

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет імені Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри інтелектуальних
інформаційних систем

_____Юрій КОНДРАТЕНКО

« ____ » _____ 20 ____ р.

КВАЛІФІКАЦІЙНА РОБОТА
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВР
ІНФОРМАЦІЙНА СИСТЕМА ПІДТРИМКИ
ОСВІТНЬОГО ПРОЦЕСУ ЗАКЛАДІВ СЕРЕДНЬОЇ
ОСВІТИ

Спеціальність 122 Комп'ютерні науки
Освітня програма «Комп'ютерні науки»

Здобувач

_____Олена БЕССОНОВА

« ____ » _____ 2025 р.

Керівник канд. техн. наук, доцент

_____Анжела БОЙКО

« ____ » _____ 2025 р.

Миколаїв – 2025

Чорноморський національний університет імені Петра Могили
(повне найменування закладу вищої освіти)

Факультет	Комп'ютерних наук
Кафедра	Інтелектуальних інформаційних систем
Рівень вищої освіти	Перший (бакалаврський)
Освітній ступень	Бакалавр
Спеціальність	122 Комп'ютерні науки
Освітня програма	Комп'ютерні науки

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних
інформаційних систем

_____ Юрій КОНДРАТЕНКО

« _____ » _____ 20__ р.

ЗАВДАННЯ
на кваліфікаційну роботу здобувача

Бессонова Олена Олександрівна

(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: _____ Інформаційна система підтримки освітнього процесу закладів середньої освіти

Керівник роботи: Бойко Анжела Петрівна, доцент кафедри КІ, кандидат технічних наук, доцент.

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

Затверджена наказом ЧНУ ім. Петра Могили від «18» листопада 2024 р. № 321.

2. Строк представлення кваліфікаційної роботи «20» червня 2025 р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні розроблена інформаційна система підтримки освітнього процесу закладів середньої освіти.

4. Перелік питань, що підлягають розробці:

– дослідження сучасних підходів до створення інформаційних систем у сфері освіти;

- визначення вимог до функціоналу інформаційної системи, що створюється, з урахуванням потреб закладу середньої освіти;
- проєктування інформаційної бази школи;
- розробка архітектури вебзастосунку;
- реалізація ключових модулів системи, включаючи управління розкладом, доступ до навчальних ресурсів і моніторинг результатів;
- тестування і впровадження системи.

5. Перелік графічних матеріалів: презентація.

Керівник роботи

(Особистий підпис)

Анжела БОЙКО
(Власне ім'я ПРІЗВИЩЕ)

Здобувач

(Особистий підпис)

Олена БЕССОНОВА
(Власне ім'я ПРІЗВИЩЕ)

Дата видачі завдання «_20_» _грудня_ 2024 р.

КАЛЕНДАРНИЙ ПЛАН

кваліфікаційної роботи

Тема: Інформаційна система підтримки освітнього процесу закладів середньої освіти.

№	Найменування роботи	Початок	Закінчення	Примітки
1	Отримання завдання на виконання КР	18.12.2024	20.12.2024	Виконано
2	Аналіз предметної області та постановка задачі	21.12.2024	30.01.2025	Виконано
3	Огляд літературних джерел за темою кваліфікаційної роботи, зокрема огляд аналогічних систем забезпечення освітнього процесу	31.01.2025	01.03.2025	Виконано
4	Аналіз сучасних освітніх вебресурсів для вирішення проблеми, та розробка дизайну інтерфейсу	02.03.2025	01.04.2025	Виконано
5	Реалізація обраних технологій з аналізом отриманих результатів	02.04.2025	15.05.2025	Виконано
6	Перший попередній захист КР на засіданні комісії кафедри	16.05.2025	16.05.2025	Виконано
7	Корегування роботи за результатами попереднього захисту	29.05.2025	29.05.2025	Виконано
8	Другий попередній захист КР на засіданні комісії кафедри	30.05.2025	30.05.2025	Виконано
9	Доробка та остаточне оформлення КР	31.05.2025	10.06.2025	Виконано
10	Подання КР, її електронної копії та інших документів (відгуку, рецензії) до захисту	11.06.2025	13.06.2025	Виконано

Керівник роботи

(Особистий підпис)

Олена БЕССОНОВА

(Власне ім'я ПРІЗВИЩЕ)

Здобувач

(Особистий підпис)

Ангела БОЙКО

(Власне ім'я ПРІЗВИЩЕ)

Дата складання календарного плану

« 30 » січня 2025 р.

АНОТАЦІЯ

до кваліфікаційної роботи
здобувачки групи 402 ЧНУ ім. Петра Могили

Бессонової Олени Олександрівни

на тему: **«ІНФОРМАЦІЙНА СИСТЕМА ПІДТРИМКИ ОСВІТНЬОГО
ПРОЦЕСУ ЗАКЛАДІВ СЕРЕДНЬОЇ ОСВІТИ»**

Актуальність розробки інформаційної системи полягає у забезпеченні централізованого доступу до загальної інформації про склад учительського колективу, навчальні програми, розклад занять, результати навчання та інші важливі аспекти шкільної діяльності. Система має сприяти автоматизації управлінських та навчальних процесів, знижуючи навантаження на адміністративний персонал і забезпечуючи прозорість та ефективність освітнього процесу. Дана розробка дозволить зробити навчальний процес прозорішим, гнучкішим та більш адаптованим до сучасних потреб.

Об'єктом кваліфікаційної роботи є освітній процес та управлінська діяльність у закладах середньої освіти.

Предметом роботи є методи та засоби проектування й розробки веборієнтованого програмного забезпечення для підтримки освітнього процесу в закладах освіти.

Метою кваліфікаційної роботи є розробка та впровадження інформаційної системи, що забезпечує ефективну підтримку освітнього процесу в закладах середньої освіти.

Робота складається зі вступу, основної частини (4 розділи), висновків та додатку. Пояснювальна записка складається зі вступу, чотирьох розділів, висновків та додатку. У вступі обґрунтовано актуальність роботи, визначено об'єкт, предмет, мету та завдання дослідження.

У першому розділі проводиться аналіз закладів середньої освіти та огляд існуючих рішень, визначення потреби в інформаційній системі у підтримці освітнього процесу.

У другому розділі описан використаний стек технологій, включаючи обрану мову, базу даних та фреймворк.

У третьому розділі описано процес проєктування вебзастосунку, розробка мокапів, діаграми використання.

У четвертому розділі описано процес розробки вебзастосунку, реалізацію серверної частин, а також тестування та розгортання системи. В результаті розроблено спеціалізовану інформаційну систему, орієнтовану на підтримку освітнього процесу закладів середньої освіти.

Кваліфікаційна робота бакалавра містить 85 сторінок, 8 таблиць, 26 рисунків, 31 джерело та 5 додатків. Ключові слова: освітній процес, вебтехнології, інформаційні системи, вдосконалення, розробка, Django, Figma, HTML, CSS.

ABSTRACT

to the qualification work by the student of the group 402 of Petro Mohyla Black Sea National University

Bessonova Olena

“INFORMATION SYSTEM FOR SUPPORTING THE EDUCATIONAL PROCESS OF SECONDARY EDUCATION INSTITUTIONS”

The relevance of developing the information system is to provide centralized access to general information about the composition of the teaching staff, curricula, class schedules, learning outcomes and other important aspects of school activities. The system should help automate management and educational processes, reducing the burden on administrative staff and ensuring transparency and efficiency of the educational process. This development will make the educational process more transparent, flexible and adapted to modern needs.

The object of qualification work is the educational process and management activities in secondary education institutions.

The subject of the work is methods and tools for designing and developing web-based software to support the educational process in educational institutions.

The purpose of the qualification work is to develop and implement an information system that provides effective support for the educational process in secondary education institutions.

The work consists of an introduction, a main part (4 chapters), conclusions and an appendix. The explanatory note consists of an introduction, four chapters, conclusions and an appendix. The introduction substantiates the relevance of the work, defines the object, subject, purpose and objectives of the study.

The first section analyzes secondary education institutions and reviews existing solutions, identifying the need for an information system to support the educational process.

The second section describes the technology stack used, including the chosen language, database, and framework.

The third section describes the process of designing a web application, developing mockups, and use cases.

The fourth section describes the process of developing a web application, implementing the server side, as well as testing and deploying the system. As a result, a specialized information system has been developed to support the educational process of secondary education institutions.

The bachelor's thesis contains 85 pages, 8 tables, 26 drawings, 31 sources and 5 appendices. Keywords: educational process, web technologies, information systems, improvement, development, Django, Figma, HTML, CSS.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ	4
ВСТУП.....	5
1 АНАЛІЗ ОСВІТНЬОЇ СФЕРИ ТА ПОСТАНОВКА ЗАДАЧІ ДО РОЗРОБКИ ІНФОРМАЦІЙНОЇ СИСТЕМИ.....	7
1.1 Аналіз вебсайтів українських та зарубіжних закладів середньої освіти	7
1.2 Основні вимоги до інформаційної системи закладу освіти	14
1.3 Технічне завдання на розробку інформаційної системи	15
Висновки до розділу 1	17
2 ВИБІР ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ	19
2.1 Вибір мови програмування	19
2.2 Вибір бази даних	22
2.3 Вибір фреймворку	25
2.4 Вибір сервісу для створення вебінтерфейсів.....	32
Висновки до 2 розділу	34
3 ПРОЄКТУВАННЯ ТА МОДЕЛЮВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ	35
3.1 Розробка діаграми використання.....	35
3.2 Опис варіантів використання.....	37
3.3 Розробка макету інформаційної системи.....	38
Висновки до 3 розділу	44
4 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ	45
4.1 Дизайн інформаційної системи.....	45
4.2 Розробка користувацької частини інформаційної системи.....	50
4.3 Розробка адмін-панелі керування системи	54
4.4 Тестування інформаційної системи.....	59
Висновки до 4 розділу	61
ВИСНОВКИ	63

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ	64
ДОДАТОК А settings.py	68
ДОДАТОК Б urls.py	70
ДОДАТОК В views.py	71
ДОДАТОК Г models.py	74
ДОДАТОК Д admin.py	78

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

CRUD - Create, Read, Update, Delete

CSS – Cascading Style Sheets

HTML – HyperText Markup Language

HTTP – HyperText Transfer Protocol

SQL – Structured Query Language

SSL – Secure Sockets Layer

UI – User Interface

UML – Unified Modeling Language

UX – User Experience

WSGI – Web Server Gateway Interface

XSS – Cross-Site Scripting

ВСТУП

Освітній процес у середній школі є важливою складовою формування знань, умінь та компетенцій молодого покоління. В умовах швидкого розвитку інформаційних технологій виникає необхідність створення інноваційних інструментів, які забезпечать оптимізацію управління навчальними процесами, покращать доступ до освітніх ресурсів та сприятимуть інтерактивній взаємодії між усіма учасниками освітнього процесу. Одним із таких інструментів є інформаційна система підтримки освітнього процесу у вигляді вебзастосунку.

Актуальність розробки такого вебзастосунку полягає у забезпеченні централізованого доступу до загальної інформації про склад учительського колективу, навчальні програми, розклад занять, результати навчання та інші важливі аспекти шкільної діяльності. Система має сприяти автоматизації управлінських та навчальних процесів, знижуючи навантаження на адміністративний персонал і забезпечуючи прозорість та ефективність освітнього процесу. Дана розробка дозволить зробити навчальний процес прозорішим, гнучкішим та більш адаптованим до сучасних потреб.

Метою кваліфікаційної роботи є розробка інформаційної системи, що забезпечує ефективну підтримку освітнього процесу в закладах середньої освіти.

Для досягнення поставленої мети були визначені такі завдання:

- дослідження сучасних підходів до створення інформаційних систем у сфері освіти;
- визначення вимог до функціоналу інформаційної системи, що створюється, з урахуванням потреб закладу середньої освіти;
- проєктування інформаційної бази школи;
- розробка архітектури вебзастосунку;
- реалізація ключових модулів системи, включаючи управління розкладом, доступ до навчальних ресурсів і моніторинг результатів;
- тестування і впровадження системи.

Об'єктом дослідження є процес розробки інформаційної системи підтримки освітнього процесу закладів середньої освіти.

Предметом дослідження є методи та інструментальні засоби розробки інформаційна система підтримки освітнього процесу в закладах середньої освіти.

Практичне значення роботи полягає у тому, що інформаційна система, що розробляється, дозволяє централізовано зберігати і обробляти дані про діяльність закладу середньої освіти, оптимізуючи його функціонування, підвищуючи прозорість і доступність інформації для учнів, вчителів та адміністрації.

1 АНАЛІЗ ОСВІТНЬОЇ СФЕРИ ТА ПОСТАНОВКА ЗАДАЧІ ДО РОЗРОБКИ ІНФОРМАЦІЙНОЇ СИСТЕМИ

1.1 Аналіз вебсайтів українських та зарубіжних закладів середньої освіти

Кожна держава, прагнучи забезпечити добробут свого народу, сприяти його розвитку та зміцненню перспектив, формує специфічні соціальні інститути, головним завданням яких є організація освіти та виховання молодого покоління.

Освіта, у свою чергу, виступає ключовим чинником інтелектуального, культурного, духовного, соціального та економічного прогресу будь-якого суспільства.

Її метою є всебічний розвиток особистості як найвищої цінності суспільства, розкриття потенціалу людини, зокрема її талантів, розумових та фізичних здібностей, а також формування моральних засад. Освіта покликана формувати свідомих громадян, здатних робити обґрунтовані суспільні вибори, і водночас збагачувати інтелектуальний, культурний і творчий потенціал нації. Крім того, вона спрямована на забезпечення економіки країни висококваліфікованими фахівцями.

Закон України «Про освіту» (ст. 3) гарантує громадянам України право на безкоштовну освіту в усіх державних навчальних закладах незалежно від статі, раси, національності, соціального і майнового стану, роду і характеру занять, світоглядних переконань, належності до партій, ставлення до релігії, віросповідання, стану здоров'я, місця проживання та інших обставин.

Це право забезпечується:

- розгалуженою мережею закладів освіти державної та інших форм власності, наукових установ;
- відкритим характером закладів освіти, створенням умов для вибору профілю навчання і виховання відповідно до здібностей, інтересів громадянина;
- різними формами навчання: стаціонарною, заочною, вечірньою, екстернатом, а також педагогічним патронатом.

Система освіти кожної держави відіграє ключову роль у забезпеченні сталого розвитку суспільства, адже вона не лише сприяє формуванню висококваліфікованого людського капіталу, але й виступає основою для розвитку інтелектуального, культурного та економічного потенціалу нації. Українська освітня система, що ґрунтується на принципах рівного доступу, інклюзивності та інноваційності, прагне інтегрувати найкращі міжнародні практики, сприяючи конкурентоспроможності своїх громадян у глобалізованому світі.

Одним із важливих аспектів сучасної освіти є її цифровізація, яка охоплює впровадження новітніх інформаційних технологій, зокрема в управлінні закладами освіти та комунікації з громадськістю. У цьому контексті офіційні вебсайти навчальних закладів стають не лише джерелом інформації, але й інструментом формування іміджу школи, засобом залучення учнів і батьків до освітнього процесу, а також платформою для впровадження сучасних форм навчання.

Дослідження практик європейських країн у створенні та розвитку вебресурсів навчальних закладів є особливо актуальним у зв'язку з потребою вдосконалення української системи освіти в умовах цифрової трансформації. Вебсайти шкіл у країнах Європи, як правило, відображають високий рівень організації, структурованості та інтерактивності. Вони виступають не лише інструментом комунікації між школою та громадськістю, але й індикатором освітніх пріоритетів держави, що дозволяє отримати цінну інформацію для порівняльного аналізу.

Аналіз вебсайтів навчальних закладів різних країн дозволяє дослідити підходи до інтеграції цифрових технологій у систему освіти, а також визначити кращі практики у створенні доступного, функціонального та інформаційно насиченого освітнього контенту. Такий аналіз може стати важливою основою для розробки рекомендацій з оптимізації українських освітніх ресурсів відповідно до сучасних стандартів і потреб суспільства.

Для аналізу з точки зору функціональності, наповнення та інтерфейсу було обрано вебсайти закладів освіти з Німеччини, України та Канади.

Шкільні вебресурси Німеччини, Канади та України демонструють суттєві відмінності в підходах до структури, дизайну, функціональності та якості представленої інформації.

Інформаційні системи закладів освіти в Німеччині характеризуються мінімалістичністю і зосередженістю на базовій функціональності. Наприклад, Albert Schweitzer Gymnasium відображає основну інформацію про школу, але має застарілий дизайн і обмежену адаптацію до мобільних пристроїв. Подібні тенденції спостерігаються у Dreilinden Gymnasium, де основний акцент зроблено на розкладі уроків та внутрішніх подіях, але бракує деталізації програм навчання та інтерактивних елементів. Навіть сучасніші вебресурси, як Diesterweg Gymnasium, демонструють простоту дизайну і базову адаптивність, однак вони не вирізняються особливим рівнем інтерактивності, що представляє заклад освіти професійно. Kant Gymnasium Berlin пропонує зручну структуру і професійний вигляд.

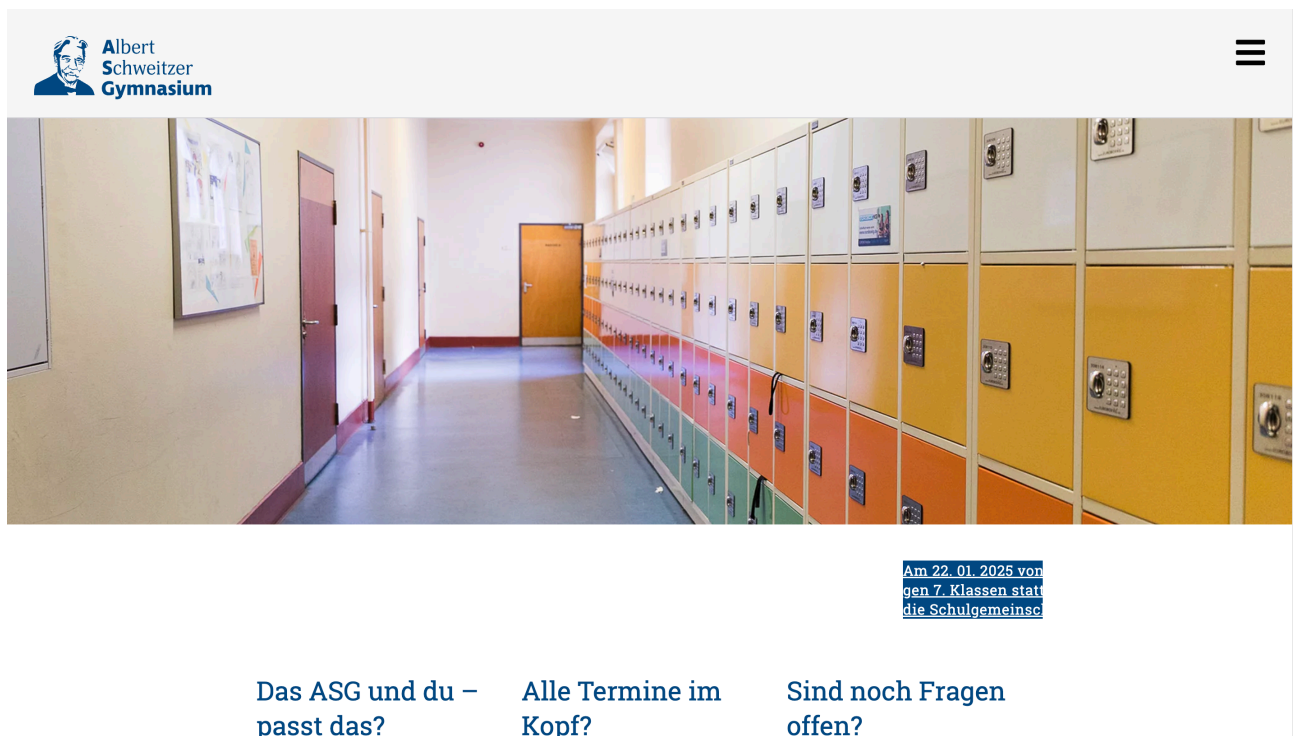


Рисунок 1.1 – Вебзастосунок зарубіжного закладу освіти

Канадські заклади освіти відзначаються сучасним дизайном, високою функціональністю та орієнтованістю на користувача. Наприклад, UIHS пропонує інтуїтивну навігацію, яскравий дизайн і повну адаптивність до різних пристроїв.

German International School Toronto вирізняється мультимовністю, інтерактивними елементами та детальним описом програм навчання, що підкреслює її міжнародний підхід. Keystone Schools демонструє високу естетику, гармонійну типографіку та інтерактивні форми для зворотного зв'язку. Усі ці вебсайти спрямовані на залучення нових учнів та забезпечення комфортної взаємодії для постійних користувачів.

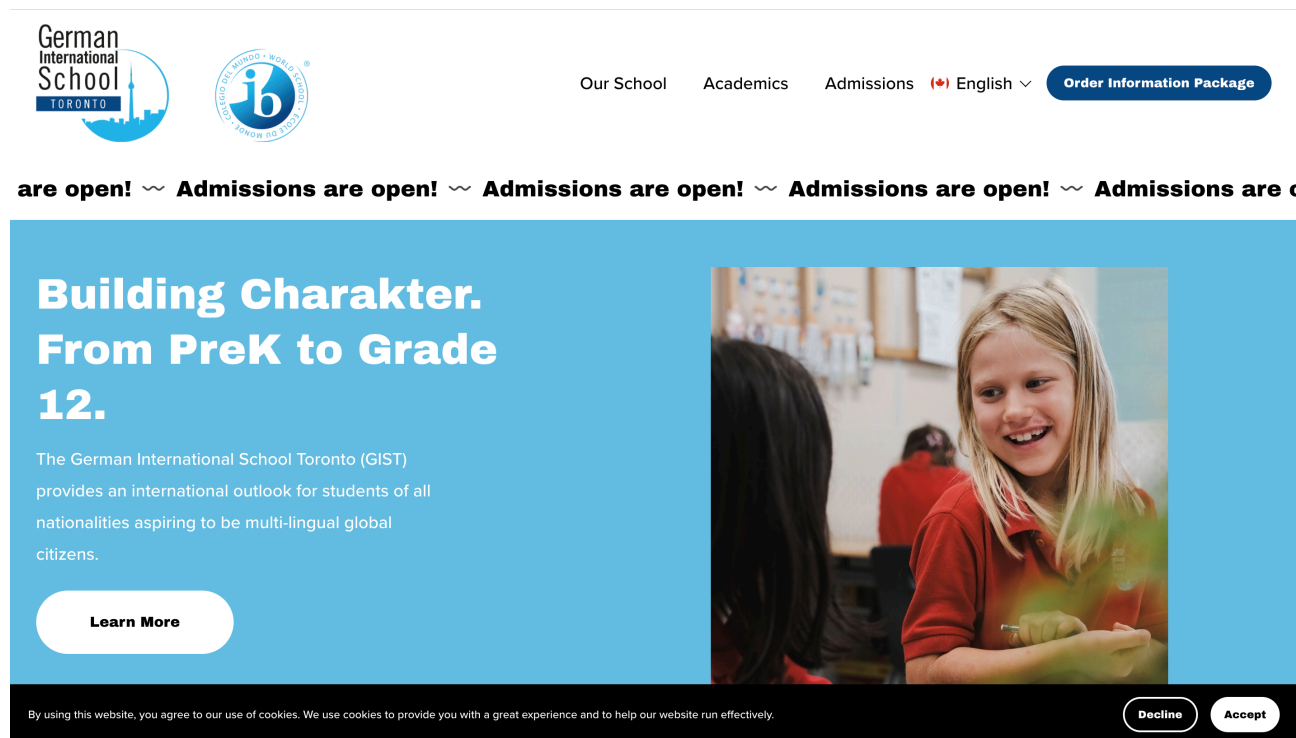


Рисунок 1.2 – Вебзастосунок зарубіжного закладу освіти

Українські вебсайти закладів загально середньої освіти, попри певні спроби структуризації інформації, поступаються за якістю дизайну та функціональністю. Це зумовлено багатим об'ємом інформації, що повинен бути представленим для загалу, згідно з Законом України «Про освіту». Загалом, українські ресурси потребують модернізації для зручного користування та представлення закладу освіти в мережі Інтернет.

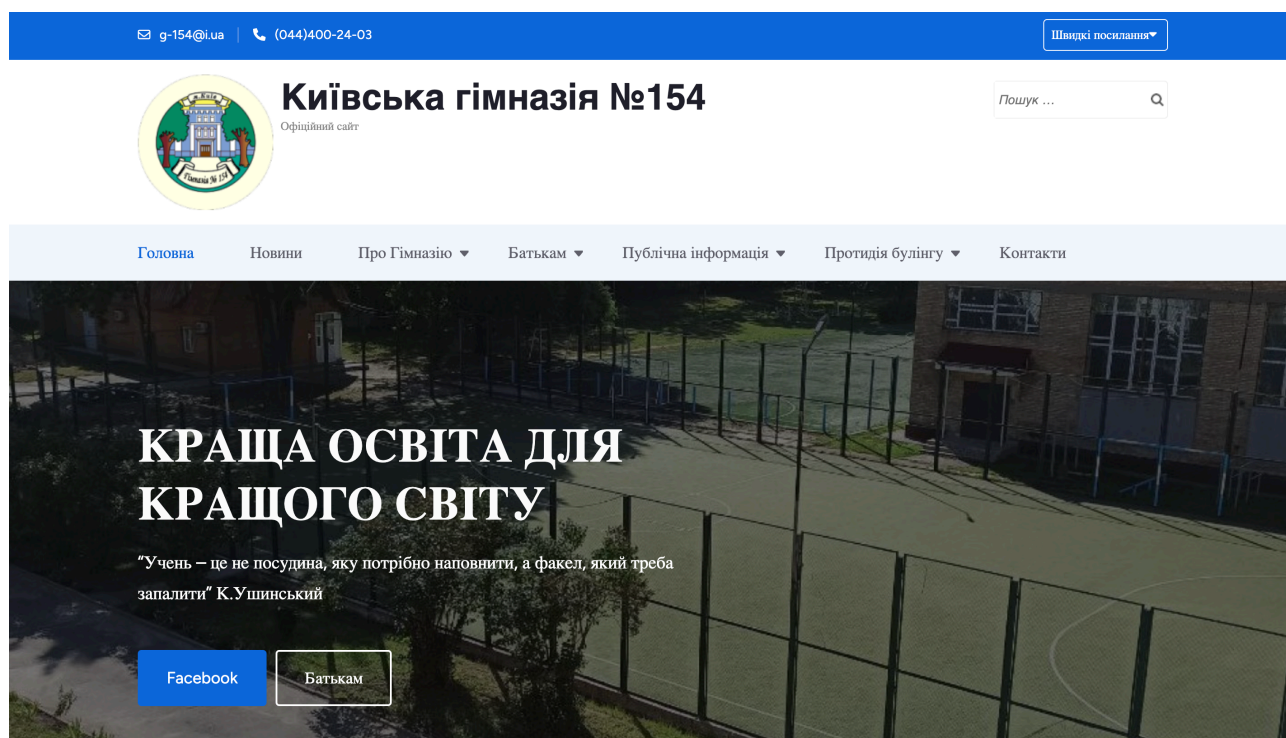


Рисунок 1.3 – Вебзастосунок українського закладу освіти

Канадські вебсайти загалом демонструють сучасний підхід до дизайну та функціональності. Вебсайти німецьких шкіл зосереджені на базових функціях, але потребують інноваційних змін для підвищення інтерактивності. Українські ресурси, в свою чергу, відчують гостру потребу в оновленні як у технічному, так і в естетичному аспектах, що сприятиме їх більшій привабливості та зручності для користувачів.

Таблиця 1.1 – Порівняльна таблиця характеристик

Критерій	Країна		
	Німеччина	Канада	Україна
Дизайн	Мінімалістичний, часто застарілий	Сучасний, естетичний, яскравий	Переважно застарілий, потребує модернізації
Функціональність	Базова, зосереджена на розкладі та внутрішніх новинах	Висока, з інтерактивними елементами і зворотним зв'язком	Частково реалізована, але обмежена через вимоги законодавства

Кінець таблиці 1.1

Адаптивність	Обмежена, не всі сайти зручні для мобільних пристроїв	Повна, з оптимізацією для різних пристроїв	Часткова, потребує покращення
Інформаційне наповнення	Основна інформація без деталізації програм навчання	Докладні описи навчальних програм, мультимовність	Об'ємне наповнення, але погано структуроване і не завжди зручне для сприйняття
Інтерактивність	Низька, відсутні сучасні елементи взаємодії	Висока, інтерактивні форми, зворотній зв'язок	Мінімальна або відсутня
Орієнтація на користувача	Обмежена	Висока, фокус на залучення нових учнів та зручність для існуючих	Низька, недостатньо враховано зручність використання
Загальна оцінка	Потребують оновлення дизайну та підвищення рівня інтерактивності	Відповідають сучасним вимогам, приклад найкращих практик	Потрібна технічна та естетична модернізація для відповідності сучасним стандартам.

У цьому контексті набуває особливої уваги реформа «Нова українська школа», яка має на меті не лише оновлення змісту освіти, але й модернізацію освітнього середовища в цілому. Один із ключових аспектів цієї трансформації – забезпечення якісної онлайн-присутності навчального закладу, що має відповідати сучасним стандартам комунікації, прозорості та доступності. Тому створення зручного, функціонального та естетично привабливого вебсайту є невід'ємною складовою розвитку сучасного освітнього закладу.

Що означає Нова Українська Школа?

Нова українська школа – це реформа Міністерства освіти і науки, яка має на меті створити школу, що забезпечуватиме учнів не лише знаннями, але й умінням

використовувати їх у повсякденному житті. Мета полягає в тому, щоб зробити процес навчання приємним для учнів.

НУШ – це школа, де учням комфортно навчатися. Тут враховують їхню думку, навчають критичному мисленню, висловленню власної думки та відповідальності як громадянина. Також батьки позитивно відгукуються про цю школу, оскільки тут існує співпраця та взаєморозуміння.

Реформа Нової української школи (НУШ) передбачена на тривалий період, оскільки неможливо швидко змінити освітню традицію, що формувалася в Україні протягом десятиліть. Проте зміни вже розпочалися, і Міністерство освіти і науки України докладає всіх зусиль для того, щоб вони були невідворотними.

Зокрема, у вересні 2017 року було ухвалено новий Закон “Про освіту”, який регулює основні засади нової освітньої системи, а у лютому 2018 року Кабінет Міністрів затвердив новий Державний стандарт початкової освіти. На черзі – ухвалення нового закону “Про загальну середню освіту”, який більш детально розкриє зміни, закладені реформою.

Новий Стандарт початкової освіти з 2017/2018 навчального року успішно проходить апробацію у 100 школах по всій Україні. З 2018/2019 навчального року за цим стандартом розпочали навчання першокласники по всій Україні.

Відповідно до нового стандарту початкової освіти, вчителі мають застосовувати інші підходи, тому протягом 2018-2019 років відбувається перенавчання вчителів початкової школи. Вони проходять дистанційне навчання (онлайн-курс на освітній платформі EdEra) та очні сесії, де спеціально підготовлені тренери допомагають закріпити знання педагогів на практиці.

Реформа "Нова українська школа" оновлює українську освіту. Якісний вебсайт навчального закладу є важливим інструментом комунікації, прозорості та взаємодії зі спільнотою. Він допомагає відповідати вимогам НУШ, підтримувати діалог з учнями і батьками та формувати позитивний імідж школи.

1.2 Основні вимоги до інформаційної системи закладу освіти

Інформаційна система закладу освіти – це не лише засіб комунікації між навчальним закладом, батьками та учнями, а й важливий елемент прозорості в роботі освітнього закладу.

Відповідно до статті 30 Закону України «Про освіту» від 05.09.2017 № 2145-VIII, кожен заклад освіти зобов'язаний створювати загальнодоступний ресурс з інформацією про свою діяльність та оприлюднювати документи відкритого доступу. Це необхідно для забезпечення права громадян на отримання інформації про освітній процес, його умови та результати.

Інформаційна система має розміщувати:

- статут закладу освіти;
- ліцензія на провадження освітньої діяльності;
- сертифікати про акредитацію освітніх програм;
- освітні програми, які реалізуються школою;
- перелік освітніх компонентів, що входять до цих програм.

Крім того, обов'язковими є:

- дані про кадровий склад закладу з інформацією про кваліфікацію педагогічних працівників;
- структура та органи управління;
- територія обслуговування, закріплена за школою;
- мова (мови) освітнього процесу;
- ліцензований обсяг;
- фактична кількість учнів;
- результати моніторингу якості освіти;
- матеріально-технічне забезпечення закладу;
- річний звіт про діяльність школи;
- правила прийому;
- додаткові освітні послуги, їх вартість та порядок надання.

Заклад освіти зобов'язаний розміщувати на своєму веб-сайті інформацію про форми здобуття освіти, зокрема про індивідуальну форму навчання відповідно до чинного законодавства, а також про можливість здобуття освіти за дистанційною формою.

Крім цього, законодавство зобов'язує державні та комунальні заклади освіти публікувати деталі процесу зарахування учнів. Не пізніше ніж за місяць до початку прийому заяв школа повинна оприлюднити інформацію про закріплену територію обслуговування, спроможність закладу приймати учнів, кількість учнів у кожному класі та наявність вільних місць.

Якщо школа проводить вступні випробування, на сайті має бути розміщено порядок вступу та зразки конкурсних завдань не менше ніж за два місяці до їхнього початку. Після завершення вступної кампанії протягом двох робочих днів школа зобов'язана оприлюднити список зарахованих учнів із зазначенням лише їхніх прізвищ, інформацію про дату, час, місце та спосіб проведення жеребкування, якщо таке проводиться, дані про кількість вільних місць і прізвища дітей, які на них претендують, а також наказ керівника закладу про створення конкурсної комісії.

Адміністрація закладу освіти зобов'язана регулярно оновлювати сайт, забезпечуючи його відповідність сучасним вимогам. Важливо, щоб вся розміщена інформація була актуальною, достовірною та відповідала вимогам законодавства.

Дотримання цих вимог забезпечує прозорість, створює довіру між школою, батьками та громадськістю, а також формує позитивний імідж закладу освіти.

1.3 Технічне завдання на розробку інформаційної системи

Практичне значення роботи полягає у тому, що інформаційна система, що розробляється, дозволяє централізовано зберігати і обробляти дані про діяльність закладу середньої освіти, оптимізуючи його функціонування, підвищуючи прозорість і доступність інформації для учнів, вчителів та адміністрації.

Для досягнення поставленої мети у розробці були визначені такі основні задачі:

- визначити вимоги до функціоналу інформаційної системи, що створюється, з урахуванням потреб закладу середньої освіти;
- розробити функціональний дизайн інформаційної системи для закладу освіти;
- визначитись з мовою програмування та фреймворком;
- обрати релевантну систему базу даних;
- реалізувати адміністративну панель та вищевказану інформаційну систему.

Постановка задачі: розробка інформаційної системи для закладу середньої освіти

Основні вимоги:

- вебзастосунок для закладу середньої освіти повинен бути функціональним і простим у використанні;
- можливість адмініструвати вебзастосунок, а саме додавати новини, редагувати інформацію;
- система повинна забезпечувати високу швидкість та безпеку.

Функціональні вимоги.

Адміністративна панель:

- керування інформацією, статтями: модерація, редагування та видалення.

Безпека даних:

- шифрування паролей адміністратора.

Нефункціональні вимоги.

Продуктивність:

- швидке завантаження сторінок.

Юзабіліті:

- зручний інтерфейс: простий та зрозумілий користувацький інтерфейс та мінімалістичний дизайн;
- адаптивний дизайн для зручного користування на будь-яких пристроях.

Технологічні вимоги:

- серверна частина реалізована на мові програмування Python з використанням фреймворку Django;
- з клієнтської частини: UI/UX дизайн реалізований за допомогою програмного забезпечення Figma, у фронтенді використані HTML, CSS, JavaScript;
- база даних – MySQL.

Етапи та терміни виконання:

- аналіз існуючого сайту, визначення переваг та недоліків (1 тиждень);
- редизайн сайту на основі аналізу за допомогою програмного забезпечення figma (2 тижні);
- реалізація backend частини, а саме адмін панель для редагування даних (4 тижні);
- реалізація frontend (html, css, javascript) на основі оновленого дизайну (4 тижні);
- тестування та вирішення багів (2 тижні).

Висновки до розділу 1

У першому розділі було проведено аналіз інформаційних систем закладів середньої освіти як за кордоном, так і в Україні. Було визначено переваги та недоліки різних програмних забезпечень, а також обґрунтовано необхідність їх оптимізації та подальшої розробки.

Враховуючи один із ключових аспектів нової української школи, а саме забезпечення якісної онлайн-присутності навчального закладу, можна зробити висновок про необхідність розробки інформаційної системи, яка буде відповідати сучасним стандартам комунікації, прозорості та доступності. Ретельний аналіз цих вимог дозволив сформулювати технічне завдання для проєктування даної системи.

2 ВИБІР ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ

Правильний вибір технологічного стеку має вирішальне значення для створення сучасного вебзастосунку, оскільки він впливає на ефективність, масштабованість та зручність подальшої підтримки і розвитку програмного продукту. На цьому етапі були ретельно проаналізовані доступні інструменти та платформи з урахуванням специфіки проєкту, технічних вимог, вимог до безпеки та можливостей інтеграції з іншими сервісами.

У даному розділі представлено основні компоненти технологічного стеку, які були обрані для реалізації вебзастосунку: мова програмування, система керування базами даних, програмний фреймворк, а також інструменти для створення користувацького вебінтерфейсу.

2.1 Вибір мови програмування

Вибір правильної мови програмування дуже важливий для будь-якого проєкту розробки. Технологія, що обирається для проєкту, має великий вплив на кінцевий результат.

Python і PHP – дві найпопулярніші мови програмування, які більшість компаній люблять використовувати для проєктів з веб-розробки.

Python – популярна багатоцільова або універсальна мова програмування, яку можна використовувати для розробки практично будь-якої комп'ютерної програми, включаючи програми та веб-сайти.

У сфері веб-розробки Python часто використовується для написання сценаріїв на стороні сервера та створення веб-додатків за допомогою таких фреймворків, як Django, Flask та Pyramid.

Python відомий своєю простотою, і саме тому багато розробників-початківців обирають Python як свою першу мову програмування. Її синтаксис абсолютно простий, а крива вивчення Python дуже плавна.

Python – це мова з динамічною типізацією. У Python розробникам не потрібно оголошувати типи даних змінних, які вони визначають. Тип змінної визначається під час виконання програми, а не на початку.

Таблиця 2.1 – Переваги та недоліки мови програмування

Переваги Python	Недоліки Python
Універсальна, об'єктно-орієнтована та універсальна мова загального призначення, яку можна використовувати в різних галузях	Не дуже корисна для мобільних обчислень, браузерів та розробки мобільних додатків
Має можливість повторного використання коду на різних платформах	Використовує велику кількість пам'яті для полегшення роботи розробника
Проста у використанні, читанні, вивченні та підтримці	Працює повільніше, ніж інші мови розробки веб-сайтів
Має функцію автоматичного збору сміття	Вносить затримки в тестування веб-додатків
Розробляє додатки з графічним інтерфейсом	Виявлення помилок ускладнюється через динамічний набір тексту
Легко інтегрується з програмним кодом інших мов, таких як Java або C++	Гігантський для простого та невеликого додатку або веб-сайту
Має бібліотеки на кшталт Tensorflow для математично інтенсивних задач	Качиний набір тексту спричиняє помилки під час виконання
Мова з відкритим вихідним кодом, що рівномірно розгортається	Трохи простіше для деяких специфічних завдань

RНР (Hypertext Preprocessor – препроцесор гіпертексту) все ще є широко використовуваною мовою програмування на стороні сервера для веб-розробки. Протягом багатьох років вона розвивалася, поповнюючись новими можливостями та оновленнями, і продовжує підтримуватися великою спільнотою розробників.

RНР активно підтримує об'єктно-орієнтоване програмування, таке як інкапсуляція даних, успадкування, абстрагування, поліморфізм тощо.

Об'єктно-орієнтоване програмування було додано в RНР5. Ця функція RНР дозволяє розробникам створювати складні багаторазові веб-сторінки і виділяє RНР поряд з такими потужними об'єктно-орієнтованими мовами, як Java і Python.

PHP – це мова з відкритим вихідним кодом, що означає, що розробникам не потрібно купувати ліцензію, перш ніж використовувати її. Крім того, PHP легше використовувати і кодувати, ніж інші мови сценаріїв.

PHP-скрипти, як правило, працюють набагато краще і швидше, ніж на інших мовах сценаріїв. Користувачі можуть завантажувати свої веб-сторінки швидше, і вони вважають це досить зручним, оскільки їм не потрібно докладати жодних зусиль.

Таблиця 2.2 – Переваги та недоліки мови програмування

Переваги PHP	Недоліки PHP
Підтримує першокласне налагодження	Уповільнена і залежна від проводів продуктивність
Має велику кількість фреймворків, що підключаються	Неможливість змінити поведінку ядра
Відкритий та об'єктно-орієнтований	Використовує слабку типізацію
Має велику екосистему	Не підходить для додатків на основі контенту
Підтримує інші інтерфейси баз даних	Відсутність альянсу з IOT
Підтримує декілька операційних систем	Недостатність протоколів та функцій безпеки

Вибір мови програмування є важливим етапом для будь-якого проєкту, оскільки від цього рішення залежить зручність розробки, можливості масштабування, швидкість роботи додатку, а також ефективність підтримки в майбутньому. У процесі аналізу було розглянуто дві популярні мови – Python та PHP. Обидві мають сильні сторони та певні обмеження. PHP широко використовується для веб-розробки, має значну екосистему та підтримує об'єктно-орієнтоване програмування. Водночас Python є універсальним та зручним для реалізації різноманітних завдань – від серверної логіки до роботи з базами даних і створення веб-додатків із використанням сучасних фреймворків.

Python вирізняється простотою, зрозумілим синтаксисом, динамічною типізацією, підтримкою численних бібліотек, а також активною спільнотою, що

сприяє швидкому вирішенню проблем під час розробки. Крім того, Python підходить для реалізації сучасних вебпроектів, що потребують швидкого прототипування, інтеграції з іншими технологіями та високої гнучкості.

З огляду на вищезазначене, було прийнято рішення обрати Python як основну мову програмування для реалізації проєкту. Це забезпечить ефективний процес розробки, легкість масштабування та підтримки системи в майбутньому.

2.2 Вибір бази даних

Вибір правильної бази даних має важливе значення для оптимізації продуктивності, забезпечення масштабованості та спрощення розробки. Кожна база даних має свої сильні сторони і підходить для певних типів додатків.

Бази даних SQL є реляційними і використовують мову структурованих запитів (SQL) для управління та маніпулювання даними. Вони відомі тим, що забезпечують цілісність схем і даних, що робить їх ідеальними для додатків, які потребують складних запитів і надійних транзакцій.

MySQL – одна з найпопулярніших реляційних баз даних з відкритим вихідним кодом. Вона широко використовується у веб-додатках, підтримується багатьма хмарними провайдерами та добре інтегрується з Python за допомогою таких бібліотек, як MySQL Connector та SQLAlchemy.

Таблиця 2.3 – Характеристика бази даних MySQL

Особливість	Опис
Надійність	Відомий своєю стабільністю та послідовністю, що робить його надійним вибором для веб-додатків.
Підтримка спільноти	Широка підтримка спільноти та документація.
Простота використання	Широка сумісність з різними інструментами та фреймворками, проста в освоєнні для початківців.
Приклад використання	Ідеально підходить для веб-додатків, систем управління контентом і сайтів електронної комерції.

PostgreSQL – ще одна популярна база даних SQL, яку часто хвалять за її розширені можливості та розширюваність. Вона пропонує підтримку даних JSON, що робить її досить універсальною для додатків, які вимагають як реляційної, так і напівструктурованої обробки даних. Розробники Python можуть підключатися до PostgreSQL за допомогою таких бібліотек, як Psycopg2 та SQLAlchemy.

Таблиця 2.4 – Характеристика бази даних PostgreSQL

Особливість	Опис
Розширені функції	Підтримка складних запитів, повнотекстового пошуку та обробки даних у форматі JSON.
Масштабованість	Висока масштабованість і здатність обробляти великі набори даних і складні операції.
Цілісність даних	Суворе відповідність вимогам ACID, що забезпечує надійність транзакцій.
Приклад використання	Підходить для аналізу даних, фінансових систем і додатків.

SQLite – це легка, файлова база даних SQL, яка входить до складу Python. Вона ідеально підходить для малих і середніх додатків або для використання в якості локальної бази даних для розробки і тестування. SQLite легкодоступна, не потребує конфігурації і може бути ефективним варіантом для невеликих проєктів.

Таблиця 2.5 – Характеристика бази даних SQLite

Особливість	Опис
Портативності	Дані зберігаються в одному файлі, що дозволяє легко передавати та інтегрувати їх у проєкти.
Налаштування не потрібне	Вбудований у Python, тому немає потреби у зовнішній конфігурації.
Обмеження	Менш підходить для середовищ з високим паралелізмом або великомасштабних застосувань.
Приклад використання	Чудово підходить для розробки, прототипування, мобільних додатків та невеликих проєктів.

Вибір найкращої бази даних для проєкту на Python залежить від декількох факторів, включаючи тип даних, вимоги до масштабованості та необхідний рівень узгодженості даних.

Таблиця 2.6 – Порівняльна характеристика

База даних	Тип	Переваги	Обмеження	Найкраще підходить для
MySQL	SQL	Надійна, широка підтримка	Менш гнучкий	Веб-додатки, електронна комерція
PostgreSQL	SQL	Розширені функції, надійна цілісність даних	Трохи складніша настройка	Аналіз даних, комплексні додатки
SQLite	SQL	Простий, портативний, не потребує налаштування	Обмежено для середовищ з високим рівнем паралелізму	Невеликі проєкти, прототипування

У процесі розробки програмного забезпечення вибір бази даних є важливим етапом, який впливає на продуктивність, масштабованість, безпеку даних і зручність підтримки додатку. Реляційні бази даних, такі як SQL-системи, забезпечують архітектуру з підтримкою складних запитів, транзакцій і цілісності даних, що робить їх корисними для проєктів, де важлива структурованість та надійність.

Порівнявши MySQL, PostgreSQL та SQLite, було проаналізовано їхні технічні можливості, зручність інтеграції з Python, простоту використання та поширені практики застосування. PostgreSQL має розширений функціонал і підходить для складних систем з високими вимогами до обробки даних. SQLite зручна для невеликих або локальних рішень, де важлива портативність. MySQL було обрано як основну базу даних для проєкту, оскільки вона забезпечує збалансовану продуктивність, стабільність та простоту використання, має активну підтримку спільноти, легко інтегрується з Python і є перевіреним рішенням для веб-додатків.

Таким чином, вибір MySQL дозволить ефективно реалізувати проєкт, зберігаючи гнучкість у розробці та надійність у роботі з даними.

2.3 Вибір фреймворку

Вибір фреймворку для розробки вебзастосунку є важливим рішенням на початковому етапі проєкту. Фреймворк визначає структуру коду, підхід до розробки (наприклад, клієнт-серверну архітектуру, розподілену логіку, компонентну модель), а також впливає на продуктивність, безпеку, масштабованість і подальшу підтримку застосунку.

У межах цього підпункту буде проаналізовано сучасні фреймворки для (Django, Flask, FastAPI) з урахуванням специфіки проєкту. Розглядатимуться такі критерії, як:

- легкість у вивченні та доступність документації;
- активність спільноти та частота оновлень;
- наявність готових рішень і бібліотек;
- підтримка масштабованої архітектури;
- інтеграція з іншими технологіями (API, бази даних, сторонні сервіси);
- досвід команди розробників.

Цей аналіз дозволить обґрунтовано обрати фреймворк, який найкраще відповідатиме технічним вимогам, ресурсам команди та очікуванням кінцевих користувачів.

Django – це фулстек веб-фреймворк, який постачається з повним набором функцій, призначених для підтримки швидкого процесу розробки. Крім того, Django має практичний дизайн і не вимагає від вас великої залежності від сторонніх пакетів, оскільки має вбудовані функції для підтримки процесу розробки.

Django – це комплексний фреймворк Python, який постачається з ключовими можливостями для розробки. До них відносяться такі функції, як маршрутизація URL-адрес, ORM (об'єктно-реляційне відображення), автентифікація та інтерфейс адміністратора.

Розробники можуть використовувати ці інструменти для швидкого та ефективного створення додатків. За допомогою маршрутизації URL-адрес можна керувати веб-трафіком, зіставляючи URL-адреси з відповідними функціями, що полегшує навігацію в додатку.

ORM Django дозволяє розробникам взаємодіяти з базами даних за допомогою об'єктів Python, а не писати SQL запити напряму. Це спрощує управління базами даних, забезпечуючи сумісність з різними системами баз даних.

Однак, хоча Django підтримує декілька серверів баз даних, його функціональність обмежена в залежності від конкретної використовуваної системи баз даних.

Django постачається з адміністративною панеллю, яка надає базові можливості налаштування для керування даними користувача та всіма зареєстрованими в ній моделями. Хоча панель за замовчуванням є потужною, потрібно погратися з деякими рядками коду для розширеної кастомізації.

Безпека зазвичай є основною проблемою у веб-розробці. Django пропонує вбудований захист від поширених ризиків безпеки, таких як SQL-ін'єкції, міжсайтовий скриптинг (XSS) та міжсайтовий підроблення запитів (CSRF). Однак, для комплексного захисту розробникам необхідно налаштувати додаткові параметри безпеки, такі як примусове застосування SSL та HTTP Strict Transport Security (HSTS).

Архітектура Django підтримує масштабованість, що чудово підходить для веб-сайтів з високим трафіком. Масштабованість залежить від конфігурацій, таких як балансування навантаження, кешування та оптимізація бази даних.

Функція кешування в Django допомагає зменшити навантаження на сервер, зберігаючи в пам'яті дані, до яких часто звертаються. Перевагою Django є його масштабована архітектура, яка легко адаптується до зростаючого попиту і робить додаток чуйним.

Плюси використання:

Універсальний фреймворк

Django надає повний пакет, мінімізуючи залежність від сторонніх інструментів, що прискорює розробку та зменшує складність.

Функції безпеки

Вбудовані засоби безпеки Django, такі як захист від SQL-ін'єкцій та CSRF-атак, за замовчуванням допомагають захистити додатки від поширених вразливостей.

Масштабована архітектура

Архітектура Django підтримує масштабованість додатків, що дозволяє ефективно працювати з великими наборами даних та високим користувацьким навантаженням. Однак ефективна масштабованість також залежить від вибору інфраструктури, такої як кешування, оптимізація баз даних та конфігурації сервера.

Швидка розробка

Завдяки широкому спектру вбудованих функцій та простому у використанні інтерфейсу, розробники можуть швидко створювати додатки за допомогою Django.

Мінуси використання

Крутіша крива навчання

Через свої складні функції робота з Django може бути досить складною для початківців. Тим не менш, розробники можуть використовувати його для невеликих проєктів, поступово нарощуючи свій досвід.

Монолітна структура

Структурований підхід Django допомагає підтримувати узгодженість та найкращі практики. Однак він може обмежувати гнучкість розробників, коли вони надають перевагу більш модульній або настроюваній архітектурі.

Надмірність для невеликих проєктів

Для простих додатків або легких проєктів широкий набір функцій Django може додати непотрібну складність і накладні витрати, що робить його менш придатним у порівнянні з більш легкими фреймворками.

Flask – це легкий мікро-фреймворк, розроблений для простоти та гнучкості. На відміну від Django, який пропонує широкий спектр вбудованих функцій, Flask дозволяє розробникам додавати компоненти за потребою, що дає їм більше контролю над структурою та інструментами проєкту.

Flask надає розробникам більше гнучкості та контролю над структуруванням їхніх додатків. Він пропонує основні компоненти для веб-розробки та дозволяє розробникам додавати компоненти відповідно до їхніх вимог. Нижче наведено деякі з функцій, які роблять Flask популярним вибором:

Мінімальне ядро

Flask надає лише базові інструменти для запуску вебдодатків, такі як маршрутизація та обробка запитів. На противагу цьому, Django включає широкий спектр вбудованих функцій, що робить Flask більш легкою та гнучкою альтернативою для тих, хто вважає за краще обирати компоненти самотійно.

Гнучкість

Flask не нав'язує жодної конкретної структури проєкту або шаблону розробки, дозволяючи розробникам пристосовувати фреймворк до своїх конкретних потреб і пропонуючи максимальну кастомізацію.

У Django існує жорстка архітектура, тому потрібно дотримуватися певних умовностей. Але за допомогою Flask можна вільно керувати файлами, структурувати каталоги та реалізовувати функції.

Екосистема розширень

Мінімалістичний дизайн Flask доповнюється великою екосистемою розширень. Розробники можуть додавати такі функції, як автентифікація, інтеграція з базами даних та обробка форм за потреби.

Хоча Flask має невелику вагу, розробники можуть обирати потрібні їм компоненти, що полегшує створення кастомізованих додатків. Ці розширення включають Flask-SQLAlchemy (інтеграція з базами даних), Flask-WTF (обробка форм) та інші.

Переваги Flask

Flask дуже легкий, що дозволяє розробникам розширювати додатки або масштабувати їх відповідно до своїх потреб. Крім того, гнучкість дає розробникам повний контроль над структурою проєкту. Деякі ключові переваги, перелічені нижче, роблять Flask першим вибором для розробників:

Легкість та простота

Простота використання та мінімальне налаштування Flask роблять його ідеальним фреймворком для швидкого створення прототипів та мінімально життєздатних продуктів (MVP).

Більший контроль

Flask надає розробникам повний контроль над структурою та компонентами їхніх додатків, що дозволяє створювати висококастомізовані рішення.

Швидке створення прототипів

Простота Flask робить його ідеальним для швидкої розробки, дозволяючи розробникам швидко створювати прототипи.

Проблеми з Flask

Хоча Flask пропонує гнучкість, його мінімалістичний характер створює певні проблеми, особливо для великих проєктів. Ось деякі з них:

Ручне налаштування

Flask вимагає від розробників ручного налаштування основних компонентів, таких як автентифікація, ORM та безпека, що може збільшити час розробки складних додатків. Хоча існують деякі розширення, їх налаштування може збільшити час розробки складних додатків.

Менше вказівок

Оскільки Flask не вимагає певної структури або шаблону розробки, може бути важко підтримувати узгодженість у великих проєктах або командах.

Не ідеально підходить для великомасштабних додатків

Flask ідеально підходить для створення невеликих проєктів та API, але йому бракує вбудованих інструментів, необхідних для управління великими, складними додатками без залежності від зовнішніх бібліотек для підтримки.

FastAPI – ще один фреймворк Python, який варто розглянути, особливо для створення високопродуктивних API. Він чудово справляється з асинхронними завданнями, особливо при роботі з паралельними запитами. ,

FastAPI створений для швидкої розробки API, з акцентом на швидкість та продуктивність. Він особливо підходить для додатків і проєктів в режимі реального часу, які потребують асинхронних можливостей, що дозволяють обробляти декілька запитів одночасно.

Плюси використання

Продуктивність

Розроблений для швидкості, FastAPI підтримує асинхронну обробку і двонаправлені веб-сокети (завдяки Starlette). Він перевершив Django і Flask в тестах, що робить його ідеальним для додатків з високим трафіком.

Масштабованість

Як і Flask, FastAPI є високомодульним, що робить його простим у масштабуванні та ідеальним для контейнерних розгортань.

Відповідність галузевим стандартам

FastAPI повністю сумісний з OAuth 2.0, OpenAPI (раніше Swagger) і JSON Schema. Як результат, можливість реалізувати безпечну автентифікацію та створити документацію API з мінімальними зусиллями.

Простота використання

FastAPI використовує Pydantic для підказок типів і валідації, що прискорює розробку завдяки перевірці типів, автозавершенню і валідації запитів.

Документація

FastAPI постачається зі значним обсягом документації та зростаючими сторонніми ресурсами, що робить його доступним для розробників усіх рівнів.

Мінуси використання

Зрілість

FastAPI, будучи новішим, не має такої зрілості, як Django або Flask. Його спільнота менша, а користувацький досвід може бути менш спрощеним через менш широке використання.

Сумісність

Як мікрофреймворк, FastAPI потребує додаткової функціональності для повноцінних додатків. Існує менше сумісних бібліотек у порівнянні з Django або Flask, що може вимагати від вас розробки власних розширень.

Django – чудовий варіант, якщо потрібен стандартний функціонал веб-додатків «з коробки», що робить його придатним для проєктів, які потребують більш надійної структури. Це особливо вигідно, якщо використовується реляційна баа даних, оскільки її ORM спрощує управління даними і надає вбудовані функції безпеки. Однак, широкі функціональні можливості можуть виявитися непосильними для невеликих проєктів або простих додатків.

Flask, з іншого боку, пропонує більшу гнучкість. Його мінімалістичний дизайн дозволяє розробникам обирати розширення та бібліотеки, які вони хочуть, що робить його придатним для проєктів, де потрібно кастомізувати функції. Цей підхід добре працює для стартапів або MVP, де ваші вимоги можуть швидко змінюватися і розвиватися. Хоча з Flask легко почати роботу, майте на увазі, що створення більш складних додатків означатиме вивчення різних розширень.

FastAPI – сильний конкурент, коли швидкість має вирішальне значення, особливо для проєктів з використанням API або машинного навчання. Він використовує сучасні можливості Python, такі як підказки типів, щоб забезпечити автоматичну перевірку даних і документацію. FastAPI – чудовий вибір для додатків, які потребують високої продуктивності, таких як мікросервіси або API, керовані даними. Незважаючи на це, він може бути не таким багатим на вбудовану функціональність, як Django або Flask, що означає, що може знадобитися реалізувати додаткові функції вручну.

Таблиця 2.7 – Порівняльна характеристика

	Django	Flask	FastAPI
Вид	Full-stack фреймворк, розроблений для вебдодатків.	Легкий мікрофреймворк сервера.	Легкий мікрофреймворк для побудови веб-API.
Простота використання	Все необхідне знаходиться в коробці, що прискорює розвиток.	Високий рівень гнучкості у виборі бажаних бібліотек і розширень.	Підказки щодо введення та валідація прискорюють розробку
Розширення	Найбільший вибір сумісних пакетів з трьох.	Велика кількість сумісних пакетів.	Менше сумісних пакетів, ніж у Flask або Django.
Продуктивність	Непогано, але не так швидко, як Flask або FastAPI.	Трохи швидший, ніж Django, але не такий продуктивний, як FastAPI.	Найшвидший з трьох.

На основі проведеного аналізу було прийнято рішення використовувати фреймворк Django завдяки його здатності забезпечувати швидку розробку, високий рівень безпеки та наявність вбудованої адміністративної панелі. Ці характеристики роблять його ідеальним вибором для шкільного вебсайту з можливістю редагування інформації.

2.4 Вибір сервісу для створення вебінтерфейсів

Figma – це інструмент для проєктування користувацького інтерфейсу (UI), який в першу чергу використовується для створення прототипів та спільної роботи. Figma є повністю веб-орієнтованою, що означає, що вона працює безпосередньо у вашому браузері; не потрібно завантажувати або встановлювати будь-яке програмне забезпечення для її використання.

Figma надає:

- створення статичних каркасів і макетів для веб-сайтів і мобільних додатків;
- створення інтерактивних прототипів (знову ж таки, для веб-сайтів і мобільних додатків);
- створювати та керувати системами дизайну з компонентами та стилями багаторазового використання;
- створювати презентації, логотипи, іконки, інфографіку та пости в соціальних мережах;
- проводити віртуальні воркшопи та спільні мозкові штурми (за допомогою figjam, інструменту цифрової дошки від figma).

Figma проста у вивченні та використанні: Figma славиться своїм інтуїтивно зрозумілим, зручним інтерфейсом. Початківці та експерти можуть швидко освоїти цей інструмент, що робить його ідеальним для всього: від графічного дизайну до детальних проєктів UI/UX дизайну.

Це браузерний інструмент, завантажувати його не потрібно.

Figma покращує співпрацю: Figma дозволяє легко працювати в команді. Це велика перевага для віддалених і розподілених команд.

Контроль версій і автоматичне збереження: Figma автоматично зберігає зміни в хмарі, що дозволяє легко повернутися до попередніх версій, якщо це необхідно, і зменшує ризик втрати вашої роботи.

Вражаючий набір функцій і можливостей: Figma має все необхідне для створення каркасів, прототипів і багатьох інших ресурсів для дизайну.

Потужна спільнота та додаткові ресурси: Спільнота Figma є чудовим джерелом натхнення та навчання. Користувачі Figma мають доступ до розроблених спільнотою шаблонів, ресурсів для дизайну та плагінів.

Висновки до 2 розділу

У цьому розділі було проведено аналіз мов програмування, фреймворків, засобів розробки інтерфейсу та систем керування базами даних для обґрунтування вибору інструментів для реалізації програмного продукту. Серед мов програмування було розглянуто Python та PHP, причому Python був обраний через його читабельність, велику спільноту та широкий вибір бібліотек.

Аналізуючи веб-фреймворки для Python, обрали Django за високу швидкість розробки, вбудовану адміністративну панель, підтримку ORM та відповідність вимогам для повнофункціонального веб-застосунку.

Для проєктування інтерфейсу користувача обрано інструмент Figma, який забезпечує ефективну командну роботу та інтерактивне прототипування.

Проаналізовано MySQL, PostgreSQL та SQLite. Для масштабованості, продуктивності та сумісності обрано MySQL як оптимальне рішення.

Таким чином, для реалізації програмного забезпечення було обрано стек технологій: мова програмування Python, фреймворк Django, засіб проєктування інтерфейсу Figma та система керування базами даних MySQL.

3 ПРОЄКТУВАННЯ ТА МОДЕЛЮВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

Процес створення інформаційної системи включає попереднє проєктування її структури, сценаріїв взаємодії користувача із системою та візуального представлення майбутнього інтерфейсу. На цьому етапі здійснюється моделювання ключових функцій системи, визначаються ролі користувачів та описується їх взаємодія із програмним продуктом. Це дозволяє зрозуміти логіку роботи системи, забезпечити її зручність та ефективність до початку програмної реалізації.

Проєктування є важливим етапом у створенні та супроводі інформаційної системи. Добре розроблена модель допомагає зменшити кількість помилок під час розробки, а також полегшує підтримку та розширення функціоналу в майбутньому.

3.1 Розробка діаграми використання

Діаграма варіантів використання (Use Case Diagram) є одним із ключових інструментів моделювання інформаційних систем у нотації UML (Unified Modeling Language). Її основна функція полягає в тому, щоб відобразити функціональність системи з перспективи зовнішніх користувачів, тобто показати, які операції можуть виконувати різні категорії користувачів під час взаємодії з системою. Така діаграма дозволяє отримати загальне уявлення про функціональні можливості майбутньої системи ще до початку її розробки.

Основними елементами діаграми є актори, варіанти використання, зв'язки між ними, а також додаткові логічні відношення типу *include* або *extend*. Актор являє собою зовнішнього учасника, який взаємодіє з системою; це може бути як людина, так і інша система. Варіант використання представляє окрему функціональну можливість, яку система надає актору. Зв'язки між акторами та варіантами використання демонструють, які саме дії доступні кожному з них.

Одним з важливих етапів розробки інформаційної системи є визначення обов'язкових та додаткових сценаріїв використання. Для цього застосовуються механізми *include* та *extend*. Механізм включення (*include*) означає, що певна функція є невід'ємною складовою іншої дії. Наприклад, перед тим як адміністратор

зможє редагувати або додавати інформацію, він має пройти авторизацію, тому дія "авторизація" буде включена в усі інші адміністративні функції. Розширення (extend), навпаки, використовується для опису умовних або додаткових дій, які не є обов'язковими, але можуть бути виконані за певних умов.

Діаграма варіантів використання дозволяє уникнути неоднозначностей у визначенні функціональних вимог до системи, забезпечує краще розуміння між розробниками і замовниками, а також є основою для подальшого створення технічного завдання, розробки сценаріїв тестування та деталізації моделі системи. У нашому випадку ця діаграма відображає ключові функції інформаційної системи для закладу середньої освіти, такі як перегляд, додавання, редагування та видалення інформації, з чітким розподілом доступу між користувачем та адміністратором.



Рисунок 3.1 – Діаграма використання інформаційної системи

У діаграмі задіяні два актори: Користувач та Адміністратор. Користувач має доступ лише до одного варіанту використання – перегляд інформації, що відповідає його ролі як звичайного відвідувача сайту. Це узгоджується з вимогою, що система

повинна бути доступною для публічного ознайомлення з інформацією без можливості її змінення.

Адміністратор, у свою чергу, володіє розширеним набором повноважень. Окрім перегляду інформації, йому доступні такі функції як авторизація, додавання, редагування та видалення інформації. Усі ці дії взаємопов'язані через механізм <<include>>, що означає обов'язковість попередньої авторизації перед виконанням будь-якої адміністративної дії (редагування, додавання чи видалення інформації).

3.2 Опис варіантів використання

Сценарій використання – це опис взаємодії між користувачем і системою для реалізації функції або задачі. Він використовується для аналізу вимог до системи та для визначення деталей функціональності. Сценарії використання можуть бути представлені у короткій, поверхневій та повній формах.

Сценарій №1 (коротка форма): Перегляд інформаційної системи користувачем.

Користувач відкриває сайт школи через браузер. На головній сторінці або в меню він обирає потрібно йому вкладку . Система автоматично завантажує всю відповідну інформацію у повному форматі, що надає можливість переглянути бажане. До прикладу можна привести розділ «Новини». Кожна новина одразу містить заголовок, дату публікації, повний текст і, за необхідності, зображення. Користувач може без додаткових дій прокручувати сторінку вниз і знайомитися з усіма новинами, що доступні на сайті. Якщо новини тимчасово відсутні, система повідомляє про це відповідним повідомленням. Користувач у такому випадку може перейти до інших розділів сайту або завершити перегляд. У результаті користувач отримує доступ до актуальної інформації у зручному та відкритому форматі без авторизації чи додаткових переходів.

Сценарій №2 (коротка форма): Управління контентом на сторінках.

Адміністратор, щоб додати певну інформацію на сайт (новина, інформація про вчителя, оновлення при вступі), спочатку переходить на сторінку авторизації, де вводить логін і пароль. Якщо дані введено правильно, система відкриває адміністративну панель. До прикладу адміністратору потрібно додати новину, він переходить до розділу «Новини», де може обрати опції редагування, видалення та додавання. Для додавання нової інформації, адміністратор заповнює форму, вводячи заголовок, основний текст та додаючи зображення, якщо це потрібно. Після натискання кнопки «Опублікувати» система перевіряє правильність заповнення полів і зберігає оновлення в базі даних. Якщо всі дані коректні, допис одразу з'являється у публічному розділі сайту. Якщо при додаванні новини було пропущено обов'язкові поля, наприклад заголовок або текст, система попереджає про помилку та не дозволяє завершити створення допису до усунення недоліків. У результаті адміністратор може ефективно керувати інформацією, яка з'являється на сайті, з дотриманням необхідних перевірок та захистом доступу. Якщо потрібно відредагувати контент на сторінці, адміністратор обирає її у переліку, натискає кнопку редагування, після чого змінює дані у формі та зберігає. Після успішної перевірки система оновлює дані та відображає повідомлення про це. Якщо необхідно видалити новину, адміністратор обирає її у переліку та натискає кнопку видалення і підтверджує дію.

3.3 Розробка макету інформаційної системи

Процес проєктування інформаційної системи включає створення вайрфрейму (wireframe) – схематичного макету інтерфейсу користувача. Вайрфрейм ілюструє структуру сторінок сайту або програми, показуючи розташування меню, кнопок, заголовків, текстових блоків, зображень тощо.

Наявність вайрфрейму дозволяє на ранніх етапах візуалізувати, як виглядатиме система з погляду користувача, і зрозуміти логіку взаємодії з елементами інтерфейсу. Це дає змогу заздалегідь оцінити зручність користування,

виявити потенційні проблеми у навігації або розташуванні функціональних блоків, ще до початку програмної реалізації. Вайрфрейм допомагає узгодити бачення між замовником, дизайнерами та розробниками, зменшуючи ризик непорозумінь і потреби у дорогому переробленні інтерфейсу на пізніших стадіях розробки.

Дуже важливо, щоб wireframes були простими. Це пов'язано з тим, що мета вайрфрейму, як зазначалося вище, полягає в тому, щоб показати, як елементи розміщені на сторінці і як повинна працювати навігація на сайті. Занадто багато кольорів або зображень може відволікати читача від зосередження на макеті та елементах навігації. Переконайтеся, що ви виключили або зменшили використання кольору, зображень, графіки і стилізованих шрифтів, щоб досягти своєї мети – зобразити просте візуальне представлення скелету.

Для початку окреслюється структура майбутнього продукту, задля розуміння завдання. На рисунку 3.2 наведено ієрархію сторінок інформаційної системи.

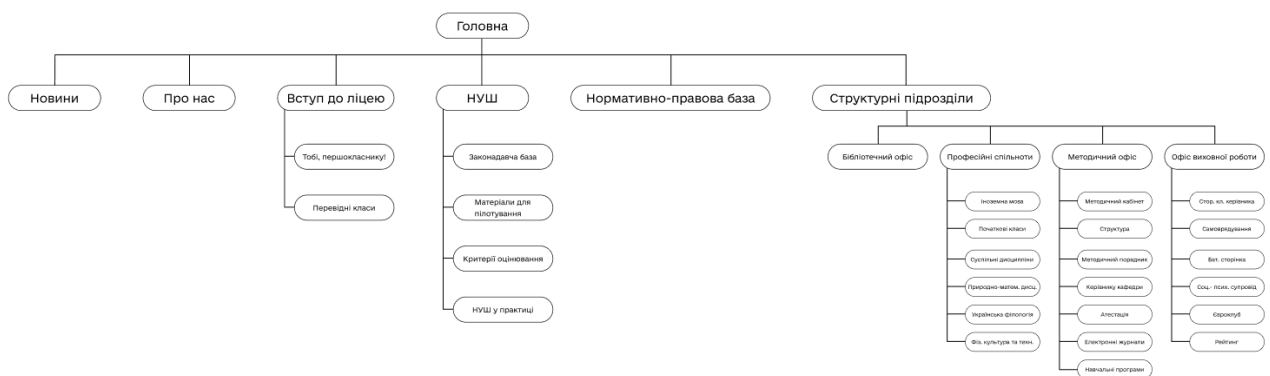


Рисунок 3.2 – Ієрархія вебзастосунку

У Figma було створено макет інформаційної системи, в якому відображено основні елементи інтерфейсу користувача. Макет демонструє загальну структуру сторінок, включаючи розміщення меню, текстових блоків, заголовків, кнопок та інших компонентів, необхідних для зручної взаємодії з сайтом.

Далі продемонстровано основні макети системи на рисунках 3.3-3.8.

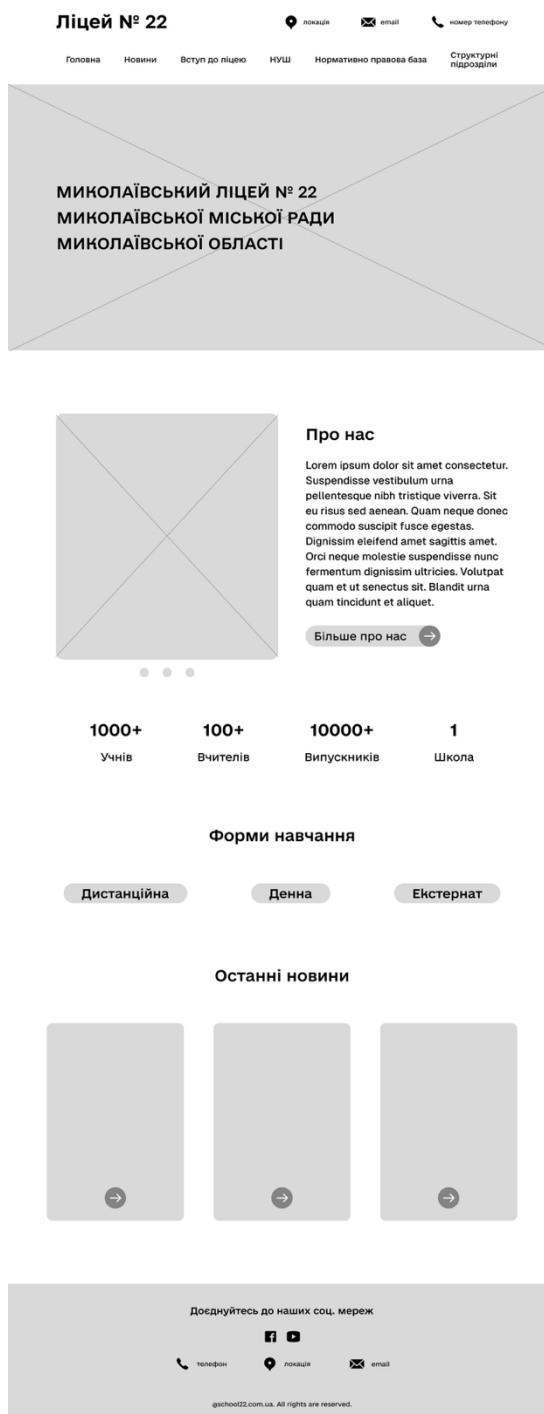


Рисунок 3.3 – Головна сторінка системи



Рисунок 3.4 – Сторінка «Новини»

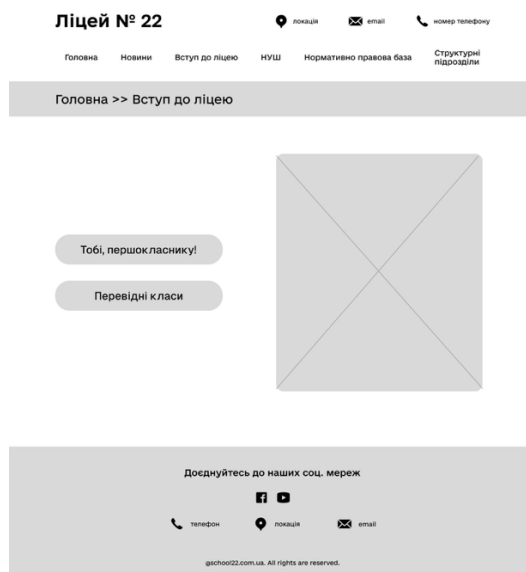


Рисунок 3.5 – Макет дочірніх сторінок

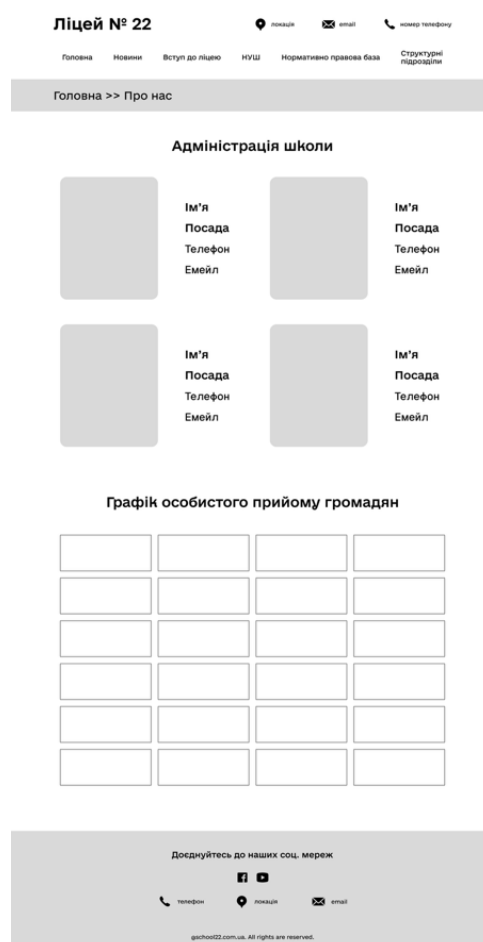


Рисунок 3.6 – Сторінка «Про нас»



Рисунок 3.7 – Макет інформаційної сторінки



Рисунок 3.8 – Макет інформаційної сторінки

Створення макету інформаційної системи в Figma дозволило візуалізувати дизайн сайту, ретельно спланувати розташування основних елементів інтерфейсу та забезпечити зручність користування ще до початку етапу розробки.

Висновки до 3 розділу

У процесі проєктування було розроблено діаграму варіантів використання, сценарії взаємодії користувача із системою, а також макет майбутнього інтерфейсу. Це дозволило точно визначити функціональність інформаційної системи, розмежувати ролі користувачів (адміністратор і звичайний користувач) та детально описати їх дії в межах системи. Завдяки застосуванню UML-діаграм, зокрема діаграми використання, вдалося візуалізувати логіку роботи системи, що забезпечило глибше розуміння архітектури проєкту ще до його реалізації.

Створення макету за допомогою Figma дозволило спроектувати зовнішній вигляд сайту, розташування ключових елементів та структуру сторінок. Важливим аспектом стало створення простого та інформативного вайрфрейму, що допомогло виявити потенційні недоліки у навігації або дизайні на ранньому етапі. Таким чином, проєктування стало фундаментальною частиною процесу створення інформаційної системи, сприяючи її подальшій ефективній реалізації, підтримці та розвитку.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

Цей розділ описує етапи програмної реалізації створеної інформаційної системи. Реалізація охоплює розробку зовнішнього вигляду та логіки роботи користувацького інтерфейсу, проектування функціональної адміністративної панелі, а також проведення тестування для перевірки працездатності системи та відповідності встановленим вимогам.

Наведений матеріал ілюструє всебічний підхід до розробки та впровадження інформаційної системи, охоплюючи всі етапи від проектування інтерфейсу до остаточного тестування програмного продукту.

4.1 Дизайн інформаційної системи

Дизайн є ключовим елементом розробки інформаційної системи, оскільки він визначає зручність, зрозумілість та ефективність взаємодії користувача з програмним продуктом. На цьому етапі розробляється зовнішній вигляд інтерфейсу, структура сторінок, розташування елементів управління, а також загальний стиль системи. При створенні дизайну враховуються принципи юзабіліті та адаптивності, щоб забезпечити комфортну роботу з системою на різних пристроях.

Першим етапом, окрім попереднього аналізу аналогів, був вибір кольорової гами та шрифту для оформлення інформаційної системи.

Згідно з наданим завданням були визначені основні 6 кольорів, що відображають настрій закладу освіти.



Рисунок 4.1 – Основні кольори інформаційної системи

Наступним етапом реалізації є вибір шрифтів. Оскільки розробляється інформаційна система для державного закладу освіти, були обрані шрифти «Цифрової держави», а саме e-Ukraine та e-Ukraine Head.

Ліцей № 22
e-Ukraine Head (Main)
Ліцей № 22
e-Ukraine (Regular)

Рисунок 4.2 – Обрані шрифти

Після визначення основних моментів був створений основний дизайн інформаційної системи.

На рисунках 4.3 – 4.9 наведений дизайн основних сторінок системи

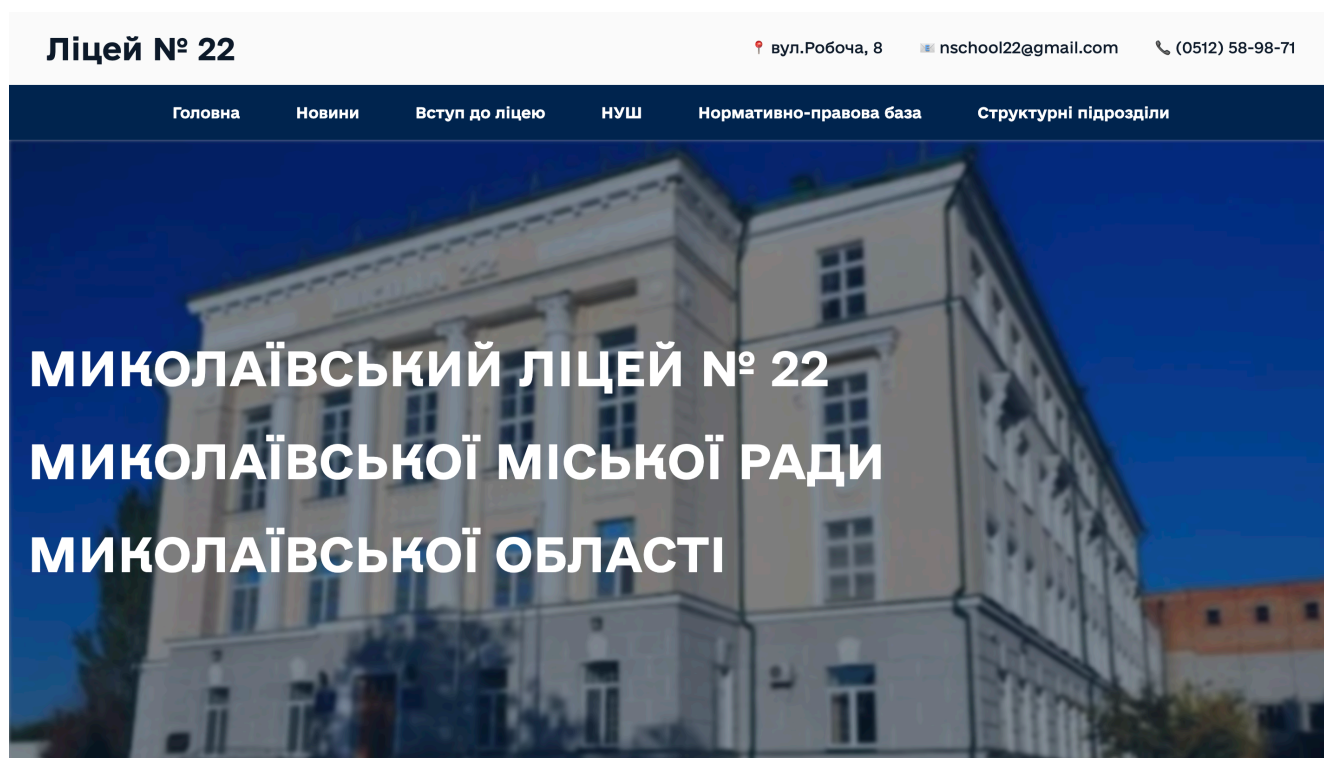


Рисунок 4.3 – Головна сторінка системи

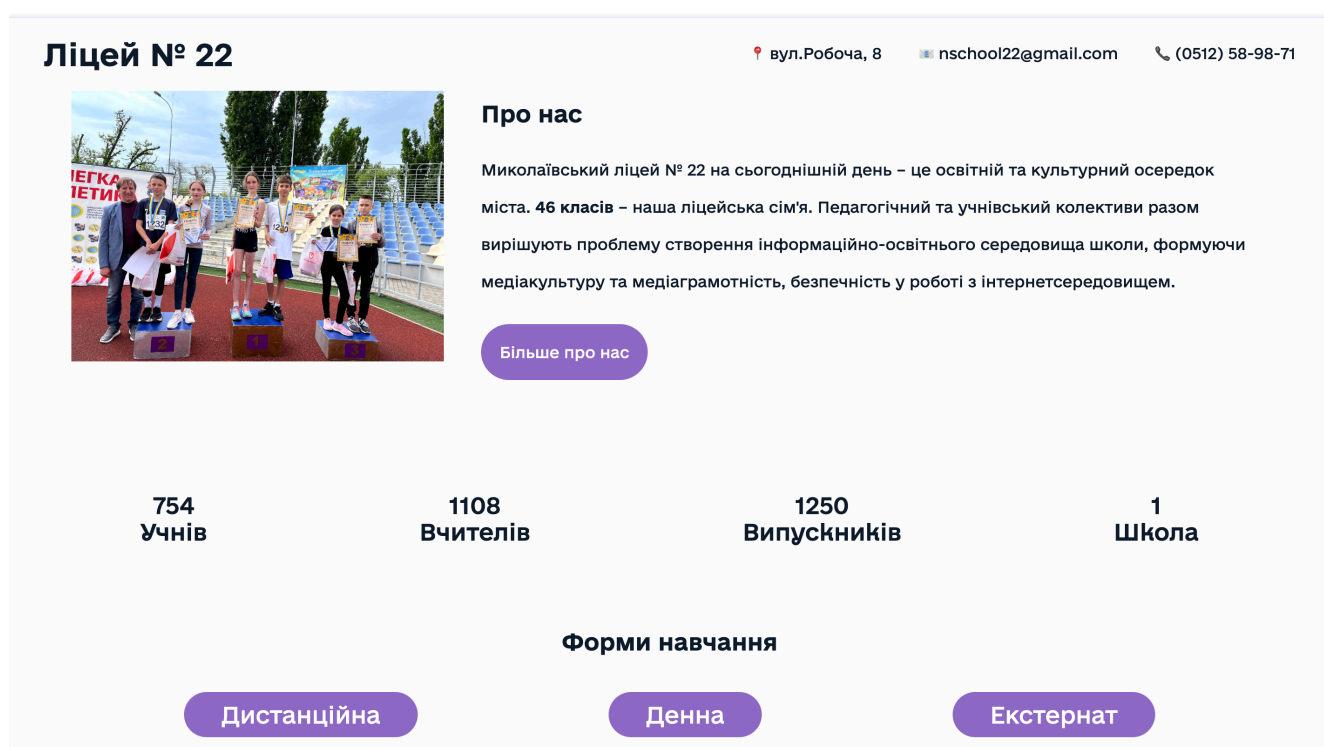


Рисунок 4.4 – Головна сторінка системи

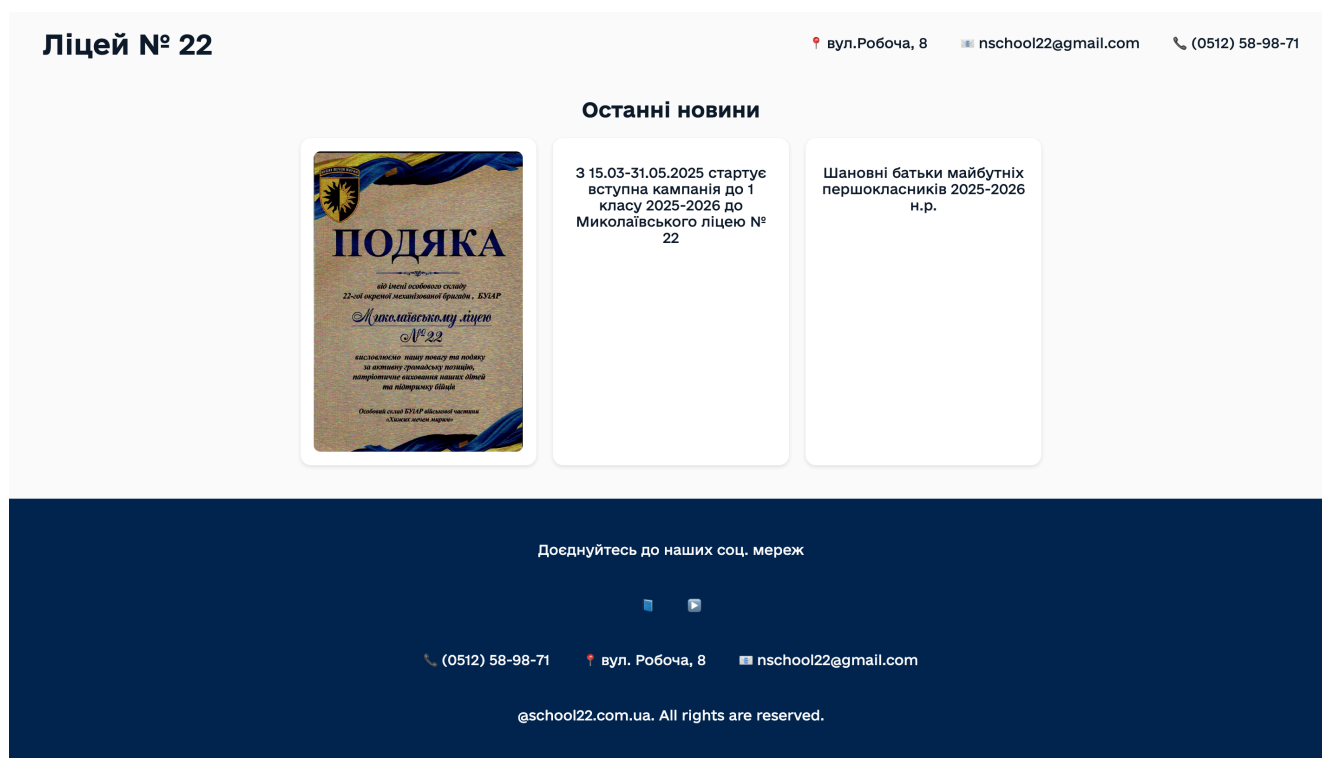


Рисунок 4.5 – Головна сторінка системи

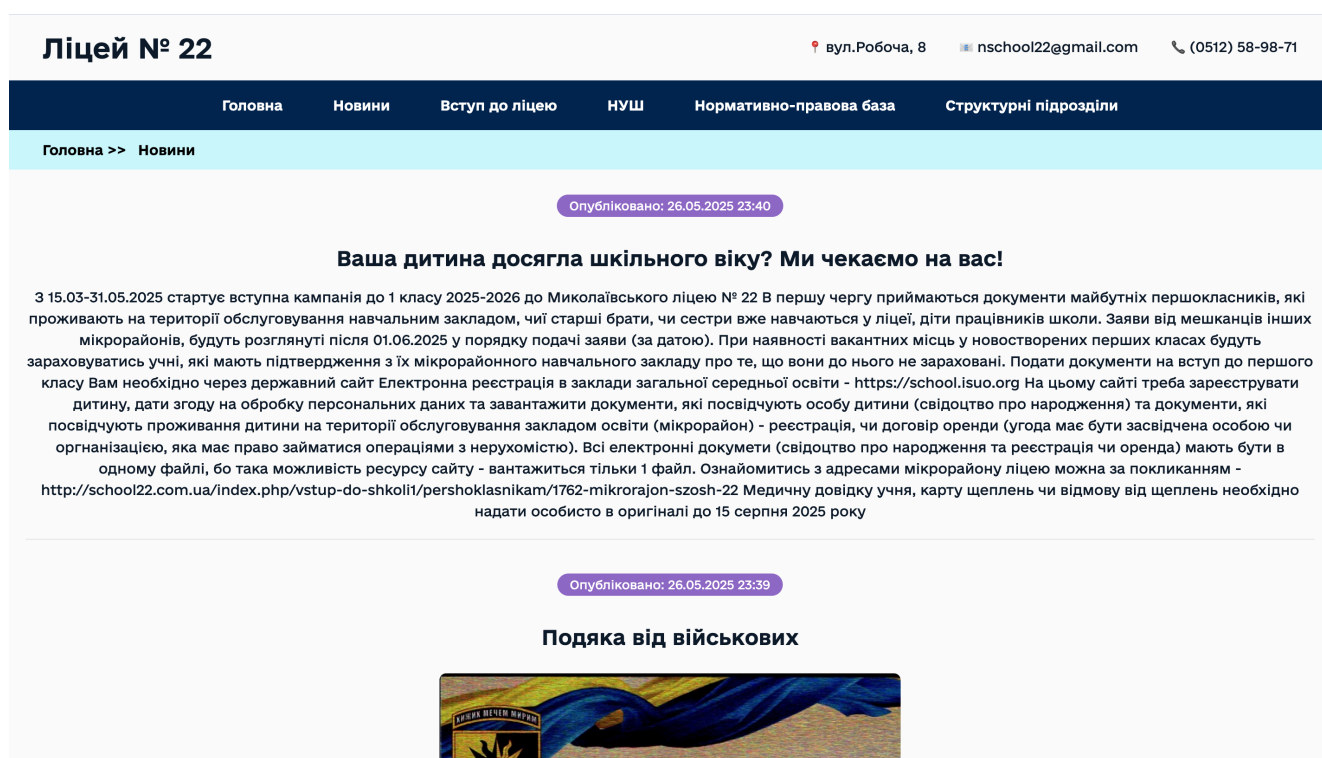


Рисунок 4.6 – Сторінка «Новини»

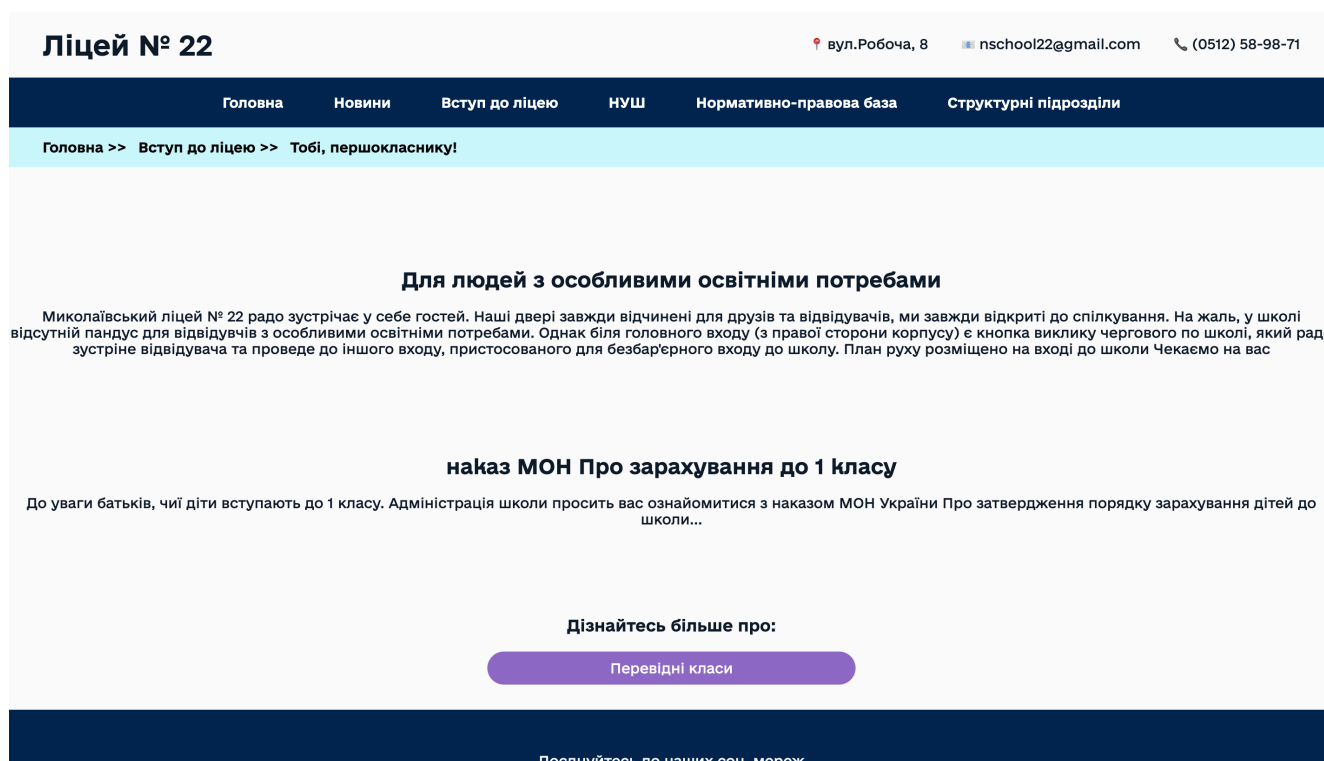


Рисунок 4.7 – Інформаційна сторінка системи

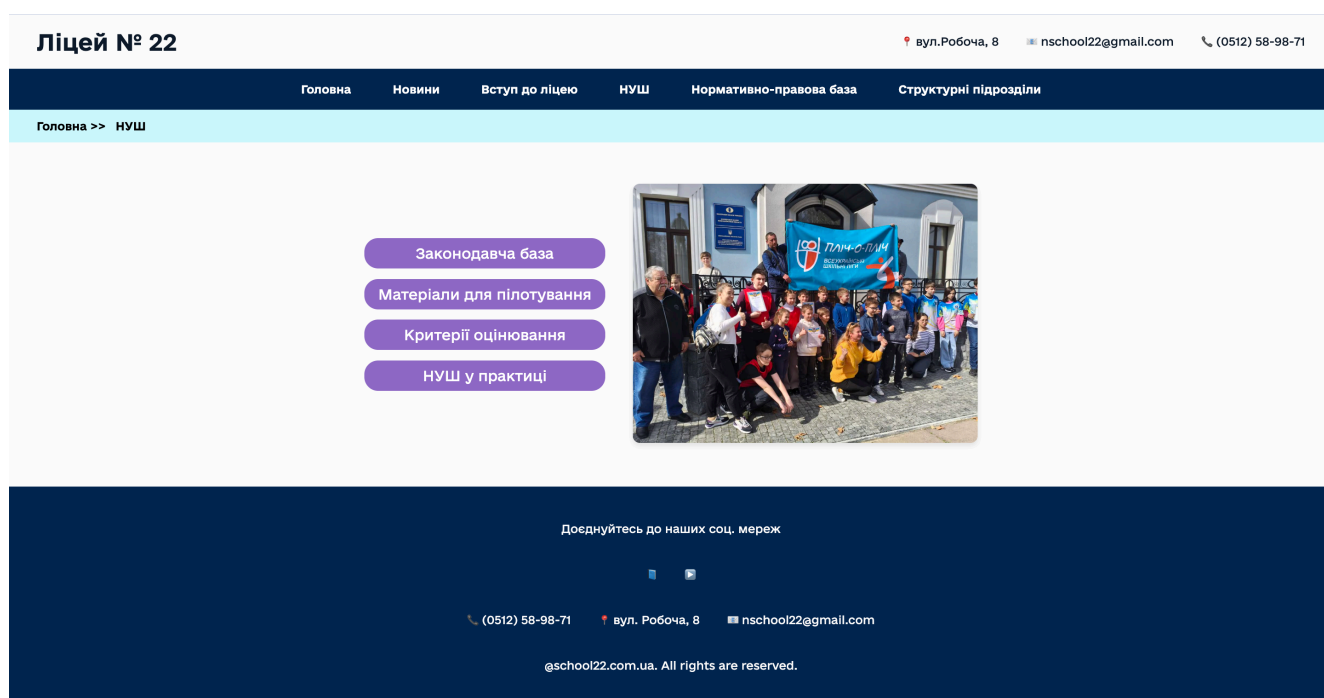


Рисунок 4.8 – Інформаційна сторінка системи

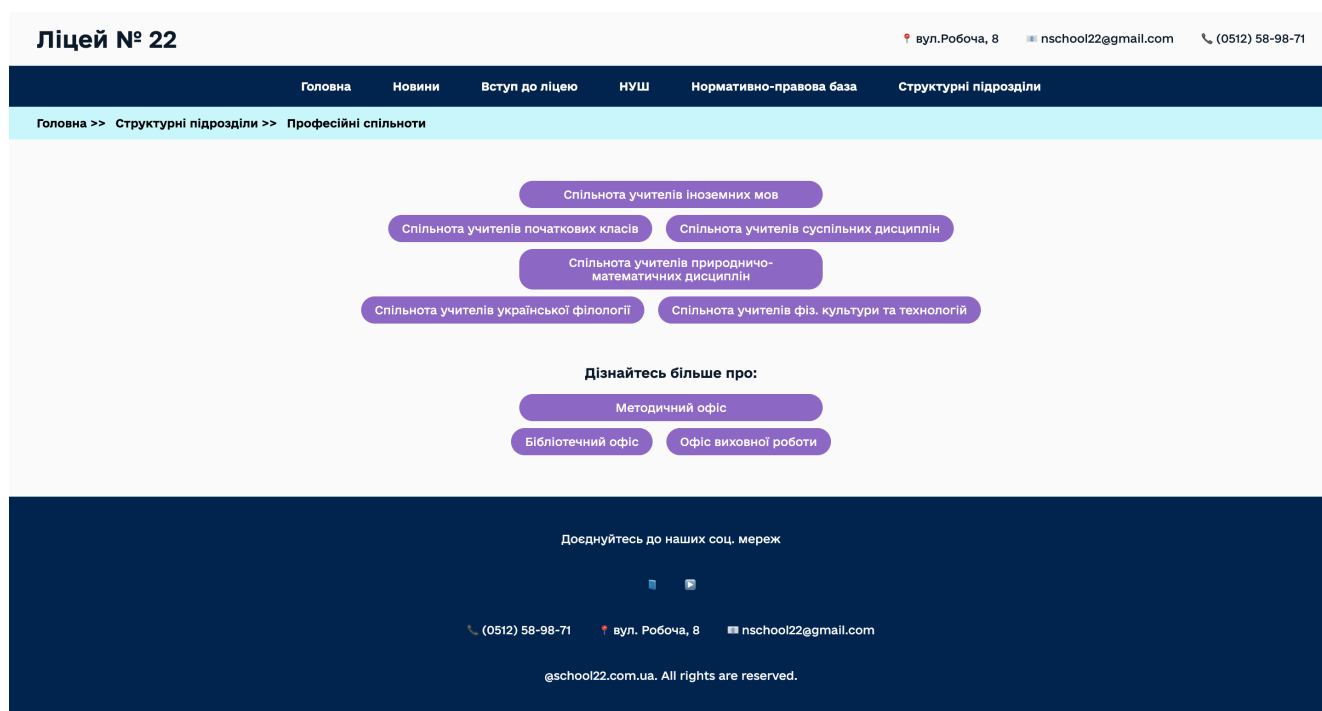


Рисунок 4.9 – Інформаційна сторінка системи

У цьому розділі продемонстровано процес розробки зовнішнього вигляду інформаційної системи, зокрема інтерфейсу, кольорової гами, шрифтів та структури сторінок. Скріншоти ілюструють реалізацію зручного та адаптивного дизайну, що відповідає принципам юзабіліті та враховує специфіку освітнього середовища. Обрані візуальні рішення підкреслюють індивідуальність закладу та сприяють комфортній взаємодії користувачів із системою.

4.2 Розробка користувацької частини інформаційної системи

Користувацький інтерфейс інформаційної системи є ключовим компонентом, оскільки відвідувач сайту взаємодіє з ним. На даному етапі було впроваджено інтерфейс, який забезпечує доступ до основної інформації про навчальний заклад, включаючи розділи «Головна», «Про нас», «Новини», «Вступ до ліцею» тощо.

Для створення структури сторінок було використано HTML та CSS, що дозволило побудувати інтуїтивно зрозумілу та логічну навігацію. Також були реалізовані елементи, які забезпечують адаптивність і коректне відображення на

різних пристроях. У цьому розділі наведені фрагменти HTML-коду, що демонструють основні компоненти реалізації користувацького інтерфейсу.

Головна сторінка системи та деякі другорядні «статичні» (тобто такі, що не редагуються адміністратором) створені у вигляді HTML-шаблонів. Оскільки ці сторінки містять стилі, зображення та інші ресурси, підключені зі статичних файлів, на початку основного коду кожного шаблону додається тег `{% load static %}`.

Цей тег є стандартною командою в шаблонізаторі Django і дозволяє правильно підключати файли зі статичної директорії проєкту.

Лістинг коду приклада.

```
{% load static %}
const images = [
    "{% static 'gallery_1.jpg' %}",
    "{% static 'gallery_2.jpg' %}",
    "{% static 'gallery_3.jpg' %}",
    "{% static 'gallery_4.jpg' %}",
    "{% static 'gallery_5.jpg' %}",
];
```

Таким чином забезпечується централізоване та гнучке підключення ресурсів, незалежно від середовища розгортання (локального чи серверного).

Окремою цікавою частиною головної сторінки є блок з анімацією.

Лістинг коду блоку з анімацією.

```
function animateValue(id, start, end, duration) {
    let current = start;
    const range = end - start;
    const increment = end > start ? 1 : -1;
    const stepTime = Math.abs(Math.floor(duration / range));
    const obj = document.getElementById(id);
    const timer = setInterval(() => {
        current += increment;
        obj.textContent = current;
    }, stepTime);
}
```

```
        if (current == end) {  
            clearInterval(timer);  
        }  
    }, stepTime);  
}  
animateValue("students", 0, 754, 1500);  
animateValue("teachers", 0, 1108, 1000);  
animateValue("graduates", 0, 1250, 2000);  
animateValue("schools", 0, 1, 500);
```

Після завершення розробки головної сторінки, яка виконує функцію вітального екрана для відвідувачів сайту та містить ключову інформацію про навчальний заклад, наступним кроком стала розробка сторінки новин.

Лістинг коду для відображення новин на сайті.

```
<main class="news-section">  
    {% for item in page_obj %}  
    <div class="news-block">  
        <div class="published-date">  
            Опубліковано: {{ item.created_at|localtime|date:"d.m.Y H:i"  
        }}  
    </div>  
    <h2 class="news-title">{{ item.title }}</h2>  
    {% if item.image %}  
    <div class="news-image">  
          
    </div>  
    {% endif %}  
    <div class="news-content">{{ item.content|safe }}</div>  
</div>  
{% empty %}  
<p>Новин поки немає.</p>  
{% endfor %}
```

Цей код забезпечує динамічне формування блоку новин та дозволяє зручно оновлювати його вміст без змін у самому шаблоні.

Оскільки інформаційна система має повторювальний характер, а саме однотипну структуру, вивід інформації користувачу буде мати однаковий код.

Наведений фрагмент HTML-шаблону відповідає за відображення інформаційних блоків, наприклад, у розділі "Вступ до ліцею" або "Нормативно-правова база", де публікується офіційна інформація (положення, правила, вимоги тощо).

Лістинг коду інформаційного блоку.

```
<div class="container">
    {% for info in legal_list %}
    <div class="admission-block">
        <h2>{{ info.title }}</h2>
        {% if info.description %}
        <p>{{ info.description }}</p>
        {% endif %}
    </div>
    {% endfor %}
</div>
```

Структура є простою й адаптованою до повторюваного вмісту, тому адміністратор може додавати нові записи через адмін-панель без потреби змінювати шаблон.

Розроблений користувацький інтерфейс інформаційної системи забезпечує зручний та зрозумілий доступ до основної інформації про навчальний заклад. Використання HTML та CSS дозволило створити структуру сторінок з адаптивним дизайном, що коректно відображається на різних пристроях. Застосування шаблонізатора Django і централізоване підключення статичних ресурсів забезпечують гнучкість і простоту підтримки проекту. Реалізовані функціональні блоки, такі як анімація на головній сторінці, динамічний розділ новин з пагінацією та інформаційні блоки з повторюваною структурою, роблять систему масштабованою та зручною для адміністрування без необхідності змін у коді

2025 р. Бессонова Олена

шаблонів. Створений інтерфейс відповідає вимогам користувачів і технічним задачам проекту.

4.3 Розробка адмін-панелі керування системи

Адміністративна панель є інструментом для керування функціональністю вебсайту та забезпечення доступу до основних елементів системи. У цьому підрозділі розглядається процес розробки адмін-панелі, яка дозволяє адміністраторам сайту редагувати вміст сторінок, додавати інформацію, керувати користувачами та здійснювати налаштування системи. Окремо розглянуто зручність інтерфейсу, безпеку доступу та інтеграцію панелі з базою даних.

Інтерфейс адміністративної панелі розроблений відповідно до стандартного шаблону фреймворку Django. Він характеризується високим рівнем зручності, простотою у використанні та логічною структурою, що є критичним для ефективного адміністрування навіть за відсутності технічних знань.

При створенні адміністративної панелі вебсайту було застосовано базові інструменти фреймворку Django, які забезпечили ефективне управління структурою, логікою та інтерфейсом адміністративної частини системи.

Файл `manage.py` є ключовим інструментом командного рядка для керування Django-проєктом. З його допомогою виконувалися операції зі створення суперкористувача, запуску локального сервера, а також застосування міграцій моделей до бази даних.

Фрагмент лістингу коду `manage.py`.

```
def main():  
    os.environ.setdefault('DJANGO_SETTINGS_MODULE',  
'school_site.settings')  
    try:  
        from django.core.management import execute_from_command_line  
    except ImportError as exc:
```

Файл `admin.py` відповідав за реєстрацію моделей для їх відображення в адміністративній панелі. У ньому налаштовувалися параметри представлення

даних, зокрема поля для сортування, фільтрації, пошуку, а також вказувалися атрибути лише для читання.

Фрагмент лістингу коду admin.py.

```
admin.site.register(News)
admin.site.register(Administrator)
```

У models.py визначалися моделі, які описують структуру таблиць бази даних. Після реєстрації цих моделей у admin.py, вони автоматично ставали доступними для керування в адміністративному інтерфейсі.

Фрагмент лістингу коду models.py.

```
class News(models.Model):
    title = models.CharField("Заголовок", max_length=600)
    content = models.TextField("Контент", blank=True, null=True)
    image = models.ImageField("Зображення", upload_to='news/',
blank=True, null=True)
    created_at = models.DateTimeField("Дата створення",
auto_now_add=True)
    def __str__(self):
        return self.title
```

Файл views.py, який зазвичай відповідає за логіку представлення сторінок користувачам, у деяких випадках використовувався для створення додаткових внутрішніх уявлень або налаштувань, пов'язаних із адмін-панеллю.

Фрагмент лістингу коду models.py.

```
def school_admins_view(request):
    admins = Administrator.objects.all()
    schedule = ReceptionSchedule.objects.all()
    return render(request, "about.html", {
        "admins": admins,
```

У urls.py здійснювалась маршрутизація, зокрема додавання стандартного шляху до адміністративної панелі за адресою /admin, що забезпечувало доступ до інтерфейсу адміністрування.

Фрагмент лістингу коду admin.py.

```
urlpatterns = [  
    path('', home, name='home'),  
    path('news/', news_list, name='news'),
```

Конфігураційний файл settings.py відігравав ключову роль у налаштуванні адміністративної панелі. Зокрема, параметр INSTALLED_APPS містив перелік активованих застосунків, де обов'язковим було включення 'django.contrib.admin'.

У секції MIDDLEWARE визначалися компоненти проміжного програмного забезпечення, які забезпечували автентифікацію, сесійне управління та інші механізми безпеки.

Налаштування TEMPLATES відповідали за обробку шаблонів, у тому числі для кастомізації вигляду панелі.

У блоці DATABASES зазначалися параметри підключення до бази даних, у якій зберігались усі об'єкти системи. В даному проєкті було створено і задіяно одну базу даних «school_db».

Значення змінної TIME_ZONE забезпечували локалізацію інтерфейсу відповідно до часового поясу.

```
MEDIA_ROOT = BASE_DIR / 'media'  
MEDIA_URL = '/media/'  
STATIC_URL = '/static/'  
STATICFILES_DIRS = [BASE_DIR / "static"]  
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'django.conf',  
    'core',  
]
```

Нижче подано приклад вигляду адмін-панелі на рисунках 4.10 – 4.15, створеної за допомогою стандартних засобів фреймворку Django.

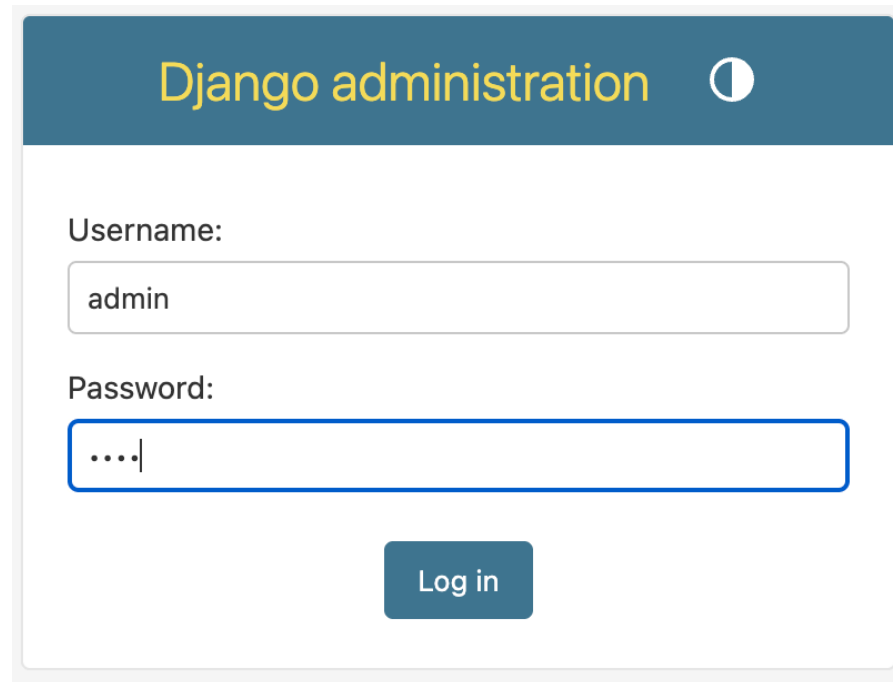
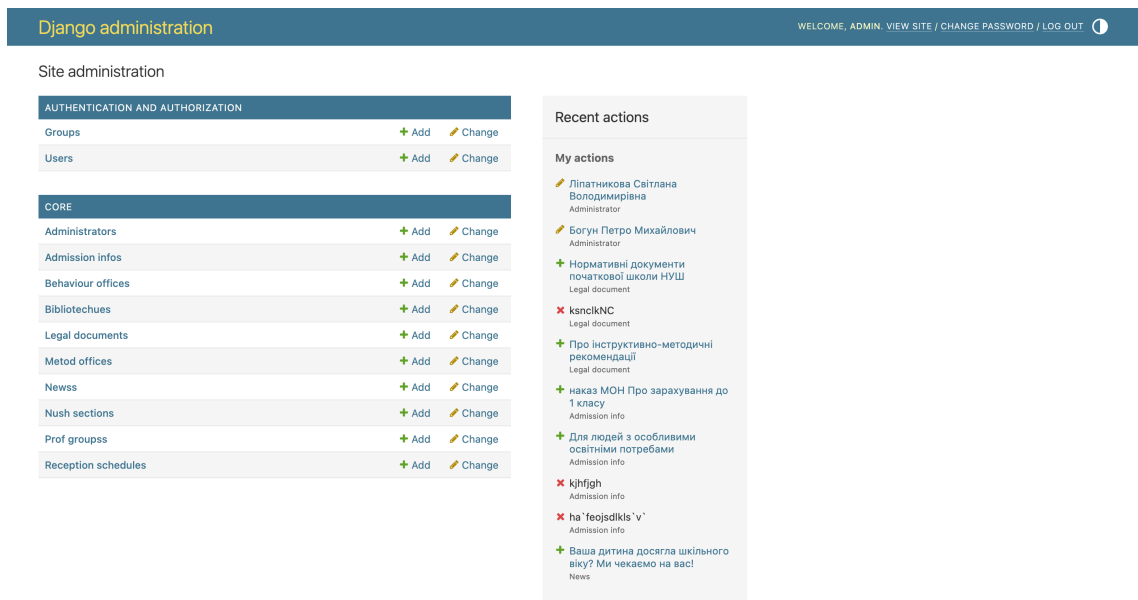


Рисунок 4.10 – Вигляд реєстрації адмін-панелі



AUTHENTICATION AND AUTHORIZATION	
Groups	+ Add Change
Users	+ Add Change

CORE	
Administrators	+ Add Change
Admission infos	+ Add Change
Behaviour offices	+ Add Change
Bibliotechues	+ Add Change
Legal documents	+ Add Change
Metod offices	+ Add Change
Newss	+ Add Change
Nush sections	+ Add Change
Prof groups	+ Add Change
Reception schedules	+ Add Change

Recent actions

My actions

- [Ліпатникова Світлана Володимирівна](#) Administrator
- [Богун Петро Михайлович](#) Administrator
- [Нормативні документи початкової школи НУШ](#) Legal document
- [ksnclikNC](#) Legal document
- [Про інструктивно-методичні рекомендації](#) Legal document
- [наказ МОН Про зарахування до 1 класу](#) Admission info
- [Для людей з особливими освітніми потребами](#) Admission info
- [kjhfgh](#) Admission info
- [ha'feojdrlks'v'](#) Admission info
- [Ваша дитина досягла шкільного віку? Ми чекаємо на вас!](#) News

Рисунок 4.11 – Вигляд головної сторінки адмін-панелі

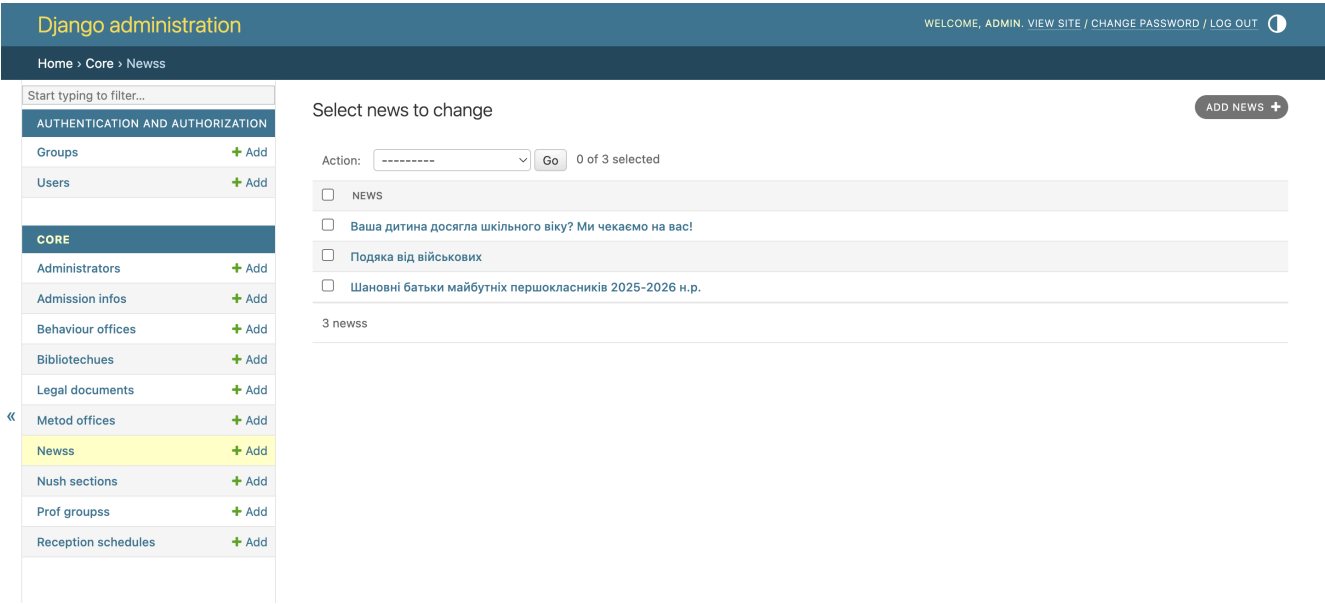


Рисунок 4.12 – Вигляд дочірньої сторінки адмін-панелі

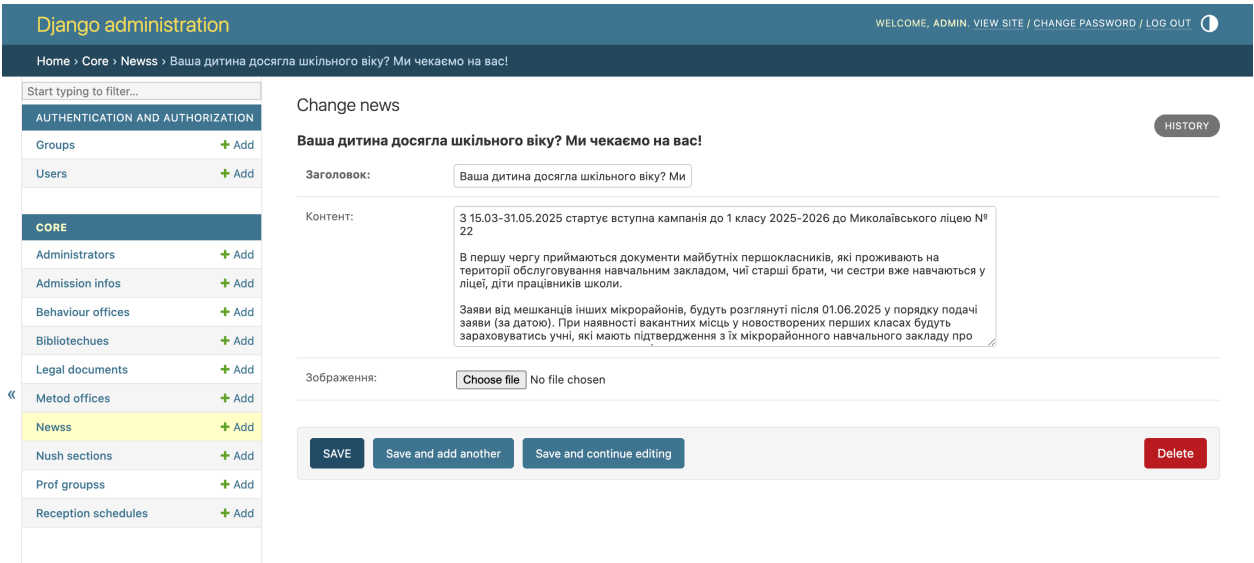


Рисунок 4.13 – Вигляд сторінки для редагування адмін-панелі

Delete

Are you sure you want to delete the news "Ваша дитина досягла шкільного віку? Ми чекаємо на вас!"? All of the following related items will be deleted:

Summary

- Newss: 1

Objects

- News: Ваша дитина досягла шкільного віку? Ми чекаємо на вас!

Рисунок 4.14 – Вигляд сторінки для видалення адмін-панелі

The screenshot shows the Django admin interface for adding a new news item. On the left is a sidebar with a search bar and a list of models. The 'Newss' model is highlighted. The main area is titled 'Add news' and contains a 'Заголовок:' (Title) text input, a 'Контент:' (Content) text area, and a 'Зображення:' (Image) section with a 'Choose file' button. At the bottom, there are three buttons: 'SAVE', 'Save and add another', and 'Save and continue editing'.

Рисунок 4.15 – Вигляд сторінки для додавання адмін-панелі

Таким чином, використання стандартної адмін-панелі Django значно пришвидшило розробку системи керування вмістом, забезпечивши гнучкість та розширюваність, необхідну для подальшого масштабування та підтримки сайту.

4.4 Тестування інформаційної системи

Після завершення етапу розробки інформаційної системи було проведено її комплексне тестування. Метою тестування було виявлення можливих помилок, перевірка відповідності реалізованої функціональності вимогам технічного завдання, оцінка зручності інтерфейсу користувача, перевірка стабільності роботи системи в умовах різних сценаріїв використання, а також забезпечення базового рівня інформаційної безпеки.

Було здійснено функціональне тестування всіх ключових модулів, зокрема перевірено коректність відображення вебсторінок, стабільність переходів між розділами, правильність обробки запитів, надійність авторизації адміністратора та точність виконання CRUD-операцій (створення, читання, оновлення, видалення інформації). Адміністративна панель була перевірена на предмет відображення моделей бази даних, працездатності інструментів сортування, пошуку та фільтрації.

Інтерфейс користувача було протестовано з урахуванням принципів юзабіліті: зручність навігації, логічність структури сайту, доступність функцій для різних типів користувачів, а також візуальна привабливість та читабельність текстових елементів. Було перевірено відповідність кольорової гами та шрифтів базовим принципам дизайну освітніх ресурсів.

З метою підвищення захищеності інформаційної системи було здійснено первинне тестування безпеки, яке включало перевірку доступу до адміністративного інтерфейсу, перевірку обробки введених даних з метою виявлення потенційних вразливостей, зокрема SQL-ін'єкцій та XSS-атак. Також було здійснено перевірку на наявність можливості несанкціонованого доступу до критично важливих даних.

Крім того, проводилося кросбраузерне тестування, яке передбачало перевірку функціональності системи у найбільш популярних веббраузерах, таких як Google Chrome, Mozilla Firefox, Microsoft Edge та Safari. Це дозволило переконатися у стабільній та однаковій роботі інтерфейсу в різних середовищах. Також було проведено тестування адаптивності для пристроїв із різними розмірами екранів, зокрема смартфонів, планшетів і настільних комп'ютерів.

Для додаткового незалежного аналізу коректності роботи сайту можуть бути використані сторонні сервіси, зокрема:

- Google LightHouse для перевірки продуктивності, доступності та SEO-параметрів сайту;
- W3C Markup Validation Service для аналізу правильності HTML-розмітки;
- GTmetrix та PageSpeed Insights для оцінки швидкодії та рекомендацій з оптимізації;
- BrowserStack для кросбраузерного тестування в хмарному середовищі;
- Security Headers для аналізу налаштувань безпеки HTTP-заголовків.

У результаті тестування були виявлені окремі незначні недоліки, які було оперативно виправлено. Після усунення усіх виявлених зауважень система визнана готовою до розгортання у мережі Інтернет.

На наступному етапі систему буде розгорнуто у відкритому доступі. Для цього буде обрано відповідний хостинг-провайдер, що підтримує фреймворк Django та роботу з базою даних MySQL. Буде зареєстровано доменне ім'я, сконфігуровано середовище виконання проєкту, перенесено файли на сервер, а також виконано необхідні міграції для створення структури бази даних у віддаленому середовищі. Додатково буде налаштовано вебсервер (наприклад, Apache або Nginx) з використанням WSGI-інтерфейсу (наприклад, Gunicorn або uWSGI), встановлено SSL-сертифікат для забезпечення захищеного з'єднання за протоколом HTTPS, а також реалізовано базові механізми моніторингу працездатності сервера.

Після завершення всіх підготовчих етапів систему буде опубліковано у глобальній мережі, що забезпечить доступ до інформації широкому колу користувачів, спростить адміністративні процеси закладу освіти та сприятиме цифровій трансформації його інформаційного середовища.

Висновки до 4 розділу

У четвертому розділі було здійснено повну програмну реалізацію інформаційної системи для навчального закладу із використанням сучасних засобів веб-розробки на базі фреймворку Django. Зокрема, у підрозділі 4.1 було детально опрацьовано дизайн системи, де визначено логічну архітектуру, структуру взаємодії компонентів, навігаційні шляхи та зовнішній вигляд інтерфейсів. Особлива увага приділялася принципам юзабіліті, адаптивності інтерфейсу та відповідності потребам кінцевих користувачів.

У підрозділі 4.2 реалізовано користувацьку частину системи, яка забезпечує доступ до публічної інформації, зокрема: новин, інформації про викладачів, вступної інформації, підрозділів закладу, нормативно-правових документів тощо.

Для цього були реалізовані відповідні представлення (views), шаблони (templates) та маршрутизація (urls), що забезпечують ефективне відображення й обробку даних з використанням шаблонізатора Django.

Підрозділ 4.3 присвячено розробці адміністративної панелі системи, яка створена на основі вбудованого механізму Django admin. Адмін-панель було адаптовано під специфіку предметної області – додано необхідні поля, фільтри, пошук, а також забезпечено зручний механізм керування вмістом для уповноважених користувачів. Таким чином, адміністративна частина системи забезпечує повний контроль над наповненням ресурсу без необхідності втручання в програмний код.

У підрозділі 4.4 описано процес тестування інформаційної системи, яке включало перевірку функціональної повноти, відповідності реалізованого функціоналу технічним вимогам, а також тестування на надійність та стабільність роботи. Під час тестування були виявлені та усунені окремі помилки інтерфейсу й логіки, що дозволило досягти високої якості реалізації системи.

Таким чином, у результаті виконаних робіт було реалізовано повнофункціональну веборієнтовану інформаційну систему навчального закладу, яка забезпечує автоматизацію основних інформаційних процесів, є масштабованою, зручною в користуванні та готовою до практичного впровадження.

ВИСНОВКИ

У процесі розробки інформаційної системи підтримки освітнього процесу для закладу середньої освіти було досягнуто головної мети – створення інформаційної системи, яка забезпечує ефективну підтримку управлінських та навчальних процесів. Реалізована система надає централізований доступ до важливої інформації про діяльність навчального закладу, зокрема про склад педагогічного колективу, навчальні програми, розклад занять, результати навчання тощо.

Система дозволяє автоматизувати значну частину рутинних завдань, знизити навантаження на адміністративний персонал і забезпечити прозору взаємодію між учнями, вчителями та адміністрацією. Завдяки цьому навчальний процес стає більш адаптованим до сучасних вимог, інтерактивним і зручним для всіх учасників освітнього середовища.

Під час виконання кваліфікаційної роботи було виконано такі завдання:

- проаналізовано сучасні підходи до створення освітніх інформаційних систем;
- визначено функціональні вимоги до системи;
- спроектовано інформаційну базу школи;
- розроблено архітектуру вебзастосунку;
- реалізовано ключові модулі системи;
- проведено тестування.

Практичне значення розробки полягає у впровадженні інноваційного інструменту, що покращує організацію та ефективність освітнього процесу, сприяючи цифровій трансформації середньої освіти.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Київська гімназія №154 – Офіційний сайт. *Київська гімназія №154 – Офіційний сайт*. URL: <https://gymnasium154.com.ua/> (дата звернення: 02.05.2025).
2. Комунальний заклад "Великопетиський опорний ліцей" Великопетиської селищної ради Каховського району Херсонської області – Великопетиський опорний ліцей. *Комунальний заклад "Великопетиський опорний ліцей" Великопетиської селищної ради Каховського району Херсонської області – Великопетиський опорний ліцей*. URL: <https://vlep1.edu.ks.ua/> (дата звернення: 02.05.2025).
3. Національна система освіти - Національний Реєстраційний Орган | ВРЮУ. *Національний Реєстраційний Орган | ВРЮУ*. URL: <https://ukrlawcouncil.org/nacziionalna-systema-osvity/> (дата звернення: 02.05.2025).
4. Школа 102. *102 ШКОЛА*. URL: <https://102.org.ua/> (дата звернення: 02.05.2025).
5. Що має, а чого не має бути на сайті школи: юридичне роз'яснення для батьків. *Нова українська школа | Вебресурс НУШ*. URL: <https://nus.org.ua/questions/shho-maye-a-chogo-ne-maye-buty-na-sajti-shkoly-yurydychne-roz-yasnennya-dlya-batkiv/> (дата звернення: 02.05.2025).
6. Best Databases for Python in 2024 - A Comprehensive Comparison - Ropstam Solutions Inc. *Ropstam Solutions Inc.* URL: <https://www.ropstam.com/best-databases-for-python/> (date of access: 02.05.2025).
7. Die Schweitzer | Albert-Schweitzer-Gymnasium. *Die Schweitzer | Albert-Schweitzer-Gymnasium*. URL: <https://www.die-schweitzer.de/hp/> (дата звернення: 02.05.2025).
8. Diesterweg-Gymnasium Berlin. *Diesterweg-Gymnasium Berlin*. URL: <https://www.dwgym.berlin> (дата звернення: 02.05.2025).
9. Digital Transformation in Primary and Secondary Education in Bulgaria / E. Gourova та ін. *Lecture Notes of the Institute for Computer Sciences, Social Informatics*

and Telecommunications Engineering. Cham, 2025. С. 263–275. URL: https://doi.org/10.1007/978-3-031-84312-9_18 (дата звернення: 18.05.2025).

10. Dreilinden-Gymnasium Berlin Wannsee. *Dreilinden-Gymnasium*. URL: <https://dreilinden-gymnasium.de> (дата звернення: 02.05.2025).

11. Friedrich-Ebert-Gymnasium Berlin – FEO. *Friedrich-Ebert-Gymnasium Berlin – FEO*. URL: <https://feoberlin.de> (дата звернення: 02.05.2025).

12. Gagniuc P. A. Python Specific. *Coding Examples from Simple to Complex*. Cham, 2024. С. 223–233. URL: https://doi.org/10.1007/978-3-031-53812-4_13 (дата звернення: 18.05.2025).

13. GeeksforGeeks. Differences Between Django vs Flask - GeeksforGeeks. *GeeksforGeeks*. URL: <https://www.geeksforgeeks.org/differences-between-django-vs-flask/> (дата звернення: 02.05.2025).

14. German International School Toronto. *German International School Toronto*. URL: <https://www.germaninternationalschooltoronto.ca/> (дата звернення: 02.05.2025).

15. Gor V. Animations and Transitions in CSS3. *Creating Responsive Websites Using HTML5 and CSS3*. Berkeley, CA, 2023. С. 151–172. URL: https://doi.org/10.1007/978-1-4842-9783-4_6 (дата звернення: 18.05.2025).

16. Gor V. HTML5 and Responsive Web Design. *Creating Responsive Websites Using HTML5 and CSS3*. Berkeley, CA, 2023. С. 23–60. URL: https://doi.org/10.1007/978-1-4842-9783-4_2 (дата звернення: 18.05.2025).

17. Gor V. Introduction to Web Development. *Creating Responsive Websites Using HTML5 and CSS3*. Berkeley, CA, 2023. С. 1–22. URL: https://doi.org/10.1007/978-1-4842-9783-4_1 (дата звернення: 22.05.2025).

18. Gourova E., Antonova A. A School to Remember: Analyzing School Best Practices in the Perspective of Digital Transformation of Schools. *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*. Cham, 2023. С. 360–371. URL: https://doi.org/10.1007/978-3-031-44668-9_28 (дата звернення: 22.05.2025).

19. Harwani B. Using CSS. *jQuery Recipes*. Berkeley, CA, 2021. С. 577–651. URL: https://doi.org/10.1007/978-1-4842-7304-3_12 (дата звернення: 22.05.2025).
20. Iyer S. V. Introduction to Python. *Simulating Nonlinear Circuits with Python Power Electronics*. Cham, 2018. С. 13–33. URL: https://doi.org/10.1007/978-3-319-73984-7_2 (дата звернення: 22.05.2025).
21. Kant-Gymnasium Berlin-Spandau. *Kant-Gymnasium Berlin-Spandau*. URL: <https://www.kant-gymnasium-berlin.de> (дата звернення: 02.05.2025).
22. Keystone School Keystone School | IB Candidate School in Toronto | Grades 7-12. *Keystone School*. URL: <https://keystoneschools.ca/> (дата звернення: 02.05.2025).
23. Lilleaas A. Building Traditional Web Apps with HTML and CSS. *Pro Kotlin Web Apps from Scratch*. Berkeley, CA, 2023. С. 161–195. URL: https://doi.org/10.1007/978-1-4842-9057-6_9 (дата звернення: 25.05.2025).
24. Python vs PHP: Which One to Choose for Web Development in 2025?. *MindInventory*. URL: <https://www.mindinventory.com/blog/php-vs-python/#:~:text=Python%20programs%20are%20shorter%20and,PHP%20as%20compared%20to%20Python.> (дата звернення: 02.05.2025).
25. Schmidt M., Huang R. Defining Learning Experience Design: Voices from the Field of Learning Design & Technology. *TechTrends*. 2021. URL: <https://doi.org/10.1007/s11528-021-00656-y> (дата звернення: 25.05.2025).
26. Tang H.-H. End-to-End Testing. *Practical Front-End Testing*. Berkeley, CA, 2025. С. 151–170. URL: https://doi.org/10.1007/979-8-8688-1451-8_4 (дата звернення: 25.05.2025).
27. The Django admin site | Django documentation. *Django Project*. URL: <https://docs.djangoproject.com/en/5.2/ref/contrib/admin/> (дата звернення: 19.05.2025).
28. UIS – Educate with passion, Care with love. *UIS – Educate with passion, Care with love*. URL: <https://www.uihs.ca> (дата звернення: 02.05.2025).
29. Upadhyaya N. Testing Your Application. *Advanced Front-End Development*. Berkeley, CA, 2025. С. 265–279. URL: https://doi.org/10.1007/979-8-8688-1318-4_14 (дата звернення: 25.05.2025).

30. What is Figma – Uses, Benefits, and Features | AND Academy. *AND Academy*. URL: <https://www.andacademy.com/resources/blog/graphic-design/what-is-figma/> (дата звернення: 02.05.2025).

31. Which Is the Best Python Web Framework: Django, Flask, or FastAPI? | The PyCharm Blog. *The JetBrains Blog*. URL: <https://blog.jetbrains.com/pycharm/2025/02/django-flask-fastapi/> (дата звернення: 02.05.2025).

ДОДАТОК А

settings.py

```
from pathlib import Path

BASE_DIR = Path(__file__).resolve().parent.parent

SECRET_KEY = 'your-secret-key'
DEBUG = True
ALLOWED_HOSTS = []

INSTALLED_APPS = [
    'core',
]
...
MEDIA_ROOT = BASE_DIR / 'media'
MEDIA_URL = '/media/'
STATIC_URL = '/static/'
STATICFILES_DIRS = [BASE_DIR / "static"]
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'django.conf',
    'core',
]
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'school_db',
        'USER': 'root',
        'PASSWORD': 'root',
        'HOST': 'localhost',
        'PORT': '3306',
        'OPTIONS': {
            'init_command': "SET sql_mode='STRICT_TRANS_TABLES'",
        },
    }
}
```

```
}  
TEMPLATES = [  
    {  
        'BACKEND': 'django.template.backends.django.DjangoTemplates',  
        'DIRS': [BASE_DIR / 'templates'],  
        'APP_DIRS': True,  
        'OPTIONS': {  
            'context_processors': [  
                'django.template.context_processors.debug',  
                'django.template.context_processors.request',  
                'django.contrib.auth.context_processors.auth',  
                'django.contrib.messages.context_processors.messages',  
            ],  
        },  
    },  
]  
MIDDLEWARE = [  
    'django.middleware.security.SecurityMiddleware',  
    'django.contrib.sessions.middleware.SessionMiddleware',  
    'django.middleware.common.CommonMiddleware',  
    'django.middleware.csrf.CsrfViewMiddleware',  
    'django.contrib.auth.middleware.AuthenticationMiddleware',  
    'django.contrib.messages.middleware.MessageMiddleware',  
    'django.middleware.clickjacking.XFrameOptionsMiddleware',  
]  
DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'  
STATIC_URL = '/static/'  
ROOT_URLCONF = 'school_site.urls'  
TIME_ZONE = 'Europe/Kyiv'  
USE_TZ = True
```


ДОДАТОК Б

urls.py

```
from django.urls import path
from .views import home, news_list,
school_admins_view, admission_info, admission_info_1, admission_info_2
from .views import nush_section, nush_section_1, nush_section_2, nush_section_3,
nush_section_4, legal_document, over_structure_info
from .views import
over_structure_info_1, over_structure_info_2, over_structure_info_3, over_structure_info_4
from .views import prof_groups_info, prof_groups_info_1, prof_groups_info_2,
prof_groups_info_3, prof_groups_info_4, prof_groups_info_5

urlpatterns = [
    path('', home, name='home'),
    path('news/', news_list, name='news'),
    path("about/", school_admins_view, name="about"),
    path("admission/", admission_info, name="enroll"),
    path("admission/enroll/", admission_info_1, name="enroll_1"),
    path("admission/enroll_2/", admission_info_2, name="enroll_2"),
    path('nush/', nush_section, name="nush"),
    path('nush/zakon/', nush_section_1, name="zakon"),
    path('nush/pilot/', nush_section_2, name="pilot"),
    path('nush/grading/', nush_section_3, name="grading"),
    path('nush/practice/', nush_section_4, name="practice_nush"),
    path("zakon_base/", legal_document, name="zakon_base"),
    path("structure/", over_structure_info, name="structure"),
    path("structure/profgroups/", over_structure_info_1, name="profgroups"),
    path("structure/metodoffice/", over_structure_info_2, name="metodoffice"),
    path("structure/bibl/", over_structure_info_3, name="biblio_office"),
    path("structure/behavoffice/", over_structure_info_4, name="behavoffice"),
    path("profgroups/languages/", prof_groups_info, name="languages"),
    path("profgroups/firstclass/", prof_groups_info_1, name="firstclass"),
    path("profgroups/languages/", prof_groups_info_2, name="natures"),
    path("profgroups/languages/", prof_groups_info_3, name="mathematics"),
    path("profgroups/languages/", prof_groups_info_4, name="ukrainian"),
    path("profgroups/languages/", prof_groups_info_5, name="sport"),
]
```

ДОДАТОК В

views.py

```
from django.shortcuts import render
from .models import News
from .models import Administrator, ReceptionSchedule
from .models import AdmissionInfo
from .models import NUSHSection
from .models import OverStructure
from .models import LegalDocument
from .models import ProfGroups
from django.core.paginator import Paginator

def home(request):
    return render(request, 'index.html')

def news_list(request):
    news = News.objects.order_by('-created_at')
    paginator = Paginator(news, 6) # 6 новин на сторінку

    page_number = request.GET.get('page')
    page_obj = paginator.get_page(page_number)
    return render(request, 'news.html', {'page_obj': page_obj})

def school_admins_view(request):
    admins = Administrator.objects.all()
    schedule = ReceptionSchedule.objects.all()
    return render(request, "about.html", {
        "admins": admins,
        "schedule": schedule
    })

def admission_info(request):
    return render(request, 'enroll.html')

def admission_info_1(request):
    info_list = AdmissionInfo.objects.filter(category='recommendation')
    return render(request, 'enroll_1.html', {'info_list': info_list})

def admission_info_2(request):
    info_list = AdmissionInfo.objects.filter(category='classes')
    return render(request, 'enroll_2.html', {'info_list': info_list})

def nush_section(request):
    return render(request, 'nush.html')
```

```
def nush_section_1(request):
    nush_list = NUSHSection.objects.filter(category='legal_base')
    return render(request, 'zakon.html', {'nush_list': nush_list})
def nush_section_2(request):
    nush_list = NUSHSection.objects.filter(category='materials')
    return render(request, 'pilot.html', {'nush_list': nush_list})
def nush_section_3(request):
    nush_list = NUSHSection.objects.filter(category='grades')
    return render(request, 'grading.html', {'nush_list': nush_list})
def nush_section_4(request):
    nush_list = NUSHSection.objects.filter(category='practice')
    return render(request, 'practice_nush.html', {'nush_list': nush_list})
def legal_document(request):
    legal_list = LegalDocument.objects.all()
    return render(request, 'zakon_base.html', {'legal_list': legal_list})
def over_structure_info(request):
    return render(request, 'structure.html')
def over_structure_info_1(request):
    structure_list = OverStructure.objects.filter(category='prof_groups')
    return render(request, 'profgroups.html', {'structure_list': structure_list})
def over_structure_info_2(request):
    structure_list = OverStructure.objects.filter(category='metod_office')
    return render(request, 'metodoffice.html', {'structure_list': structure_list})
def over_structure_info_3(request):
    structure_list = OverStructure.objects.filter(category='bibl')
    return render(request, 'biblio_office.html', {'structure_list': structure_list})
def over_structure_info_4(request):
    structure_list = OverStructure.objects.filter(category='behaviour')
    return render(request, 'behavoffice.html', {'structure_list': structure_list})
def prof_groups_info(request):
    groups_list = ProfGroups.objects.filter(category='languages')
    return render(request, 'languages.html', {'groups_list': groups_list})
def prof_groups_info_1(request):
    groups_list = ProfGroups.objects.filter(category='first_classes')
    return render(request, 'firstclass.html', {'groups_list': groups_list})
def prof_groups_info_2(request):
    groups_list = ProfGroups.objects.filter(category='natures')
    return render(request, 'natures.html', {'groups_list': groups_list})
def prof_groups_info_3(request):
    groups_list = ProfGroups.objects.filter(category='mathematics')
    return render(request, 'mathematics.html', {'groups_list': groups_list})
```

```
def prof_groups_info_4(request):  
    groups_list = ProfGroups.objects.filter(category='ukrainian')  
    return render(request, 'ukrainian.html', {'groups_list': groups_list})  
def prof_groups_info_5(request):  
    groups_list = ProfGroups.objects.filter(category='sport')  
    return render(request, 'sport.html', {'groups_list': groups_list})
```

ДОДАТОК Г

models.py

```
from django.db import models

# Новини
class News(models.Model):
    title = models.CharField("Заголовок", max_length=600)
    content = models.TextField("Контент", blank=True, null=True)
    image = models.ImageField("Зображення", upload_to='news/', blank=True, null=True)
    created_at = models.DateTimeField("Дата створення", auto_now_add=True)

    def __str__(self):
        return self.title

# Про нас
class Administrator(models.Model):
    full_name = models.CharField("ПІБ", max_length=100)
    position = models.CharField("Посада", max_length=200)
    phone = models.CharField("Телефон", max_length=20, blank=True)
    photo = models.ImageField("Фото", upload_to='administration_photos/')

    def __str__(self):
        return self.full_name

class ReceptionSchedule(models.Model):
    day = models.CharField("День", max_length=20)
    time = models.CharField("Час", max_length=20)
    person = models.CharField("Особа", max_length=300)
    title = models.CharField("Посада", max_length=300, blank=True, null=True)
    email = models.EmailField("Емейл", max_length=300, default='example@example.com')

    def __str__(self):
        return f"{self.day} - {self.person}"

# Вступ
class AdmissionInfo(models.Model):
    CATEGORY_CHOICES = [
```

```
( 'recommendation', 'Тобі, першокласнику!'),
( 'classes', 'Перелік класів'),
]

title = models.CharField("Назва", max_length=200)
description = models.TextField("Опис", blank=True, null=True)
category = models.CharField("Категорія", max_length=30, choices=CATEGORY_CHOICES)
content = models.TextField("Контент", blank=True, null=True)

def __str__(self):
    return self.title

# НУШ (Нова українська школа)
class NUSHSection(models.Model):
    CATEGORY_CHOICES = [
        ('legal_base', 'Законодавча база'),
        ('materials', 'Матеріали для пілотування'),
        ('grades', 'Критерії оцінювання'),
        ('practice', 'НУШ у практиці'),
    ]

    title = models.CharField("Назва", max_length=200)
    content = models.TextField("Контент", blank=True, null=True)
    category = models.CharField("Категорія", max_length=30, choices=CATEGORY_CHOICES)

    def __str__(self):
        return self.title

# Нормативно-правова база
class LegalDocument(models.Model):
    title = models.CharField("Назва", max_length=200)
    description = models.TextField("Опис", blank=True)

    def __str__(self):
        return self.title

class OverStructure(models.Model):
    CATEGORY_CHOICES = [
        ('prof_groups', 'Професійні спільноти'),
```

```
        ('metod_office', 'Методичний офіс'),
        ('bibl', 'Бібліотечний офіс'),
        ('behaviour', 'Офіс виховної роботи'),
    ]

    title = models.CharField("Назва", max_length=200)
    content = models.TextField("Контент", blank=True, null=True)
    category = models.CharField("Категорія", max_length=30, choices=CATEGORY_CHOICES)

    def __str__(self):
        return self.title

# Професійні спільноти
class ProfGroups(models.Model):
    CATEGORY_CHOICES = [
        ('languages', 'Іноземна мова'),
        ('first_classes', 'Початкові класи'),
        ('natures', 'Суспільні дисципліни'),
        ('mathematics', 'Природно-математичні'),
        ('ukrainian', 'Укр. філологія'),
        ('sport', 'Фіз. культура та техн.'),
    ]

    name = models.CharField("Ім'я", max_length=200)
    photo = models.ImageField("Фото", upload_to='teachers/', blank=True, null=True)
    bio = models.TextField("Дані", blank=True, null=True)
    category = models.CharField("Категорія", max_length=30, choices=CATEGORY_CHOICES)

    def __str__(self):
        return self.title

# Методичний офіс
class MetodOffice(models.Model):
    CATEGORY_CHOICES = [
        ('metod_room', 'Методичний кабінет'),
        ('structure', 'Структура'),
        ('advisor', 'Методичний порадник'),
        ('head', 'Керівнику кафедри'),
        ('atestation', 'Атестація'),
        ('journals', 'Електронні журнали'),
        ('study_programs', 'Навчальні програми'),
```

```
]

title = models.CharField("Назва", max_length=200)
content = models.TextField("Контент", blank=True, null=True)
category = models.CharField("Категорія", max_length=30, choices=CATEGORY_CHOICES)

def __str__(self):
    return self.title

# Офіс виховної роботи
class BehaviourOffice(models.Model):
    CATEGORY_CHOICES = [
        ('class_head', 'Стор. кл. керівника'),
        ('samovrad', 'Самоврядування'),
        ('parents_page', 'Батківська сторінка'),
        ('suprovid', 'Супровід'),
        ('euroclub', 'Євроклуб'),
        ('reiting', 'Рейтинг'),
    ]

    title = models.CharField("Назва", max_length=200)
    content = models.TextField("Контент", blank=True, null=True)
    category = models.CharField("Категорія", max_length=30, choices=CATEGORY_CHOICES)

    def __str__(self):
        return self.title

# Бібліотечний офіс
class Bibliotechue(models.Model):
    title = models.CharField("Заголовок", max_length=600)
    content = models.TextField("Контент", blank=True, null=True)

    def __str__(self):
        return self.title
```


ДОДАТОК Д

admin.py

```
from django.contrib import admin
from .models import News, Administrator, AdmissionInfo, NUSHSection, LegalDocument,
ProfGroups, MetodOffice, BehaviourOffice, Bibliotechue, ReceptionSchedule

admin.site.register(News)
admin.site.register(Administrator)
admin.site.register(AdmissionInfo)
admin.site.register(NUSHSection)
admin.site.register(LegalDocument)
admin.site.register(ProfGroups)
admin.site.register(MetodOffice)
admin.site.register(BehaviourOffice)
admin.site.register(Bibliotechue)
admin.site.register(ReceptionSchedule)
```