викладача	Оновлення документа	Успішно
Видалення викладача	Видалення з БД	Успішно
Додавання коментаря	Створення документа	Успішно
Редагування коментаря	Оновлення документа	Успішно
Видалення коментаря	Видалення з БД	Успішно
Відображення викладачів, що відносяться до одного факультету	Список викладачів	Успішно
Відображення коментарів, що відносяться до одного викладача	Список коментарів	Успішно
Завантаження сторінки без даних	Коректне відображення	Успішно
Вихід з адмін панелі	Повернення на головну сторінку	Успішно

Усі тестові сценарії були виконані успішно. Помилки, що призводять до відмови системи, не виявлено. Валідація форм та механізми обробки винятків забезпечують коректну реакцію програми на дії користувачів.

4.3 Експериментальне дослідження програми

Метою експериментального дослідження було визначення продуктивності вебзастосунку при роботі з великим обсягом даних та оцінювання швидкодії основних операцій.

Для проведення експерименту до бази даних було внесено 12 факультетів, 50 викладачів та 200 коментарів. Випробування проводилися на персональному комп'ютері з процесором Intel Core i3-10105F, 16 ГБ оперативної пам'яті та операційною системою Windows 10.

Під час експерименту вимірювалися:

- час завантаження головної сторінки;
- час відкриття списку викладачів;
- час збереження нового коментаря;
- час відкриття адміністративної панелі;
- час фільтрації коментарів.

Результати навантажувального тестування наведені в табл. 4.2.

Таблиця 4.2 – Результати експериментального дослідження

Операція	Середній час	Оцінка
1	2	3
Завантаження головної сторінки	0,45 с	Відмінно
Відкриття білої книги	0,24 с	Відмінно
Відкриття чорної книги	0,24 с	Відмінно

Продовження таблиці 4.2

1	2	3
Відкриття списку факультетів	0,55 с	Відмінно
Відкриття списку викладачів	0,62 с	Відмінно
Відкриття списку коментарів	0,52 с	Відмінно
Збереження коментаря	0,28 с	Відмінно
Відкриття адмін-панелі	0,54 с	Відмінно
Додавання нового факультету	0,20 с	Відмінно
Редагування існуючого факультету	0,29 с	Відмінно
Видалення факультету	0,15 с	Відмінно
Додавання нового викладача	0,23 с	Відмінно
Редагування існуючого викладача	0,32 с	Відмінно
Видалення викладача	0,15 с	Відмінно
Додавання нового коментаря	0,25 с	Відмінно
Редагування існуючого коментаря	0,33 с	Відмінно
Видалення коментаря	0,15 с	Відмінно
Відкриття сторінки з викладачами, що відносяться до одного факультету	0,73 с	Відмінно
Відкриття сторінки з коментарями, що відносяться до одного викладача	0,59 с	Відмінно

Результати дослідження показали, що система забезпечує швидке виконання всіх основних операцій. Навіть при значному обсязі даних час обробки запитів не перевищував однієї секунди, що свідчить про високу ефективність використаних технологій.

4.4 Висновки за розділом 4

Проведене тестування та експериментальне дослідження підтвердили, що розроблений вебзастосунок повністю реалізує поставлені завдання та відповідає вимогам до сучасних інформаційних систем.

Система забезпечує зручний інтерфейс для студентів, надійні засоби адміністрування та ефективне зберігання інформації в базі даних MongoDB. Використання технологій Node.js, Express і MongoDB дозволило створити масштабоване рішення, яке може бути адаптоване до потреб інших закладів освіти.

Практична цінність розробки полягає в автоматизації збору та аналізу відгуків, що сприяє підвищенню якості освітнього процесу. Подальший розвиток програми може включати реалізацію статистичних звітів, інтеграцію з університетськими інформаційними системами та підтримку мобільних пристроїв.

ВИСНОВКИ

Під час написання дипломної роботи було:

- проаналізовано предметну область;
- описано основні рішення щодо реалізації компонентів системи;
- створено методiku роботи з системою користувача і програміста.

Програмний продукт було розроблено в середовищі програмування Visual Studio Code 2022. Основними мовами застосунку є HTML, CSS, JS.

Додаток був протестований та виявився працездатним, має зручний інтерфейс та нормальну швидкість роботи.

Програма має зручний у використанні інтерфейс, не потребує багато часу на опанування принципів роботи з нею.

Інтерфейс програми виконує наступні функції:

- доступ до колекцій бази даних;
- додавання інформації в колекції;
- редагування інформації в колекціях;
- видалення інформації з колекцій.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Grosky W., Ruas T. Software Engineering: A Practitioner's Approach. 9th Edition. Ohio : Columbus, 2020. 671 с.
2. Rate my professors. URL: <https://www.ratemyprofessors.com/> (date of access: 27.05.2026)
3. The student room URL: <https://www.thestudentroom.co.uk/> (date of access: 27.05.2026)
4. Stud search. URL: <https://studsearch.org/uk> (date of access: 27.05.2026)
5. Hunt A., Thomas D. Visual Studio Code Distilled: Evolved Code Editing for Windows, macOS, and Linux. Boston : Addison-Wesley Professional, 2021. 276 с.
6. Duckett J. HTML and CSS: Design and Build Websites. Indianapolis : John Wiley & Sons, 2011. 512 с.
7. Flanagan D. JavaScript: The Definitive Guide. 7th Edition. Sebastopol : O'Reilly Media, 2020. 706 с.
8. Brown E. Web Development with Node and Express: Leveraging the JavaScript Stack. 2nd Edition. Sebastopol : O'Reilly Media, 2019. 343 с.
9. Casciaro M., Mammino L. Node.js Design Patterns. 3rd Edition. Birmingham : Packt Publishing, 2020. 660 с.
10. Hahn E. Express in Action: Writing, Building, and Testing Node.js Applications. Shelter Island : Manning Publications, 2016. 256 с.
11. Bradshaw S., Brazil E., Chodorow K. MongoDB: The Definitive Guide. 3rd Edition. Sebastopol : O'Reilly Media, 2019. 425 с.
12. Karpov V. Mongoose for Application Development. Birmingham : Packt Publishing, 2020. 142 с.

ДОДАТОК А
Текст програми

A.1 Текст файла server.js

```

const express = require('express');
const morgan = require('morgan');
const mongoose = require('mongoose');
const methodOverride = require('method-override');
const departmentRoutes = require('./routes/department-routes');
const professorRoutes = require('./routes/professor-routes');
const commentRoutes = require('./routes/comment-routes');
const blackandwhiteRoutes = require('./routes/blackandwhite-routes');
const createPath = require('./helpers/create-path');

const app = express();

app.set('view engine', 'ejs');

const PORT = 3000;
const db = 'mongodb+srv://dorofeevpavlik:Pass321@cluster0.jjytx.mongodb.net/BlackAndWhite?retryWrites=true&w=majority&appName=Cluster0';

mongoose
  .connect(db, { useNewUrlParser: true, useUnifiedTopology: true })
  .then((res) => console.log('Connected to DB'))
  .catch((error) => console.log(error));

app.listen(PORT, (error) => {
  error ? console.log(error) : console.log(`listening port ${PORT}`);
});

app.use(express.urlencoded({ extended: false }));

app.use(morgan(':method :url :status :res[content-length] - :response-time ms'));

app.use(express.static('styles'));

app.use(methodOverride('_method'));

app.get('/', (req, res) => {
  const title = 'Home';
  res.render(createPath('index'), { title });
});

```

```

app.use(departmentRoutes);
app.use(professorRoutes);
app.use(commentRoutes);
app.use(blackandwhiteRoutes);

app.use((req, res) => {
  const title = 'Error Page';
  res
    .status(404)
    .render(createPath('error'), { title });
});

```

A.2 Текст файла department.js

```

const mongoose = require('mongoose');
const Schema = mongoose.Schema;

const departmentSchema = new Schema({
  department_name: {
    type: String,
    required: true,
  },
}, { timestamps: true });

const Department = mongoose.model('Department', departmentSchema);
module.exports = Department;

```

A.3 Текст файла professor.js

```

const mongoose = require('mongoose');
const Schema = mongoose.Schema;

const professorSchema = new Schema({
  professor_name: {
    type: String,
    required: true,
  },
  department_name: {
    type: String,
    required: true,
  },
  book_name: {
    type: String,
    required: true,
  },
});

```

```

    }
  }, { timestamps: true });

```

```

const Professor = mongoose.model('Professor', professorSchema);
module.exports = Professor;

```

A.4 Текст файла **comment.js**

```

const mongoose = require('mongoose');
const Schema = mongoose.Schema;

```

```

const commentSchema = new Schema({
  professor_name: {
    type: String,
    required: true,
  },
  text: {
    type: String,
    required: true,
  },
  visible: {
    type: Boolean,
    required: true,
  }
}, { timestamps: true });

```

```

const Comment = mongoose.model('Comment', commentSchema);
module.exports = Comment;

```

A.5 Текст файла **department-controller.js**

```

const Department = require('../models/department');
const Professor = require('../models/professor');
const createPath = require('../helpers/create-path');

```

```

const handleError = (res, error) => {
  console.log(error);
  res.render(createPath('error'), { title: 'Error' });
}

```

```

const getDepartment = async(req, res) => {
  const title = 'Professor';
  Promise.all([
    Department

```

```

        .findById(req.params.id),
        Professor
        .find()
        .sort({ createdAt: -1 }),
    ])
    .then(dp => { res.render(createPath('department'), { dp, title }) })
    .catch((error) => handleError(res, error));
}

const deleteDepartment = (req, res) => {
    Department
    .findByIdAndDelete(req.params.id)
    .then((result) => {
        res.sendStatus(200);
    })
    .catch((error) => handleError(res, error));
}

const getEditDepartment = (req, res) => {
    const title = 'Edit department';
    Department
    .findById(req.params.id)
    .then(department => res.render(createPath('edit-department'), {
department, title })))
    .catch((error) => handleError(res, error));
}

const editDepartment = (req, res) => {
    const { department_name } = req.body;
    const { id } = req.params;
    Department
    .findByIdAndUpdate(id, { department_name })
    .then((result) => res.redirect('/departments'))
    .catch((error) => handleError(res, error));
}

const getDepartments = (req, res) => {
    const title = 'Departments';
    Department
    .find()
    .sort({ createdAt: -1 })
    .then(departments => res.render(createPath('departments'), { departments,
title })))
    .catch((error) => handleError(res, error));
}

```

```

const getAddDepartment = (req, res) => {
  const title = 'Add Department';
  res.render(createPath('add-department'), { title });
}

const addDepartment = (req, res) => {
  const { department_name } = req.body;
  const department = new Department({ department_name });
  department
    .save()
    .then((result) => res.redirect('/departments'))
    .catch((error) => handleError(res, error));
}

module.exports = {
  getDepartment,
  deleteDepartment,
  getEditDepartment,
  editDepartment,
  getDepartments,
  getAddDepartment,
  addDepartment,
};

```

A.6 Текст файла professor-controller.js

```

const Professor = require('../models/professor');
const Department = require('../models/department');
const Comment = require('../models/comment');
const createPath = require('../helpers/create-path');

const handleError = (res, error) => {
  console.log(error);
  res.render(createPath('error'), { title: 'Error' });
}

const getProfessor = async(req, res) => {
  const title = 'Professor';
  Promise.all([
    Professor
      .findById(req.params.id),
    Comment
      .find()

```

```

        .sort({ createdAt: -1 }),
      ])
      .then(pc => { res.render(createPath('professor'), { pc, title }) })
      .catch((error) => handleError(res, error));
    }

    const deleteProfessor = (req, res) => {
      Professor
        .findByIdAndDelete(req.params.id)
        .then((result) => {
          res.sendStatus(200);
        })
        .catch((error) => handleError(res, error));
    }

    const getEditProfessor = async(req, res) => {
      const title = 'Edit professor';
      Promise.all([
        Professor
          .findById(req.params.id),
        Department
          .find()
          .sort({ createdAt: -1 }),
      ])
      .then(pd => { res.render(createPath('edit-professor'), { pd, title }) })
      .catch((error) => handleError(res, error));
    }

    const editProfessor = (req, res) => {
      const { professor_name, department_name, book_name } = req.body;
      const { id } = req.params;
      Professor
        .findByIdAndUpdate(id, { professor_name, department_name,
book_name })
        .then((result) => res.redirect('/professors'))
        .catch((error) => handleError(res, error));
    }

    const getProfessors = (req, res) => {
      const title = 'Professors';
      Professor
        .find()
        .sort({ createdAt: -1 })
        .then(professors => res.render(createPath('professors'), { professors, title
    )))

```

```

    .catch((error) => handleError(res, error));
  }

const getAddProfessor = (req, res) => {
  const title = 'Add Professor';

  Department
    .find()
    .sort({ createdAt: -1 })
    .then(departments => res.render(createPath('add-professor'), {
departments, title })))
    .catch((error) => handleError(res, error));
}

const addProfessor = (req, res) => {
  const { professor_name, department_name, book_name } = req.body;
  const professor = new Professor({ professor_name, department_name,
book_name });
  professor
    .save()
    .then((result) => res.redirect('/professors'))
    .catch((error) => handleError(res, error));
}

module.exports = {
  getProfessor,
  deleteProfessor,
  getEditProfessor,
  editProfessor,
  getProfessors,
  getAddProfessor,
  addProfessor,
};

```

A.7 Текст файлу comment-controller.js

```

const Comment = require('../models/comment');
const Professor = require('../models/professor');
const createPath = require('../helpers/create-path');

const handleError = (res, error) => {
  console.log(error);
  res.render(createPath('error'), { title: 'Error' });
};

```

```

    }

    const getComment = (req, res) => {
      const title = 'Comment';
      Comment
        .findById(req.params.id)
        .then(comment => res.render(createPath('comment'), { comment, title }))
        .catch((error) => handleError(res, error));
    }

    const deleteComment = (req, res) => {
      Comment
        .findByIdAndDelete(req.params.id)
        .then((result) => {
          res.sendStatus(200);
        })
        .catch((error) => handleError(res, error));
    }

    const getEditComment = (req, res) => {
      const title = 'Edit comment';
      Promise.all([
        Comment
          .findById(req.params.id),
        Professor
          .find()
          .sort({ createdAt: -1 }),
      ])
        .then(cp => { res.render(createPath('edit-comment'), { cp, title }) })
        .catch((error) => handleError(res, error));
    }

    const editComment = (req, res) => {
      const { professor_name, text, visible } = req.body;
      const { id } = req.params;
      Comment
        .findByIdAndUpdate(id, { professor_name, text, visible })
        .then((result) => res.redirect('/comments'))
        .catch((error) => handleError(res, error));
    }

    const getComments = (req, res) => {
      const title = 'Comments';
      Comment
        .find()

```

```

        .sort({ createdAt: -1 })
        .then(comments => res.render(createPath('comments'), { comments, title
    )))
        .catch((error) => handleError(res, error));
    }

    const getAddComment = (req, res) => {
        const title = 'Add Comment';
        Professor
            .find()
            .sort({ createdAt: -1 })
            .then(professors => res.render(createPath('add-comment'), { professors,
title })))
            .catch((error) => handleError(res, error));
    }

    const addComment = (req, res) => {
        const { professor_name, text, visible } = req.body;
        const comment = new Comment({ professor_name, text, visible });
        comment
            .save()
            .then((result) => res.redirect('/comments'))
            .catch((error) => handleError(res, error));
    }

    module.exports = {
        getComment,
        deleteComment,
        getEditComment,
        editComment,
        getComments,
        getAddComment,
        addComment,
    };

```

A.8 Текст файла blackandwhite-controller.js

```

const Professor = require('../models/professor');
const Department = require('../models/department');
const Comment = require('../models/comment');
const createPath = require('../helpers/create-path');

const handleError = (res, error) => {

```

```

    console.log(error);
    res.render(createPath('error'), { title: 'Error' });
  }

  const bb = (req, res) => {
    const title = 'Black Book';
    res.render(createPath('bb'), { title });
  }

  const bbdepartment = (req, res) => {
    const title = 'Black Book';
    Promise.all([
      Department
        .findById(req.params.id),
      Professor
        .find()
        .sort({ createdAt: -1 }),
    ])
    .then(dp => { res.render(createPath('bbdepartment'), { dp, title }) })
    .catch((error) => handleError(res, error));
  }

  const bbform = (req, res) => {
    const title = 'Black Book';
    Professor
      .find()
      .sort({ createdAt: -1 })
      .then(professors => res.render(createPath('bbform'), { professors, title }))
      .catch((error) => handleError(res, error));
  }

  const bbformPost = (req, res) => {
    const { professor_name, text, visible } = req.body;
    const comment = new Comment({ professor_name, text, visible });
    comment
      .save()
      .then((result) => res.redirect('/'))
      .catch((error) => handleError(res, error));
  }

  const bblist = (req, res) => {
    const title = 'Black Book';
    Department
      .find()
      .sort({ createdAt: -1 })

```

```

        .then(departments => res.render(createPath('bblist'), { departments, title
    )))
    .catch((error) => handleError(res, error));
}

const bbprofessor = (req, res) => {
  const title = 'Black Book';
  Promise.all([
    Professor
      .findById(req.params.id),
    Comment
      .find()
      .sort({ createdAt: -1 }),
  ])
  .then(pc => { res.render(createPath('bbprofessor'), { pc, title }) })
  .catch((error) => handleError(res, error));
}

const wb = (req, res) => {
  const title = 'White Book';
  res.render(createPath('wb'), { title });
}

const wbdepartment = (req, res) => {
  const title = 'White Book';
  Promise.all([
    Department
      .findById(req.params.id),
    Professor
      .find()
      .sort({ createdAt: -1 }),
  ])
  .then(dp => { res.render(createPath('wbdepartment'), { dp, title }) })
  .catch((error) => handleError(res, error));
}

const wbform = (req, res) => {
  const title = 'White Book';
  Professor
    .find()
    .sort({ createdAt: -1 })
    .then(professors => res.render(createPath('wbform'), { professors, title }))
    .catch((error) => handleError(res, error));
}

```

```

const wbformPost = (req, res) => {
  const { professor_name, text, visible } = req.body;
  const comment = new Comment({ professor_name, text, visible });
  comment
    .save()
    .then((result) => res.redirect('/'))
    .catch((error) => handleError(res, error));
}

const wblist = (req, res) => {
  const title = 'White Book';
  Department
    .find()
    .sort({ createdAt: -1 })
    .then(departments => res.render(createPath('wblist'), { departments, title
    )))
    .catch((error) => handleError(res, error));
}

const wbprofessor = (req, res) => {
  const title = 'White Book';
  Promise.all([
    Professor
      .findById(req.params.id),
    Comment
      .find()
      .sort({ createdAt: -1 }),
  ])
    .then(pc => { res.render(createPath('wbprofessor'), { pc, title }) })
    .catch((error) => handleError(res, error));
}

const admin = (req, res) => {
  const title = 'Admin';
  res.render(createPath('admin'), { title });
}

const login = (req, res) => {
  const title = 'Admin';
  res.render(createPath('departments'), { title });
}

module.exports = {
  bb,
  bbdepartment,

```

```

    bbform,
    bbformPost,
    bblist,
    bbprofessor,
    wb,
    wbdepartment,
    wbform,
    wbformPost,
    wblist,
    wbprofessor,
    admin,
    login
  };

```

A.9 Текст файла department-routes.js

```

const express = require('express');
const {
  getDepartment,
  deleteDepartment,
  getEditDepartment,
  editDepartment,
  getDepartments,
  getAddDepartment,
  addDepartment
} = require('../controllers/department-controller');

const router = express.Router();

router.get('/departments/:id', getDepartment);
router.delete('/departments/:id', deleteDepartment);
router.get('/edit-department/:id', getEditDepartment);
router.put('/edit-department/:id', editDepartment);
router.get('/departments', getDepartments);
router.get('/add-department', getAddDepartment);
router.post('/add-department', addDepartment);

module.exports = router;

```

A.10 Текст файла professor-routes.js

```

const express = require('express');
const {
  getProfessor,

```

```

deleteProfessor,
getEditProfessor,
editProfessor,
getProfessors,
getAddProfessor,
addProfessor
} = require('../controllers/professor-controller');

const router = express.Router();

router.get('/professors/:id', getProfessor);
router.delete('/professors/:id', deleteProfessor);
router.get('/edit-professor/:id', getEditProfessor);
router.put('/edit-professor/:id', editProfessor);
router.get('/professors', getProfessors);
router.get('/add-professor', getAddProfessor);
router.post('/add-professor', addProfessor);

module.exports = router;

```

A.11 Текст файла comment-routes.js

```

const express = require('express');
const {
  getComment,
  deleteComment,
  getEditComment,
  editComment,
  getComments,
  getAddComment,
  addComment
} = require('../controllers/comment-controller');

const router = express.Router();
router.get('/comments/:id', getComment);
router.delete('/comments/:id', deleteComment);
router.get('/edit-comment/:id', getEditComment);
router.put('/edit-comment/:id', editComment);
router.get('/comments', getComments);
router.get('/add-comment', getAddComment);
router.post('/add-comment', addComment);

module.exports = router;

```

A.12 Текст файла blackandwhite-routes.js

```

const express = require('express');
const {
  bb,
  bbdepartment,
  bbform,
  bbformPost,
  bblist,
  bbprofessor,
  wb,
  wbdepartment,
  wbform,
  wbformPost,
  wblast,
  wbprofessor,
  admin,
  login
} = require('../controllers/BlackAndWhite-controller');

const router = express.Router();

router.get('/bb', bb);
router.get('/bbdepartment/:id', bbdepartment);
router.get('/bbform', bbform);
router.post('/bbform', bbformPost);
router.get('/bblist', bblist);
router.get('/bbprofessor/:id', bbprofessor);
router.get('/wb', wb);
router.get('/wbdepartment/:id', wbdepartment);
router.get('/wbform', wbform);
router.post('/wbform', wbformPost);
router.get('/wblast', wblast);
router.get('/wbprofessor/:id', wbprofessor);
router.get('/admin', admin);
router.post('/departments', login);

module.exports = router;

```

A.13 Текст файла create-path.js

```

const path = require('path');
const createPath = (page) => path.resolve(__dirname, '..', 'views',
`${page}.ejs`);

```

```
module.exports = createPath;
```

A.14 Текст файлу head.ejs

```
<head>
  <meta charset="UTF-8">
  <title>BlackAndWhiteBook | <%= title %></title>
  <link rel="preconnect" href="https://fonts.gstatic.com">
  <link
href="https://fonts.googleapis.com/css2?family=Montserrat&display=swap"
rel="stylesheet">
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.3/css/all.min.css">
  <link rel="stylesheet" href="/main.css">
</head>
```

A.15 Текст файлу nav.ejs

```
<nav>
  <ul>
    <li><a href="/">Вийти</a></li>
    <li><a href="/departments">Факультети</a></li>
    <li><a href="/professors">Викладачі</a></li>
    <li><a href="/comments">Коментарі</a></li>
  </ul>
</nav>
```

A.16 Текст файлу bwhead.ejs

```
<head>
  <meta charset="UTF-8">
  <title>BlackAndWhiteBook</title>
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAi
S6JXm" crossorigin="anonymous">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" type="text/css" href="/style.css">
  <link rel="stylesheet" type="text/css" href="/adaptive.css">
</head>
```

A.17 Текст файлу index.ejs

```

<!DOCTYPE html>
<html lang="en">
  <%- include('./partials/bwhead.ejs') %>
<body class="home">

  <div id="main" class="content">

    <div id="black_book">
      <p id="blackbook_head">Black book</p>
      <a href="/bb" class="nav-link">
        <button id="blackbook_button">
          Поділися крінжою про викладачів
        </button>
      </a>
    </div>
    <div id="white_book">
      <p id="whitebook_head">White book</p>
      <a href="/wb" class="nav-link">
        <button id="whitebook_button">
          Поділися топовими викладачами
        </button>
      </a>
    </div>
    <a href="/admin" class="nav-link" id="href_admin"></a>

  </div>
</body>
</html>

```

A.18 Текст файлу error.ejs

```

<!DOCTYPE html>
<html lang="en">
  <%- include('./partials/head.ejs') %>
<body class="error">
  <h1 class="error-code">Помилка</h1>
  <p>Щось пішло не так...</p>
</body>
</html>

```

A.19 Текст файлу wb.ejs

```

<!DOCTYPE html>
<html lang="en">
  <%- include('./partials/bwhead.ejs') %>
  <body>
    <div id="main">

      <div id="wb_main">
        <p id="wb_head">White book</p>
        <a href="/wbform" class="nav-link">
          <button id="wb_button_1" class="subscribe-for-push">
            Заповнити форму
          </button>
        </a>
        <a href="/wblist" class="nav-link">
          <button id="wb_button_2" class="subscribe-for-push">
            Переглянути свій факультет
          </button>
        </a>
        <p id="wb_text">Якщо твій викладач створює неймовірну
атмосферу
        під час навчання - він/вона мусить бути в White book!</p>
      </div>

    </div>

  </body>
</html>

```

A.20 Текст файлу wbform.ejs

```

<!DOCTYPE html>
<html lang="en">
  <%- include('./partials/bwhead.ejs') %>
  <body>
    <div id="main">
      <form id="comment-form" method="post" action="/wbform">

        <div id="wb_main">
          <p id="wbform_text_department">Викладач:</p>
          <select name="professor_name" id="wbform_department"
required="required">
            <% if (professors.length) { %>

```

```

        <% professors.forEach(({ id, professor_name }) => { %>
            <option><%= professor_name %></option>
        <% }) %>
    <% } %>
</select>
    <p id="wbform_text_professor">Коментар:</p>
        <textarea name="text" id="wb_comment" minlength="5"
maxlength="500" required="required" autocomplete="off"></textarea>
        <input name="visible" id="visible" required="required"
value="false" >
        <div class="form-button">
            <input type="submit" id="wbform_button" value="Залишити
коментар" onclick="Add()">
        </div>
    </div>

</form>
</div>
<script>
    function Add() {
        let text = document.getElementById("wb_comment").value;
        if(text.length >= 5 && text.length <= 500){
            alert("Ваш коментар буде перевірено і додано до сайту
найближчим часом");
        }
    }
</script>
</body>
</html>

```

A.21 Текст файлу wblast.ejs

```

<!DOCTYPE html>
<html lang="en">
    <%- include('./partials/bwhead.ejs') %>
    <body style="background: #fff;">
        <div id="main">

            <div id="wb_main">
                <p id="wb_head">White book</p>

                <ul id="wb_departments">
                    <% if (departments.length) { %>

```

```

=> { %>
    <% departments.forEach(({ id, department_name, createdAt })
    <article>
    <li>
    <a href="/wbdepartment/<%= id %>"><%= department_name
%></a>
    </li>
    </article>
    <% }) %>
    <% } %>

    <a href="/" class="nav-link">
    <button id="wb_button_3" class="subscribe-for-push">
    Назад
    </button>
    </a>
    <section></section>
    </ul>
    </div>
    </div>

    </body>
    </html>

```

A.22 Текст файла wbdepartment.ejs

```

<!DOCTYPE html>
<html lang="en">
  <%- include('./partials/bwhead.ejs') %>
  <body style="background: #fff;">
    <div id="main">

      <div id="wb_main">
        <p id="wb_head">White book</p>
        <p id="wb_list_head"><%= dp[0].department_name %></p>
        <ul id="wb_proffesors">
          <% const professors = []; %>
          <% if (dp[1].length) { %>
            <% dp[1].forEach(function (professor) { %>
              <% if (professor.department_name === dp[0].department_name)
{ %>
                <% professors.push(professor); %>
              <% } %>
            <% }) %>

```

```

        <% professors.forEach(({ id, professor_name, createdAt,
department_name, book_name }) => { %>
        <% if (book_name == "WhiteBook") { %>
        <li>
        <a id="head_link" href="/wbprofessor/<%= id %>"><%=
professor_name %></a>
        </li>
        <% } %>
        <% }) %>
        <% } %>

        <a href="/" class="nav-link">
        <button id="wb_button_3" class="subscribe-for-push">
        Назад
        </button>
        </a>
        <section></section>
    </ul>
</div>
</div>

</body>
</html>

```

A.23 Текст файла wbprofessor.ejs

```

<!DOCTYPE html>
<html lang="en">
    <%- include('./partials/bwhead.ejs') %>
    <body style="background: #fff;">
        <div id="main">

            <div id="wb_main">
                <p id="wb_head">White book</p>
                <p id="wb_list_head"><%= pc[0].professor_name %></p>
                <ul id="wb_comments">
                    <% const comments = []; %>
                    <% if (pc[1].length) { %>
                        <% pc[1].forEach(function (comment) { %>
                            <% if (comment.professor_name === pc[0].professor_name) {
%>
                                <% comments.push(comment); %>
                                <% } %>
                            <% }) %>

```

```

    <% comments.forEach(({ id, professor_name, createdAt, text,
visible }) => { %>
    <% if (visible == true) { %>
    <li>
    <div id="wb_comments_text">
    <p><%= text %></p>
    </div>
    </li>
    <div class="info">
    <span><%= createdAt.toLocaleDateString() %></span>
    </div>
    <% } %>
    <% }) %>
    <% } %>

    <a href="/" class="nav-link">
    <button id="wb_button_4" class="subscribe-for-push">
    Назад
    </button>
    </a>
    <section></section>
  </ul>
</div>
</div>

</body>
</html>

```

A.24 Текст файлу bb.ejs

```

<!DOCTYPE html>
<html lang="en">
  <%- include('./partials/bwhead.ejs') %>
<body>
<div id="main">

  <div id="bb_main">
    <p id="bb_head">Black book</p>
    <a href="/bbform" class="nav-link">
      <button id="bb_button_1" class="subscribe-for-push">
        Заповнити форму
      </button>
    </a>
    <a href="/bblist" class="nav-link">

```

```

        <button id="bb_button_2" class="subscribe-for-push">
            Переглянути свій факультет
        </button>
    </a>
    <p id="bb_text">Якщо твій викладач творить крінжу та порушує
        права студентів - він/вона мусить бути в Black book!</p>
</div>

</div>

</body>
</html>

```

A.25 Текст файлу bbform.ejs

```

<!DOCTYPE html>
<html lang="en">
    <%- include('./partials/bwhead.ejs') %>
    <body>
        <div id="main">
            <form id="comment-form" method="post" action="/bbform">

                <div id="bb_main">
                    <p id="bbform_text_department">Викладач:</p>
                    <select name="professor_name" id="bbform_department"
required="required">
                        <% if (professors.length) { %>
                        <% professors.forEach(( { id, professor_name }) => { %>
                            <option><%= professor_name %></option>
                        <% } %>
                        <% } %>
                    </select>
                    <p id="bbform_text_professor">Коментар:</p>
                    <textarea name="text" id="bb_comment" minlength="5"
maxlength="500" required="required" autocomplete="off"></textarea>
                    <input name="visible" id="visible" required="required"
value="false" >
                    <div class="form-button">
                        <input type="submit" id="bbform_button" value="Залишити
коментар" onclick="Add()">
                    </div>
                </div>

            </form>

```

```

</div>
<script>
function Add() {
  let text = document.getElementById("bb_comment").value;
  if(text.length >= 5 && text.length <= 500){
    alert("Ваш коментар буде перевірено і додано до сайту
найближчим часом");
  }
}
</script>
</body>
</html>

```

A.26 Текст файлу bblist.ejs

```

<!DOCTYPE html>
<html lang="en">
  <%- include('./partials/bwhead.ejs') %>
  <body style="background: #000000;">
    <div id="main">

      <div id="bb_main">
        <p id="bb_head">Black book</p>

        <ul id="bb_departments">
          <% if (departments.length) { %>
            <% departments.forEach(({ id, department_name, createdAt })
=> { %>

              <article>
                <li>
                  <a href="/bbdepartment/<%= id %>"><%= department_name
%></a>

                  </li>
                </article>
              <% }) %>
            <% } %>

            <a href="/" class="nav-link">
              <button id="bb_button_3" class="subscribe-for-push">
                Назад
              </button>
            </a>
          <section></section>
        </ul>

```

```

    </div>
  </div>

</body>
</html>

```

A.27 Текст файла `bbdepartment.ejs`

```

<!DOCTYPE html>
<html lang="en">
  <%- include('./partials/bwhead.ejs') %>
  <body style="background: #000000;">
    <div id="main">

      <div id="bb_main">
        <p id="bb_head">Black book</p>
        <p id="bb_list_head"><%= dp[0].department_name %></p>
        <ul id="bb_proffesors">
          <% const professors = []; %>
          <% if (dp[1].length) { %>
            <% dp[1].forEach(function (professor) { %>
              <% if (professor.department_name === dp[0].department_name)
{ %>

                <% professors.push(professor); %>
                <% } %>
              <% }) %>
              <% professors.forEach((( { id, professor_name, createdAt,
department_name, book_name }) => { %>
                <% if (book_name == "BlackBook") { %>
                  <li>
                    <a id="head_link" href="/bbprofessor/<%= id %>"><%=
professor_name %></a>
                  </li>
                <% } %>
              <% }) %>
            <% } %>

            <a href="/" class="nav-link">
              <button id="bb_button_3" class="subscribe-for-push">
                Назад
              </button>
            </a>
          <section></section>
        </ul>

```

```

    </div>
  </div>

```

```

</body>
</html>

```

A.28 Текст файла bbprofessor.ejs

```

<!DOCTYPE html>
<html lang="en">
  <%- include('./partials/bwhead.ejs') %>
  <body style="background: #000000;">
    <div id="main">

      <div id="bb_main">
        <p id="bb_head">Black book</p>
        <p id="bb_list_head"><%= pc[0].professor_name %></p>
        <ul id="bb_comments">
          <% const comments = []; %>
          <% if (pc[1].length) { %>
            <% pc[1].forEach(function (comment) { %>
              <% if (comment.professor_name === pc[0].professor_name) {
%>
                <% comments.push(comment); %>
              <% } %>
            <% }) %>
            <% comments.forEach(({ id, professor_name, createdAt, text,
visible }) => { %>
              <% if (visible == true) { %>
                <li>
                  <div id="bb_comments_text">
                    <p><%= text %></p>
                  </div>
                </li>
                <div class="info">
                  <span><%= createdAt.toLocaleDateString() %></span>
                </div>
              <% } %>
            <% }) %>
          <% } %>

```

```

    <a href="/" class="nav-link">
    <button id="bb_button_4" class="subscribe-for-push">
      Назад
    </button>
  </a>
</section></section>
</ul>
</div>
</div>

</body>
</html>

```

A.29 Текст файла admin.ejs

```

<!DOCTYPE html>
<html lang="en">
  <%- include('./partials/bwhead.ejs') %>
  <body>

    <div id="main" class="content">

      <div id="adminback">
        <div class="container pb-5" id="adminlogin">
          <div class="row">
            <div class="col-12 navigation"></div>
            <div class="col-12">
              <h1 class="mb-5">Login</h1>

              <form>
                <div class="mb-3">
                  <label class="form-label">Login</label>
                  <input type="text" autocomplete="off" class="form-
control" id="login" name="login">

                </div>
                <div class="mb-3">
                  <label for="password" class="form-
label">Password</label>
                  <input type="password" class="form-control"
id="password" name="password">
                </div>
                <div class="text-end">

```

```

        onclick="Login()">Login</button>
      </div>
    </form>

  </div>
</div>
</div>
</div>
</div>

</div>
<script>
  function Login() {
    let login = document.getElementById("login").value;
    let password = document.getElementById("password").value;
    if (login == "zp" && password == "admin") {
      fetch(`/departments`, {
        method: 'POST',
      }).then(() => {
        window.location.href = '/departments';
      });
    } else {
      alert("ERROR! invalid login or password");
    }
  }
</script>

</body>
</html>

```

A.30 Текст файлу departments.ejs

```

<!DOCTYPE html>
<html lang="en">
  <%- include('./partials/head.ejs') %>
<body>
  <%- include('./partials/nav.ejs') %>
  <h1>Факультети:</h1>
  <button class="btn-add" >
    <a href="/add-department" class="btn-add-text">Додати</a>
  </button>
  <ul>
    <% if (departments.length) { %>
      <% departments.forEach(({ id, department_name, createdAt }) => { %>

```

```

<article>
  <h2>
    <a href="/departments/<%= id %>"><%= department_name %></a>
  </h2>
  <div class="info">
    <span><%= createdAt.toLocaleDateString() %></span>
  </div>
  <a href="/edit-department/<%= id %>" class="btn-edit">
    <i class="fas fa-edit"></i>
  </a>
  <button class="btn-delete" data-id="<%= id %>">
    <i class="fas fa-trash-alt" data-id="<%= id %>"></i>
  </button>
</article>
<% }) %>
<% } %>
</ul>
<script>
document.addEventListener('click', (e) => {
  const id = e.target?.dataset?.id || null;
  if (id) {
    fetch(`/departments/${id}`, {
      method: 'DELETE',
    }).then(() => {
      window.location.reload();
    });
  }
});
</script>
</body>
</html>

```

A.31 Текст файла add-department.ejs

```

<!DOCTYPE html>
<html lang="en">
  <%- include('./partials/head.ejs') %>
<body>
  <%- include('./partials/nav.ejs') %>
  <form id="department-form" method="post" action="/add-department">
    <div class="form-info">
      <label>Назва факультету
      <input type="text" name="department_name">
    </div>
  </form>

```

```

    </label>
  </div>
  <div class="form-button">
    <input type="submit" value="Прийняти">
  </div>
</form>
</body>
</html>

```

A.32 Текст файлу edit-department.ejs

```

<!DOCTYPE html>
<html lang="en">
  <%- include('./partials/head.ejs') %>
<body>
  <%- include('./partials/nav.ejs') %>
  <form id="department-form" method="post" action="/edit-
department/<%= department.id %>?_method=PUT">
    <div class="form-info">
      <label>Назва факультету
      <input type="text" name="department_name" value="<%=
department.department_name %>">
    </label>
  </div>
  <div class="form-button">
    <input type="submit" value="Прийняти">
  </div>
</form>
</body>
</html>

```

A.33 Текст файлу professors.ejs

```

<!DOCTYPE html>
<html lang="en">
  <%- include('./partials/head.ejs') %>
<body>
  <%- include('./partials/nav.ejs') %>
  <h1>Викладачі:</h1>
  <button class="btn-add" >
    <a href="/add-professor" class="btn-add-text">Додати</a>
  </button>

```

```

<ul>
  <% if (professors.length) { %>
    <% professors.forEach(({ id, professor_name, createdAt,
department_name, book_name }) => { %>
      <article>
        <h2>
          <a href="/professors/<%= id %>"><%= professor_name %></a>
        </h2>
        <div class="info">
          <span><%= createdAt.toLocaleDateString() %></span>
          <span><%= department_name %></span>
          <span><%= book_name %></span>
        </div>
        <a href="/edit-professor/<%= id %>" class="btn-edit">
          <i class="fas fa-edit"></i>
        </a>
        <button class="btn-delete" data-id="<%= id %>">
          <i class="fas fa-trash-alt" data-id="<%= id %>"></i>
        </button>
      </article>
    <% }) %>
  <% } %>
</ul>
<script>
document.addEventListener('click', (e) => {
  const id = e.target?.dataset?.id || null;
  if (id) {
    fetch(`/professors/${id}`, {
      method: 'DELETE',
    }).then(() => {
      window.location.reload();
    });
  }
});
</script>
</body>
</html>

```

A.34 Текст файла add-professor.ejs

```

<!DOCTYPE html>
<html lang="en">
  <%- include('./partials/head.ejs') %>

```

```

<body>
  <%- include('./partials/nav.ejs') %>
  <form id="professor-form" method="post" action="/add-professor">
    <div class="form-info">
      <label>ПІБ викладача
        <input type="text" name="professor_name">
      </label>

      <label>Назва факультету
        <select name="department_name" required="required">
          <% if (departments.length) { %>
            <% departments.forEach(({ id, department_name }) => { %>
              <option><%= department_name %></option>
            <% }) %>
          <% } %>
        </select>
      </label>

      <label>Назва книги
        <select name="book_name" required="required">
          <option>WhiteBook</option>
          <option>BlackBook</option>
        </select>
      </label>

    </div>
    <div class="form-button">
      <input type="submit" value="Прийняти">
    </div>
  </form>

</body>
</html>

```

A.35 Текст файлу edit-professor.ejs

```

<!DOCTYPE html>
<html lang="en">
  <%- include('./partials/head.ejs') %>
  <body>
    <%- include('./partials/nav.ejs') %>

```

```

    <form id="professor-form" method="post" action="/edit-professor/<%=
pd[0].id %>?_method=PUT">
      <div class="form-info">
        <label>ПІБ викладача
          <input type="text" name="professor_name" value="<%=
pd[0].professor_name %>">
        </label>
        <label>Назва факультету

        <select name="department_name" required="required">
          <option selected hidden><%= pd[0].department_name %></option>
          <% if (pd[1].length) { %>
            <% pd[1].forEach(( { id, department_name }) => { %>
              <option><%= department_name %></option>
            <% } ) %>
          <% } %>
        </select>
      </label>

      <label>Назва книги
      <select name="book_name" required="required">
        <% if (pd[0].book_name == "WhiteBook") { %>
          <option selected hidden>WhiteBook</option>
        <% } else { %>
          <option selected hidden>BlackBook</option>
        <% } %>
        <option>WhiteBook</option>
        <option>BlackBook</option>
      </select>
    </label>
  </div>
  <div class="form-button">
    <input type="submit" value="Прийняти">
  </div>
</form>
</body>
</html>

```

A.36 Текст файлу comments.ejs

```

<!DOCTYPE html>
<html lang="en">
  <%- include('./partials/head.ejs') %>
  <body>

```

```

<%- include('./partials/nav.ejs') %>
<h1>Коментарі:</h1>
<button class="btn-add" >
  <a href="/add-comment" class="btn-add-text">Додати</a>
</button>
<ul>
  <% if (comments.length) { %>
    <% comments.forEach(({ id, professor_name, createdAt, text, visible })
=> { %>
      <article>
        <h2>
          <a><%= professor_name %></a>
        </h2>
        <p><%= text %></p>
        <div class="info">
          <span><%= createdAt.toLocaleDateString() %></span>
          <% if (visible == true) { %>
            <span>Увімкнено</span>
          <% } else { %>
            <span>Вимкнено</span>
          <% } %>
        </div>
        <a href="/edit-comment/<%= id %>" class="btn-edit">
          <i class="fas fa-edit"></i>
        </a>
        <button class="btn-delete" data-id="<%= id %>">
          <i class="fas fa-trash-alt" data-id="<%= id %>"></i>
        </button>
      </article>
    <% }) %>
  <% } %>
</ul>
<script>
  document.addEventListener('click', (e) => {
    const id = e.target?.dataset?.id || null;
    if (id) {
      fetch(`/comments/${id}`, {
        method: 'DELETE',
      }).then(() => {
        window.location.reload();
      });
    }
  });
</script>
</body>

```

```
</html>
```

A.37 Текст файлу add-comment.ejs

```
<!DOCTYPE html>
<html lang="en">
  <%- include('./partials/head.ejs') %>
<body>
  <%- include('./partials/nav.ejs') %>
  <form id="comment-form" method="post" action="/add-comment">
    <div class="form-info">
      <label>ПІБ викладача

      <select name="professor_name" required="required">
        <% if (professors.length) { %>
          <% professors.forEach(({ id, professor_name }) => { %>
            <option><%= professor_name %></option>
          <% }) %>
        <% } %>
      </select>
    </label>

    <label>Видимість
      <select name="visible" required="required">
        <option value="false">Вимкнено</option>
        <option value="true">Увімкнено</option>
      </select>
    </label>

  </div>
  <div class="form-text">
    <label>Текст коментаря
      <textarea name="text"></textarea>
    </label>
  </div>
  <div class="form-button">
    <input type="submit" value="Прийняти">
  </div>
</form>
</body>
</html>
```

A.38 Текст файлу edit-comment.ejs

```

<!DOCTYPE html>
<html lang="en">
  <%- include('./partials/head.ejs') %>
<body>
  <%- include('./partials/nav.ejs') %>
  <form id="comment-form" method="post" action="/edit-comment/<%=
cp[0].id %>?_method=PUT">
    <div class="form-info">

      <label>ПІБ викладача
      <select name="professor_name" required="required">
        <option selected hidden><%= cp[0].professor_name %></option>
        <% if (cp[1].length) { %>
          <% cp[1].forEach(({ id, professor_name }) => { %>
            <option><%= professor_name %></option>
          <% }) %>
        <% } %>
      </select>
    </label>

    <label>Видимість
    <select name="visible" required="required">
      <% if (cp[0].visible == true) { %>
        <option selected hidden>Увімкнено</option>
        <% } else { %>
          <option selected hidden>Вимкнено</option>
        <% } %>

        <option value="true">Увімкнено</option>
        <option value="false">Вимкнено</option>
      </select>
    </label>
  </div>
  <div class="form-text">
    <label>Текст коментаря
    <textarea name="text"><%= cp[0].text %></textarea>
  </label>
  </div>
  <div class="form-button">
    <input type="submit" value="Прийняти">
  </div>

```

```

</form>
</body>
</html>

```

A.39 Текст файла department.ejs

```

<!DOCTYPE html>
<html lang="en">
  <%- include('./partials/head.ejs') %>
<body>
  <%- include('./partials/nav.ejs') %>
  <article class="department">
    <div class="info">
      <span><%= dp[0].createdAt.toLocaleDateString() || 'No date'
%></span>
    </div>
    <h1><%= dp[0].department_name %></h1>
    <a href="/edit-department/<%= dp[0].id %>" class="btn-edit">
      <i class="fas fa-edit"></i>
    </a>
    <button class="btn-delete" data-id="<%= dp[0].id %>">
      <i class="fas fa-trash-alt" data-id="<%= dp[0].id %>"></i>
    </button>
  </article>

  <ul>
    <% const professors =[]; %>
    <% if (dp[1].length) { %>
      <% dp[1].forEach(function (professor) { %>
        <% if (professor.department_name === dp[0].department_name) { %>
          <% professors.push(professor); %>
          <% } %>
        <% } %>
      <% }) %>
      <% professors.forEach(({ id, professor_name, createdAt,
department_name, book_name }) => { %>
        <article>
          <h2>
            <a href="/professors/<%= id %>"><%= professor_name %></a>
          </h2>
          <div class="info">
            <span><%= createdAt.toLocaleDateString() %></span>
            <span><%= department_name %></span>
            <span><%= book_name %></span>

```

```

    </div>
    <a href="/edit-professor/<%= id %>" class="btn-edit">
      <i class="fas fa-edit"></i>
    </a>
    <button class="btn-delete" data-id="<%= id %>">
      <i class="fas fa-trash-alt" data-id="<%= id %>"></i>
    </button>
  </article>
<%= }) %>
<%= } %>
</ul>

<script>
  document.addEventListener('click', (e) => {
    const id = e.target?.dataset?.id || null;
    if (id) {
      fetch(`/departments/${id}`, {
        method: 'DELETE',
      }).then(() => {
        window.location.href = '/departments';
      });
    }
  });
</script>
</body>
</html>

```

A.40 Текст файла professor.ejs

```

<!DOCTYPE html>
<html lang="en">
  <%= include('./partials/head.ejs') %>
<body>
  <%= include('./partials/nav.ejs') %>
  <article class="professor">
    <div class="proffesor-info">
      <span><%= pc[0].createdAt.toLocaleDateString() || 'No date'
%></span>
      <span><%= pc[0].department_name %></span>
      <span><%= pc[0].book_name %></span>
    </div>
    <h1><%= pc[0].professor_name %></h1>
    <a href="/edit-professor/<%= pc[0].id %>" class="btn-edit">

```

```

    <i class="fas fa-edit"></i>
  </a>
  <button class="btn-delete" data-id="<%= pc[0].id %>">
    <i class="fas fa-trash-alt" data-id="<%= pc[0].id %>"></i>
  </button>
</article>

<ul>
  <%= const comments =[]; %>
  <%= if (pc[1].length) { %>
    <%= pc[1].forEach(function (comment) { %>
      <%= if (comment.professor_name === pc[0].professor_name) { %>
        <%= comments.push(comment); %>
      <%= } %>
    <%= }) %>
    <%= comments.forEach(({ id, professor_name, createdAt, text, visible })
=> { %>
      <article>
        <h2>
          <a><%= professor_name %></a>
        </h2>
        <p><%= text %></p>
        <div class="info">
          <span><%= createdAt.toLocaleDateString() %></span>
          <%= if (visible == true) { %>
            <span>УВІМКНЕНО</span>
          <%= } else { %>
            <span>ВИМКНЕНО</span>
          <%= } %>
        </div>
        <a href="/edit-comment/<%= id %>" class="btn-edit">
          <i class="fas fa-edit"></i>
        </a>
        <button class="btn-delete" data-id="<%= id %>">
          <i class="fas fa-trash-alt" data-id="<%= id %>"></i>
        </button>
      </article>
    <%= }) %>
  <%= } %>
</ul>

<script>
document.addEventListener('click', (e) => {
  const id = e.target?.dataset?.id || null;
  if (id) {

```

```
    fetch(`/professors/${id}`, {
      method: 'DELETE',
    }).then(() => {
      window.location.href = '/professors';
    });
  }
});
</script>
</body>
</html>
```

ДОДАТОК Б
Слайди презентації



Рисунок Б.1 – Слайд 1

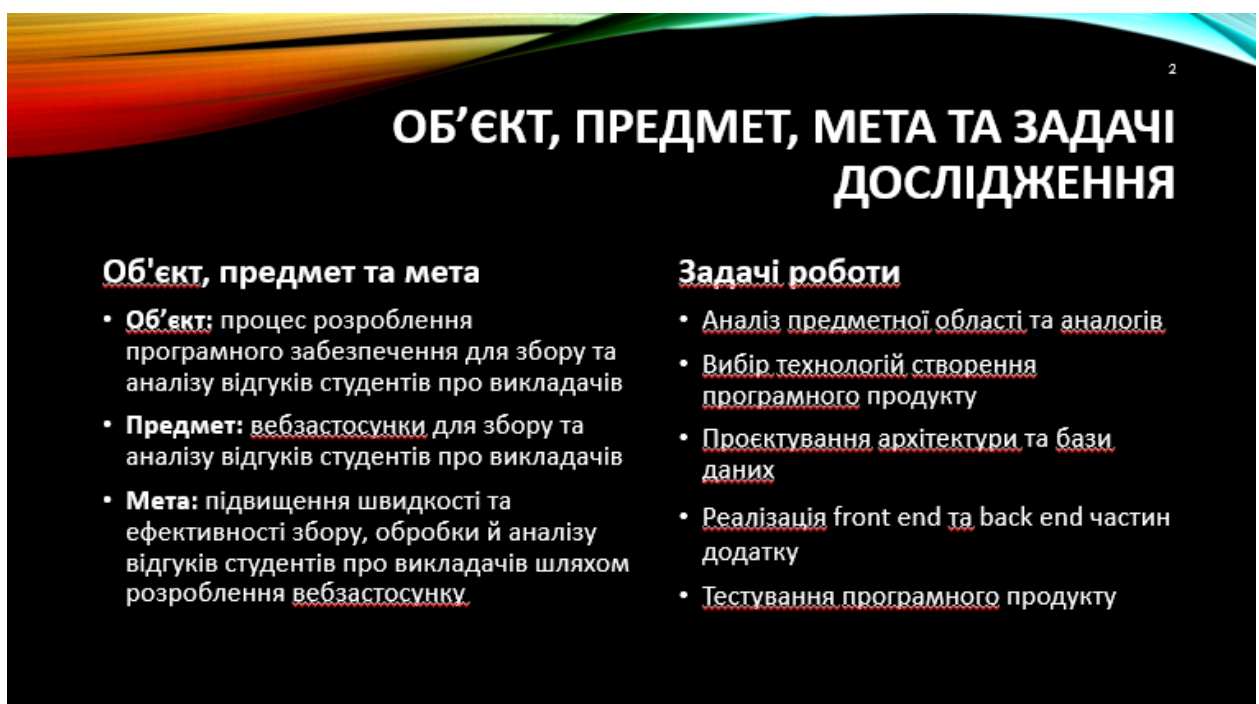


Рисунок Б.2 – Слайд 2

АНАЛІЗ ІСНУЮЧИХ АНАЛОГІВ

Критерій	Rate My Professors	The Student Room	Stud Search
Структурованість даних	Висока	Низька	Низька
Зручність пошуку інформації	Висока	Низька	Низька
Поділ відгуків на позитивні та негативні	Ні	Ні	Ні
Можливість адаптації під конкретний університет	Обмежена	Відсутня	Відсутня
Система модерації контенту	Базова	Обмежена	Відсутня
Можливість швидкого аналізу інформації	Так	Ні	Ні
Наглядність для кожного зацікавленого у навчальному процесі конкретного закладу освіти	Частково	Ні	Ні

Рисунок Б.3 – Слайд 3

ОБГРУНТУВАННЯ ВИБОРУ МОВИ ПРОГРАМУВАННЯ

Критерій	JavaScript	PHP	Python	Java	C#
Типізація	Динамічна	Динамічна	Динамічна	Статична	Статична
Виконання	Клієнт/сервер	Сервер	Сервер	Сервер	Сервер
Продуктивність	Висока	Середня	Висока	Дуже висока	Дуже висока
Масштабованість	Висока	Середня	Висока	Дуже висока	Дуже висока
Сфера використання	Вебінтерфейси, API	Сайти та CMS	Вебсистеми, AI	Корпоративні системи	Корпоративні системи
Ефективність у Front end	Дуже висока	Низька	Низька	Низька	Низька
Ефективність у Back end	Висока	Висока	Висока	Дуже висока	Дуже висока

Рисунок Б.4 – Слайд 4

5

ОБГРУНТУВАННЯ ВИБОРУ СИСТЕМИ КЕРУВАННЯ БАЗАМИ ДАНИХ

Критерій	MySQL	PostgreSQL	MongoDB	Oracle Database
Тип БД	Реляційна	Реляційна	NoSQL (документна)	Реляційна
Ліцензія	Open Source	Open Source	Open Source/Commercial	Commercial
Мова запитів	SQL	SQL	MQL	PL/SQL
Шардінг	Обмежено	Через розширення	Вбудований	Підтримується
Складність адміністрування	Низька	Середня	Середня	Висока
Основна перевага	Простота	Надійність	Гнучкість схеми	Максимальна функціональність
Основний недолік	Менше можливостей	Складніше налаштування	Менш жорстка структура	Дуже висока вартість

Рисунок Б.5 – Слайд 5

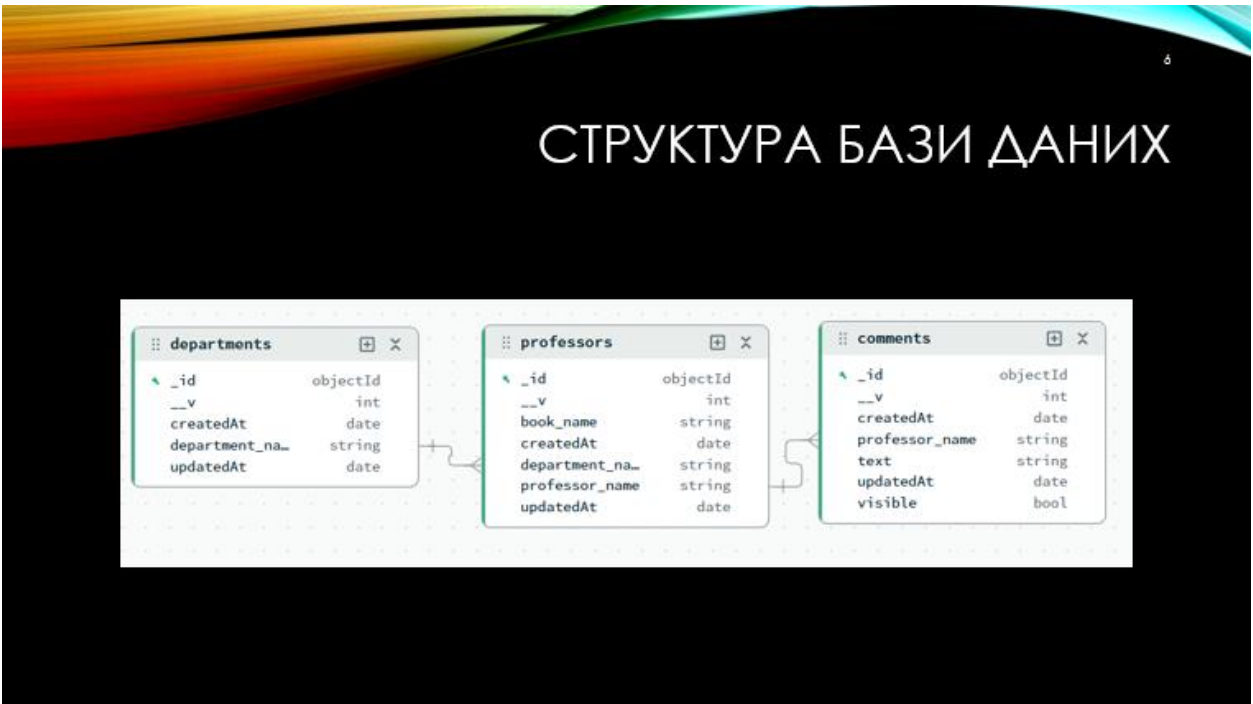


Рисунок Б.6 – Слайд 6

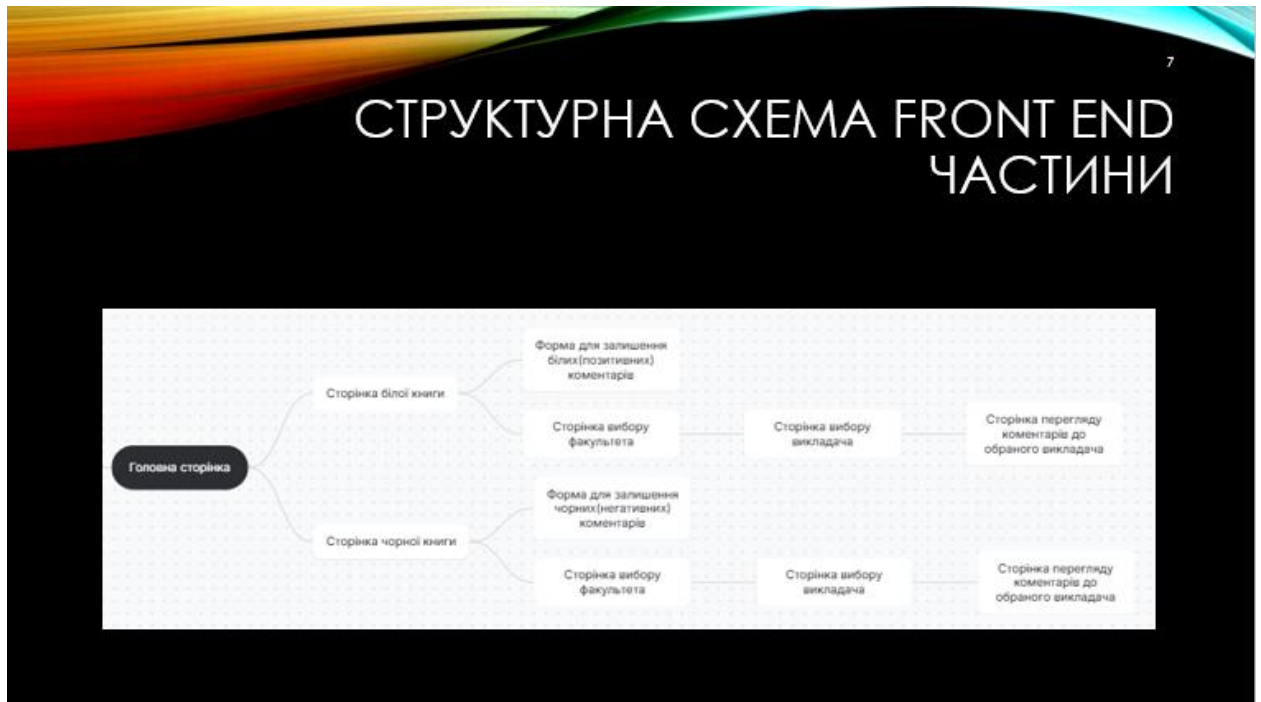


Рисунок Б.7 – Слайд 7



Рисунок Б.8 – Слайд 8

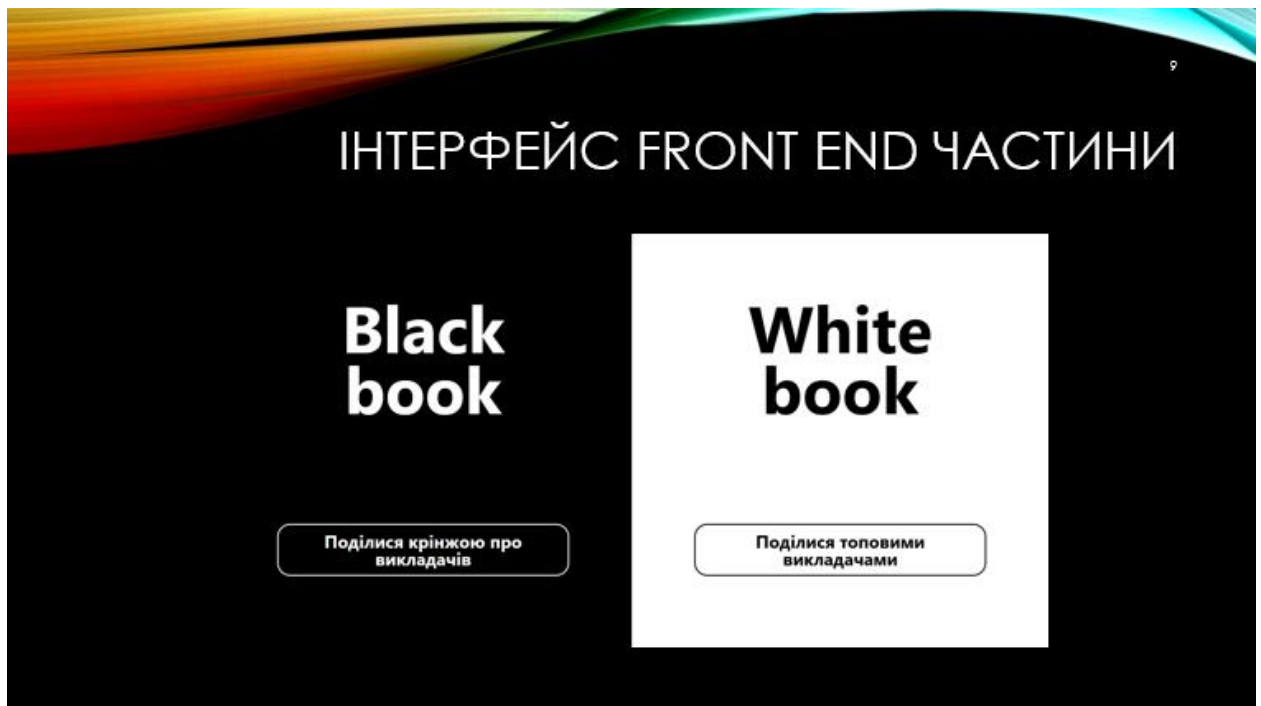


Рисунок Б.9 – Слайд 9



Рисунок Б.10 – Слайд 10

ТЕСТУВАННЯ СИСТЕМИ			ТЕСТУВАННЯ СИСТЕМИ		
Тестовий сценарій	Очікуваний результат	Фактичний результат	Тестовий сценарій	Очікуваний результат	Фактичний результат
Запуск головної сторінки	Відображення головної сторінки	Успішно	Вхід до адмін панелі з неправильним логіном	Відмова у доступі	Успішно
Відображення списку факультетів	Список факультетів	Успішно	Додавання факультету	Створення документа	Успішно
Відображення списку викладачів	Список викладачів	Успішно	Редагування факультету	Оновлення документа	Успішно
Відображення списку коментарів	Список коментарів	Успішно	Видалення факультету	Видалення з БД	Успішно
Додавання позитивного коментаря	Збереження в базі даних (БД)	Успішно	Додавання викладача	Створення документа	Успішно
Додавання негативного коментаря	Збереження в БД	Успішно	Редагування викладача	Оновлення документа	Успішно
Введення порожнього коментаря	Повідомлення про помилку	Успішно	Видалення викладача	Видалення з БД	Успішно
Відображення підтвердження	Повідомлення про успіх	Успішно	Додавання коментаря	Створення документа	Успішно
Вхід до адмін панелі з валідними даними	Доступ дозволено	Успішно	Редагування коментаря	Оновлення документа	Успішно
Вхід до адмін панелі з неправильним паролем	Відмова у доступі	Успішно	Видалення коментаря	Видалення з БД	Успішно
			Відображення викладачів, що відносяться до одного факультету	Список викладачів	Успішно
			Відображення коментарів, що відносяться до одного викладача	Список коментарів	Успішно
			Завантаження сторінки без даних	Коректне відображення	Успішно
			Вихід з адмін панелі	Повернення на головну сторінку	Успішно

Рисунок Б.11 – Слайд 11

ВИСНОВКИ	
• Проаналізовано предметну область • Описано основні рішення щодо реалізації компонентів системи • Створено <u>вебзастосунок</u> що складається з front end та back end частин • Створено методику роботи з системою користувача і програміста • <u>Забезпечено надійність</u> через покриття тестами <u>ключових компонентів</u> • <u>Практична цінність</u> полягає у створенні <u>вебзастосунку</u> для збору, зберігання та публікації відгуків студентів про викладачів університету для оцінки якості освіти	

Рисунок Б.12 – Слайд 12