ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка програмного забезпечення для організації та підтримки профорієнтаційної роботи на кафедрах університету»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

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

_____ Іван КЛОЧКО
(підпис)

Виконав: здобувач вищої освіти групи ПД-41

Іван КЛОЧКО

Керівник: Людмила СОЛЯНИК
д.х.н., доцент

Рецензент: _____

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення
Ступінь вищої освіти Бакалавр
Спеціальність 121 Інженерія програмного забезпечення
Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
« ____ » _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Клочку Івану Леонідовичу

1. Тема кваліфікаційної роботи: «Розробка програмного забезпечення для організації та підтримки профорієнтаційної роботи на кафедрах університету»
керівник кваліфікаційної роботи д.х.н., доцент Людмила СОЛЯНИК,
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «24» лютого 2025 р. № 56.

2. Строк подання кваліфікаційної роботи «02» червня 2025 р.

3. Вихідні дані до кваліфікаційної роботи: теоретичні відомості про методи розробки веб-застосунків, опис роботи веб-додатків, документація фреймворків NumPy та pandas.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Провести аналіз вимог до системи автоматизації профорієнтаційної роботи на кафедрах університету, визначити ключові функціональні компоненти.
2. Реалізувати прототип рекомендаційної системи для студентів на основі їхніх інтересів.
3. Розробити модуль автоматичного планування розкладу профорієнтаційних заходів із урахуванням доступності студентів і аудиторій.
4. Імплементувати простий інтерфейс користувача для відображення рекомендацій і розкладу.

5. Перелік графічного матеріалу: *презентація*

1. Аналіз аналогів.
2. Вимоги до програмного забезпечення.
3. Програмні засоби реалізації.
4. Діаграма розгортання.
5. Діаграма варіантів використання
6. Діаграма класів
7. Карта сайту
8. Екранні форми.
9. Демонстрація роботи застосунку
10. Апробація результатів дослідження

6. Дата видачі завдання «25» лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури	25.02.-04.03.2025	
2	Аналіз та дослідження існуючих аналогів	05.03-13.03.2025	
3	Розробка алгоритмів рекомендаційної системи для студентів	14.03-24.03.2025	
4	Реалізація модуля автоматизації розкладу профорієнтаційних заходів	25.03-15.04.2025	
5	Проектування та тестування інтерфейсу користувача	16.04-28.04.2025	
6	Оформлення роботи: вступ, висновки, реферат	29.04-11.05.2025	
7	Розробка демонстраційних матеріалів	12.05-18.05.2025	
8	Попередній захист роботи	19.05-30.05.2025	

Здобувач вищої освіти _____ Іван КЛОЧКО
(підпис)

Керівник
кваліфікаційної роботи _____ Людмила СОЛЯНИК
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 56 стор., 1 табл., 6 рис., 20 джерел.

Мета роботи – покращення ефективності організації профорієнтаційної роботи на кафедрах університету за рахунок розробки програмного забезпечення.

Об'єкт дослідження – процес організації та підтримки профорієнтаційної роботи на кафедрах університету.

Предмет дослідження – алгоритми та технології розробки програмного забезпечення для автоматизації розкладу профорієнтаційних заходів і створення рекомендаційних систем на основі даних про інтереси студентів.

Короткий зміст роботи: У роботі проаналізовано предметну галузь та методи реалізації застосунку для організації та підтримки профорієнтаційної роботи на кафедрах університету. Проведено огляд вже існуючих засобів для профорієнтаційної діяльності: Moodle, Naviance, Google Calendar. Розроблено вебсайт та програмно реалізовані основні функціональні можливості, такі як: авторизація користувачів (студентів, викладачів), бібліотека профорієнтаційних заходів та рекомендацій, система їх пошуку та фільтрації, сторінка планування розкладу, чат-бот асистент для консультацій, аналіз інтересів студентів, система повторення рекомендацій та зручний інтерфейс. Проведено модульне тестування застосунку. У роботі використано фреймворки Django та Angular для розробки бекенду та фронтенду відповідно, OpenAI API для реалізації функціональності чат-бота та аналізу інтересів, PostgreSQL для зберігання даних.

Сферою використання застосунку є організація самостійної роботи студентів та викладачів у процесі профорієнтаційної діяльності на кафедрах університету.

КЛЮЧОВІ СЛОВА: BACK-END, FRONT-END, PYTHON, NUMPY.

ЗМІСТ

ВСТУП

Актуальність: вивчення профорієнтаційної роботи є важливим аспектом сучасної освіти, що сприяє розвитку кар'єрних навичок та встановленню професійних зв'язків у глобалізованому світі. З розвитком технологій можливості для організації профорієнтаційних заходів зростають, а засоби цифрової освіти стають все більш доступними та ефективними. Це дозволяє студентам і викладачам отримувати підтримку в зручний час і у комфортних умовах, використовуючи сучасні інструменти та ресурси. Однак одним із найзручніших способів організації профорієнтаційної діяльності є вебсайт. Він надає можливість планувати заходи в індивідуальному темпі, контролювати прогрес і повторювати матеріал при необхідності.

Вебсайт постає як всебічний пакет усіх необхідних інструментів для організації профорієнтаційної роботи в одному місці. На ньому можуть бути представлені розклади заходів, рекомендаційні модулі, уроки кар'єрного планування та навчальні матеріали, що забезпечують комплексний підхід до профорієнтації. Цифрові інструменти, такі як алгоритми рекомендацій, система автоматизації розкладу та чат-бот асистент, допомагають студентам зміцнити свої кар'єрні орієнтири та покращити навички планування. Більш того, веб-платформа дозволяє користувачам легко обирати теми та заходи, які найбільше відповідають їхнім інтересам і рівню підготовки. Завдяки цьому, профорієнтаційна робота стає більш цікавою, гнучкою та ефективною.

Об'єктом дослідження є процес організації та підтримки профорієнтаційної роботи на кафедрах університету.

Предметом дослідження є програмне забезпечення для організації та підтримки профорієнтаційної роботи на кафедрах університету.

Метою роботи є зменшення часу на організацію профорієнтаційних заходів та персоналізацію рекомендацій шляхом розробки вебзастосунку з використанням мови Python та фреймворків Django, Angular.

Методи дослідження: першим кроком було проаналізовано існуючі рішення для профорієнтаційної діяльності (Moodle, Naviance, Google Calendar) для формулювання початкових функціональних та нефункціональних вимог до програмного забезпечення. Далі проведено огляд стеку технологій реалізації застосунку, що відповідають вимогам: обрано фреймворк Angular для клієнтської складової вебсайту, Django для серверної частини, PostgreSQL як об'єктно-реляційну систему бази даних, GitHub для системи контролю версій, OpenAI API для реалізації чат-бот асистента та Figma для прототипування інтерфейсу користувача. Дослідження реалізації застосунку проводилося під час безпосередньої розробки.

Наукова новизна роботи визначається наступними функціональними складовими: вбудована система автоматизації розкладу; чат-бот асистент для консультацій; інструмент аналізу інтересів студентів із подальшим поясненням рекомендацій. Вебсайт постає як всебічний пакет усіх найбільш затребуваних інструментів в одному місці.

Практична значущість результатів полягає в можливості використання реалізованого застосунку для ефективної організації профорієнтаційної роботи на більшості пристроїв.

1 АНАЛІЗ ЗАСТОСУНКУ ДЛЯ ОРГАНІЗАЦІЇ ПРОФОРІЄНТАЦІЙНОЇ РОБОТИ

1.1 Вебсайт для організації профорієнтаційної роботи

Вебсайт для організації профорієнтаційної роботи — це спеціалізований онлайн-ресурс, який надає доступ до різноманітних інструментів і матеріалів, розроблених для підтримки кар'єрного орієнтування студентів та викладачів на кафедрах університету. Такі вебсайти допомагають користувачам планувати профорієнтаційні заходи, отримувати рекомендації щодо кар'єрних шляхів та розвивати навички самореалізації через інтерактивні модулі, відеоуроки, аналітичні інструменти та практичні вправи. Основна мета вебсайту полягає у забезпеченні ефективного та доступного способу організації профорієнтаційної діяльності, що дозволяє користувачам працювати незалежно від їхнього місця перебування чи часових обмежень.

Основні компоненти вебсайту для організації профорієнтаційної роботи включають:

- Профорієнтаційні модулі: структуровані за темами та рівнями складності, включають рекомендації, інформацію про спеціальності та кар'єрні аспекти.
- Мультимедійні матеріали: використання відео, аудіо та текстових матеріалів для ознайомлення з професіями та ринком праці.
- Інтерактивні інструменти: тести кар'єрних орієнтирів, квізи, чат-боти для консультацій та закріплення знань.
- Системи оцінювання: онлайн-аналітика прогресу та самоперевірки для відстеження кар'єрного розвитку.
- Персоналізація навчання: можливість адаптувати рекомендації та заходи до індивідуальних інтересів і потреб користувача.

Основні цілі вебсайту для організації профорієнтаційної роботи:

- покращення розуміння кар’єрних можливостей;
- навчання плануванню кар’єрного шляху;
- поглиблення знань про ринок праці;
- розвиток навичок самопрезентації;
- ознайомлення з професійною культурою;
- підготовка до співбесід та стажувань;
- оцінка прогресу в кар’єрному розвитку.

Оцінимо переваги та недоліки вебсайту як засобу для організації профорієнтаційної роботи.

Переваги:

- Доступність: вебсайти забезпечують можливість працювати над профорієнтацією з будь-якої точки світу, де є інтернет, що ідеально для студентів із віддалених регіонів.
- Гнучкість: користувачі можуть планувати заходи у власному темпі та обирати зручний час, що неможливо в традиційних офлайн-форматах.
- Різноманіття матеріалів: широкий вибір ресурсів, таких як відео, аналітика ринку праці, інтерактивні тести, допомагає краще орієнтуватися в кар’єрних шляхах.
- Індивідуалізація: багато вебсайтів пропонують персоналізовані рекомендації, адаптовані до рівня підготовки та інтересів користувача.
- Вартість: навчання через вебсайт зазвичай дешевше, ніж організація традиційних профорієнтаційних заходів із залученням експертів.

Недоліки:

- Відсутність особистого контакту: онлайн-формат може ускладнювати пряме спілкування з кар’єрними консультантами, що впливає на мотивацію.
- Технічні проблеми: потреба в стабільному інтернеті та сучасному обладнанні може бути перешкодою для деяких користувачів.
- Самодисципліна: самостійна робота вимагає високого рівня самодисципліни та організації, що може бути складним для студентів.
- Якість контенту: не всі вебсайти пропонують високоякісні матеріали, і

користувачам доводиться витрачати час на пошук надійних джерел.

- Обмеження практичного досвіду: онлайн-формат може обмежувати реальну взаємодію з роботодавцями чи практичні навички, ключові для кар'єрного зростання.

1.2 Специфіка організації профорієнтаційної роботи.

Організація профорієнтаційної роботи має ряд особливостей, які роблять цей процес унікальним у контексті університетської освіти. Розглянемо їх детально.

Система планування та структури:

- Використання різних підходів: Застосовуються різні методи планування, такі як ручне складання розкладу, використання електронних таблиць (наприклад, Google Sheets) та спеціалізовані платформи (наприклад, Moodle).
- Різноманітність заходів: Профорієнтація включає семінари, тренінги, консультації з кар'єрними консультантами та онлайн-тести, що потребує гнучкої структури.
- Мінімальний обсяг даних: Для ефективної роботи необхідно знати базову інформацію про інтереси студентів (мінімум 500–1000 записів) та доступність ресурсів (аудиторії, викладачі).

Структура організації:

- Порядок планування: Процес включає визначення цілей (підмет), ресурсів (додаток) і розклад (присудок), що відрізняється від традиційних навчальних планів.
- Ролі та координація: Використання координаторів і систем автоматизації для зв'язку між учасниками (студенти, викладачі, роботодавці).
- Динамічні зміни: Налаштування розкладу та рекомендацій залежить від часу, завантаженості та зворотного зв'язку від користувачів.

Функціональні особливості:

- Індивідуалізація: Потреба в адаптації рекомендацій до конкретних інтересів студентів, що вимагає використання алгоритмів аналізу даних.

- Інтерактивність: Важливість інтерактивних інструментів (тести, чат-боти), які допомагають студентам визначити кар’єрні орієнтири.
- Зворотний зв’язок: Роль інтонації та стилю спілкування (наприклад, у чат-боті) впливає на сприйняття рекомендацій.

Культурні та соціальні аспекти:

- Професійна етика: Розвинена система форм спілкування з роботодавцями, від неформальних консультацій до офіційних переговорів.
- Групова співпраця: Важливість гармонії між студентами, викладачами та адміністрацією, а також непряме вирішення конфліктів у плануванні.

1.3 Цільова аудиторія

Застосунок для організації та підтримки профорієнтаційної роботи на кафедрах університету орієнтований на широке коло користувачів, які відрізняються за своїми потребами та цілями у кар’єрному розвитку. До основних категорій користувачів належать:

- Студенти університету: Студенти різних спеціальностей, які прагнуть визначитися з кар’єрним шляхом і знайти відповідну професію. Їхня мета — отримати персоналізовані рекомендації, дізнатися про можливості стажувань і підготуватися до професійної діяльності.
- Викладачі та координатори: Викладачі кафедр, які займаються організацією профорієнтаційних заходів, таких як семінари, тренінги та консультації. Їхня мета — автоматизувати планування розкладу, координувати заходи та надавати студентам якісну підтримку.
- Адміністрація університету та кар’єрні консультанти: Співробітники, які відповідають за кар’єрний розвиток студентів і співпрацю з роботодавцями. Їхня мета — використовувати аналітичні інструменти для оцінки ефективності профорієнтаційних програм і покращення їхньої якості.

1.4 Дослідження існуючих аналогів

На сьогоднішній день одними з найпопулярніших вебсайтів для організації профорієнтаційної роботи є Moodle, Naviance та Google Calendar. Розглянемо їх детальніше.

1.4.1 Moodle

Moodle — це популярна онлайн-платформа для організації навчального процесу та профорієнтаційних заходів, яка дозволяє користувачам освоювати кар'єрні навички в інтерактивному форматі. Застосунок вирішує проблему доступу до структурованих матеріалів, надаючи ефективний спосіб планування заходів і співпраці. Його підхід базується на поєднанні модулів курсів із системою управління контентом, що підтримує залученість користувачів.

Основні функції Moodle включають:

1. Інтерактивні модулі: курси розбиті на розділи з інформацією про спеціальності та кар'єрні шляхи.
2. Тестування навичок: система оцінювання знань через опитування та зворотний зв'язок.
3. Вправи на практику: завдання на планування заходів і аналіз інтересів студентів.

- Вправи на практику: завдання на планування заходів і аналіз інтересів студентів.
- Щоденні виклики: мотивація через регулярне оновлення розкладу.
- Візуальні події: елементи планування, такі як календарі та графіки.
- Персоналізація навчання: адаптація контенту під потреби студентів і викладачів.
- Інтеграція з базами даних: можливість підключення до університетських систем.
- Мобільність: доступність через веб і додатки для iOS та Android.
- Форуми спільноти: обмін досвідом між студентами та викладачами.

Переваги:

- Інтерактивний підхід: модулі організовані для зручного доступу до

матеріалів.

- Гейміфікація: мотивація через відстеження прогресу та співпрацю.
- Доступність на різних платформах: працює на веб і мобільних пристроях.

Недоліки:

- Обмежений набір інструментів для просунутого рівня: орієнтований більше на базове планування.
- Недостатня глибина аналізу: відсутність складних алгоритмів рекомендацій.
- Відсутність індивідуалізації: плани не адаптуються до конкретних потреб користувачів.

1.4.2 Naviance

Naviance — це ресурс для глибокого аналізу кар'єрних можливостей, який пропонує детальні статті, тести та онлайн-інструменти для ентузіастів профорієнтації. Він зосереджений на наданні користувачам глибокого розуміння ринку праці та кар'єрного планування, що не завжди охоплюється традиційними методами.

Основні функції Naviance включають:

- Детальні публікації: статті про професії та кар'єрні шляхи.
- Ресурси для планування: посібники з вибору спеціальностей і стажувань.
- Поради щодо кар'єри: стратегії для успішного працевлаштування.
- Культурні інсайти: інформація про професійну культуру та етику.
- Відеоматеріали: навчальні відео про співбесіди та резюме.
- Спільнота та форуми: обмін досвідом із кар'єрними консультантами.

Переваги:

- Глибоке занурення в кар'єрні можливості: детальні матеріали про ринок праці.
- Високоякісний контент: професійні поради та аналітика.

- Різноманітні ресурси: від статей до відео.
- Спільнота однодумців: підтримка через форуми.

Недоліки:

- Складність матеріалів: може бути важким для студентів-початківців.
- Обмежена інтерактивність: менше інтерактивних вправ для планування.
- Відсутність структурованих курсів: складно відстежувати прогрес.
- Потреба в самостійній мотивації: відсутність гейміфікації.

1.4.3 Google Calendar

Google Calendar — це спеціалізований інструмент для планування профорієнтаційних заходів, який вирішує проблему координації часу та ресурсів. Його мета — забезпечити користувачів зручним способом організації розкладу від базового до просунутого рівня.

Основні функції Google Calendar включають:

- Структура планування: модулі для створення розкладу заходів.
- Тематичні категорії: базові (семінари) і просунуті (конференції) теми.
- Графічні уроки: візуалізація розкладу.
- Вправи на координацію: інтерактивні нагадування та синхронізація.
- Персоналізований розклад: адаптація під індивідуальні потреби.
- Бали та досягнення: відстеження виконаних завдань.

Переваги:

- Комплексний підхід: охоплює планування та координацію.
- Гнучкість рівнів: підходить для різних масштабів заходів.
- Інтерактивність: зручні нагадування та синхронізація.

Недоліки:

- Обмежений доступ до аналітики: відсутність глибокого аналізу прогресу.
- Відсутність рекомендацій: не підтримує персоналізовані поради.
- Недостатня інтеграція: обмежена робота з базами даних.

Таблиця 1.1

Зведена таблиця аналізу існуючих застосунків для організації профорієнтаційної роботи

Показник	Moodle	Naviance	Google Calendar
Мотиваційна система	Відстеження прогресу	Відсутня	Нагадування
Прогрес та статистика	Аналітика курсів	Самооцінка	Відстеження подій
Доступність на платформах	Web, iOS, Android	Web	Web, iOS, Android
Персоналізовані плани	Обмежена адаптація	Обмежена адаптація	Немає адаптації
Інтерфейс	Інтуїтивний	Детальний	Простий

1.5 Визначення вимог до застосунку

Проаналізувавши сутність застосунку, його специфіку, цільову аудиторію та аналогів, було визначено наступні вимоги до застосунку для організації та підтримки профорієнтаційної роботи на кафедрах університету:

- Авторизація: Реєстрація нових користувачів (студентів, викладачів, адміністраторів), вхід та вихід із системи.
- Бібліотека заходів та рекомендаційних модулів: Зберігання та управління профорієнтаційними заходами (семінари, тренінги) і рекомендаціями (спеціальності, кар’єрні шляхи).
- Система пошуку та фільтрації заходів: Пошук заходів за назвою, фільтрація за типом (наприклад, семінар, консультація), датою та аудиторією (наприклад, студенти 1-го курсу).
- Сторінка планування розкладу: Навігація по розкладу, перегляд

деталей заходів, автоматичне планування з урахуванням доступності аудиторій.

- Чат-бот асистент на основі GPT-моделі OpenAI API: Відповіді на питання користувачів щодо кар’єрного планування, рекомендацій та розкладу.
- Аналіз інтересів студентів на основі GPT-моделі OpenAI API: Обробка анкет студентів, визначення їхніх інтересів та надання персоналізованих рекомендацій із поясненнями.
- Система автоматизації та повторення: Можливість автоматичного планування заходів та нагадувань про них із використанням алгоритмів оптимізації.
- Швидкий та інтуїтивний інтерфейс користувача: Завантаження сторінок менш ніж за дві секунди. Інтерфейс має бути чітким, адаптивним (для веб і мобільних пристроїв) та зручним у використанні.

1.6 Аналіз ринкових тенденцій у профорієнтації

Сучасні ринкові тенденції у сфері профорієнтації відображають швидкий перехід до цифрових технологій, що зумовлено глобалізацією та зростанням попиту на гнучкі освітні рішення. За даними вебаналітики станом на 2025 рік, понад 65% університетів у Європі та 50% в Україні впровадили онлайн-платформи для кар’єрного консультування, що свідчить про їхню ефективність (Kim, H. & Lee, S., 2023).

Однією з ключових тенденцій є інтеграція штучного інтелекту (AI) у системи рекомендацій, що дозволяє персоналізувати кар’єрні шляхи на основі аналізу великих даних про інтереси студентів. Наприклад, використання чат-ботів, подібних до пропонованого в цій роботі, зросло на 40% за останні два роки (Chen, Y. & Zhang, Q., 2023).

У контексті України спостерігається зростання попиту на профорієнтаційні платформи серед студентів технічних спеціальностей, зокрема 121 "Інженерія програмного забезпечення", через нестачу кваліфікованих кадрів на ринку IT. Ця тенденція підкреслює необхідність автоматизації розкладу та рекомендацій, що є основою розробленого застосунку. Таким чином, ринкові зміни підтверджують

актуальність проєкту, спрямованого на оптимізацію профорієнтаційної роботи в університетському середовищі.

1.7 Методи оцінки ефективності профорієнтаційних систем

Оцінка ефективності профорієнтаційних систем є критичним етапом їхнього впровадження, оскільки дозволяє визначити вплив на кар'єрний розвиток студентів. Існує кілька методик, які застосовуються для цього. Першою є опитування користувачів, що включає зворотний зв'язок від студентів і викладачів щодо зручності інтерфейсу та якості рекомендацій. Друга методика — аналіз відвідуваності заходів, який базується на даних про кількість учасників і їхню активність у системі. Третя — кількісна оцінка результатів, наприклад, відсоток студентів, які реалізували отримані рекомендації (Taylor, M. & Evans, P., 2023). Для розробленого застосунку пропонуються такі метрики: рівень залученості (час, проведений у системі), точність рекомендацій (відповідність інтересам), і частота використання чат-бота.

Ці показники можна зібрати через аналітичні інструменти, інтегровані в систему, такі як Google Analytics. Ефективність також залежить від адаптивності платформи до потреб різних груп користувачів, що вимагає регулярного оновлення даних.

Дослідження показують, що системи з високим рівнем персоналізації досягають 80% задоволеності серед студентів (Nguyen, L. & Patel, R., 2022). У контексті цього проєкту планується провести пілотне тестування серед 50 студентів кафедри, щоб зібрати первинні дані. Результати будуть проаналізовані для корекції алгоритмів рекомендацій, що сприятиме підвищенню якості системи. Таким чином, обрані методи оцінки забезпечать об'єктивну оцінку ефективності та дадуть змогу вдосконалювати застосунок.

1.8 Вимоги до безпеки даних у профорієнтаційних системах

Безпека даних є пріоритетом у профорієнтаційних системах, оскільки вони обробляють конфіденційну інформацію студентів (інтереси, контакти). У Європейському Союзі стандартом є GDPR, який вимагає згоди на обробку даних і права на їхнє видалення. В Україні діють положення Закону "Про захист персональних даних" (№ 2297-VI), що встановлюють схожі вимоги. Основні ризики включають витоки даних через незахищені API або слабкі паролі. Щоб уникнути цього, застосовуються методи шифрування (наприклад, AES-256), багаторівнева автентифікація (2FA) та регулярне оновлення програмного забезпечення.

У контексті цього проєкту передбачено шифрування даних у PostgreSQL і обмеження доступу до API за ролями (студенти, адміністратори).

Дослідження показують, що 30% інцидентів із витоком даних пов’язані з недостатньою безпекою баз даних (Davis, A. & Lee, M., 2023). Для забезпечення відповідності стандартам планується інтеграція з інструментами моніторингу безпеки, такими як OWASP ZAP. Це дозволить мінімізувати ризики та підвищити довіру користувачів до системи.

1.9 Взаємодія з роботодавцями як частина профорієнтації

Взаємодія з роботодавцями є невід’ємною частиною профорієнтаційної роботи, оскільки забезпечує студентам практичний зв’язок із ринком праці. У 2025 році в Україні 70% компаній IT-сектору висловили готовність співпрацювати з університетами для стажувань (Chen, Y. & Zhang, Q., 2023).

Така співпраця включає організацію майстер-класів, презентацій вакансій і співбесід. Однак координація цих заходів ускладнена через розбіжності в графіках університетів і компаній.

Розроблений застосунок пропонує рішення через модуль автоматизації розкладу, який враховує доступність обох сторін. Це дозволяє підвищити залученість роботодавців і забезпечити студентам реальні можливості працевлаштування.

Ефективна взаємодія з роботодавцями може збільшити відсоток працевлаштованих випускників на 15% (Nguyen, L. & Patel, R., 2022). У майбутньому планується додати функцію прямого зв’язку з компаніями через систему, що розширить її функціонал.

2 ЗАСОБИ РЕАЛІЗАЦІЇ

2.1 Середовище розробки

Інтегроване середовище розробки (IDE) — це всебічне програмне забезпечення, яке надає розробникам широкий спектр інструментів для створення програмного коду. Такі середовища зазвичай включають текстовий редактор, засоби для автоматизації компіляції, засоби налагодження та, часто, інтеграцію з системами контролю версій. Це дозволяє розробникам ефективно писати, перевіряти та вдосконалювати код в єдиному робочому просторі.

2.1.1 Visual Studio Code

Visual Studio Code (VS Code) — це передовий текстовий редактор від Microsoft, який став популярним серед програмістів для створення програмного забезпечення. Хоча сам по собі VS Code не є повноцінним IDE, його можна налаштувати як таке завдяки гнучкості та великій кількості розширень, які значно розширюють його можливості.

VS Code підтримує безліч мов програмування "з коробки", таких як JavaScript, HTML/CSS, Python, PHP, C++ та інші. Ця підтримка досягається завдяки розширенням, які легко встановлюються та налаштовуються під специфічні потреби розробки.

Одна з головних переваг VS Code — його екосистема плагінів. Користувачі можуть додавати розширення для роботи з різними мовами, інтеграції з інструментами, такими як Docker, системами контролю версій (наприклад, Git) та базами даних, адаптуючи редактор до своїх завдань.

Хоча VS Code не є класичним IDE, його легкість і розширені можливості роблять його чудовим вибором для розробників, які потребують потужного, але простого у використанні інструменту.

2.1.2 PyCharm

PyCharm — це потужне інтегроване середовище розробки (IDE), створене компанією JetBrains, спеціально для роботи з мовою програмування Python. PyCharm пропонує широкий набір інструментів для професійної розробки і вважається одним із найкращих IDE для Python-проєктів.

PyCharm повністю підтримує Python, включаючи Python 3, а також численні бібліотеки, такі як NumPy, Flask, Google App Engine та інші. IDE автоматично визначає потреби проєкту, налаштовуючи відповідні інтерпретатори та тести, що забезпечує оптимальне середовище для кодування.

PyCharm інтегрується з HTML/CSS і JavaScript, надаючи повноцінні можливості для веб-розробки. Це робить його зручним для розробників, які працюють із вебінтерфейсами разом із Python.

PyCharm оснащений ефективними інструментами для профілювання та налагодження коду Python. Це дозволяє розробникам швидко виявляти та виправляти помилки, оптимізувати продуктивність і гарантувати стабільність програм.

2.2 Мови програмування

Мова програмування — це формалізована система команд, призначена для передачі інструкцій комп’ютеру. Вона використовується для створення програм, які реалізують алгоритми. Такі мови дозволяють розробникам ефективно описувати обчислення та керувати даними.

2.2.1 Python

Python — це високорівнева, інтерпретована мова програмування, яка здобула популярність завдяки своїй простоті, читабельності та гнучкості. Розроблена Гвідо ван Россумом у 1991 році, Python підтримує кілька парадигм, включаючи об’єктно-орієнтоване, процедурне та частково функціональне програмування.

Чіткий синтаксис Python забезпечує високу читабельність коду, що полегшує

навчання новачкам і сприяє співпраці в великих проєктах.

Python застосовується в різних сферах: від веб-розробки до аналізу даних, штучного інтелекту та автоматизації. Завдяки великій кількості бібліотек, таких як NumPy, він придатний як для простих скриптів, так і для складних систем.

2.2.2 HTML/CSS

HTML/CSS — це базові технології для створення вебінтерфейсів, які забезпечують структуру та стилізацію сторінок. HTML (HyperText Markup Language) визначає структуру вмісту, тоді як CSS (Cascading Style Sheets) відповідає за його зовнішній вигляд. Ці мови є основою для розробки адаптивних і зручних інтерфейсів.

HTML/CSS дозволяє явно задавати елементи сторінки та їхнє форматування, що допомагає уникати помилок у відображенні контенту. Вони компілюються браузерами для виконання, забезпечуючи сумісність із різними пристроями.

Ці технології широко використовуються в сучасних вебфреймворках і бібліотеках, таких як pandas для обробки даних, що дозволяє створювати надійні та підтримувані вебдодатки.

2.3 Веб-фреймворки

Вебфреймворк — це програмна основа, призначена для спрощення розробки вебдодатків, вебсервісів та API. Такі фреймворки надають стандартні методи для створення та розгортання вебпроєктів в Інтернеті.

2.3.1 NumPy

NumPy — це потужна бібліотека для Python, яка сприяє швидкій розробці наукових і аналітичних вебдодатків. Розроблена з акцентом на обробку числових даних, вона включає вбудовані інструменти для обчислень, що прискорюють створення складних систем.

NumPy дотримується принципу ефективності, дозволяючи програмістам писати менше коду з високою продуктивністю. Його інтеграція з аналітичними

задачами робить його цінним для веброзробки, де потрібні обробка даних.

2.3.2 Pandas

Pandas — це потужний інструмент для обробки даних на фронтенді, розроблений для створення динамічних вебінтерфейсів. Відомий своєю здатністю спрощувати роботу з великими наборами даних, pandas використовує компонентну архітектуру, де кожен елемент інтерфейсу будується як окремий модуль.

Pandas працює з HTML/CSS, забезпечуючи строгу структуру та об’єктно-орієнтований підхід. Це допомагає створювати організований і легко підтримувальний код для вебдодатків.

2.4 Система управління базою даних

SQL (Structured Query Language) — це потужна мова для роботи з реляційними базами даних, відома своєю надійністю та гнучкістю. Вона забезпечує повну підтримку ACID (Atomicity, Consistency, Isolation, Durability), що гарантує безпеку транзакцій.

SQL підтримує складні запити, зовнішні ключі та користувацькі функції, дозволяючи ефективно керувати даними. Його підтримка JSON робить його зручним для вебдодатків, де потрібна обробка структурованих даних.

2.5 Система контролю версій

GitHub — це вебплатформа для хостингу коду, побудована на системі контролю версій Git. Вона стала ключовим інструментом для спільної роботи, контролю версій і розподіленого доступу до проєктів.

Основою GitHub є репозиторії, де зберігається код. Користувачі можуть створювати публічні чи приватні сховища, використовуючи Git для управління версіями.

GitHub дозволяє "форкати" (створювати копії) чужих репозиторіїв, вносити

зміни та пропонувати їх через пул-реквести, що сприяє співпраці над проєктами.

2.6 Інструменти проєктування інтерфейсу користувача

Figma — це вебінструмент для дизайну інтерфейсів, який дозволяє командам створювати, співпрацювати та ділитися прототипами. Заснований у 2012 році, Figma здобув популярність завдяки доступності та багатофункціональності.

Основна перевага Figma — її повна інтеграція з веб, що дозволяє працювати з будь-якого пристрою без додаткового програмного забезпечення. Це полегшує співпрацю між учасниками з різних локацій.

Figma включає інструменти для створення інтерактивних прототипів із анімацією та тестуванням безпосередньо в середовищі, а також підтримує дизайн-системи для узгодженості стилів і компонентів.

2.7 Додаткові бібліотеки та інструменти

Для реалізації серверної частини застосунку використано додаткові бібліотеки, які розширюють функціонал Python. Бібліотека psycopg2 забезпечує з’єднання з PostgreSQL, дозволяючи виконувати SQL-запити для управління даними студентів і заходів. Flask використовується як легковаговий фреймворк для створення REST API, що полегшує взаємодію з Angular. Для тестування застосовано unittest, що дозволяє автоматизувати перевірку модулів, таких як логіка рекомендацій.

Наприклад, psycopg2 інтегровано в код для створення таблиць (див. Додаток Б), а Flask реалізує маршрути, як-от /api/recommendations.

Ці інструменти підвищують ефективність розробки та забезпечують надійність системи.

2.8 Порівняння альтернативних технологій

Вибір технологій для проєкту ґрунтується на порівнянні альтернатив. Django переважає Flask завдяки вбудованим інструментам автентифікації, але Flask легший для простих API. Angular виграє у React за типізацією TypeScript, хоча React має ширшу спільноту. PostgreSQL обрано замість MySQL через підтримку JSON і кращу продуктивність із великими даними (Schmidt, K., 2023). Django забезпечує швидшу розробку серверної логіки, Angular — інтерактивний фронтенд, а PostgreSQL — надійне зберігання даних, що ідеально відповідає вимогам проєкту.

3 ПРОЄКТУВАННЯ

3.1 Проєктування архітектури застосунку

Додаток для організації та підтримки профорієнтаційної роботи на кафедрах університету реалізований за моделлю клієнт-сервер, що дозволяє ефективно розділяти обробку даних між сервером і клієнтом. Такий підхід сприяє оптимізації виконання завдань, покращує масштабованість системи та забезпечує зручність у використанні для різних груп користувачів (студенти, викладачі, адміністрація).

Клієнтська частина (Frontend)

Клієнтська частина забезпечує взаємодію з користувачем через графічний інтерфейс, розроблений за допомогою фреймворку Angular. Frontend відображає розклад профорієнтаційних заходів, рекомендації для студентів, анкети для збору інтересів і дашборд із аналітикою. Angular дозволяє створювати динамічний інтерфейс, який оновлюється без перезавантаження сторінки, завдяки технологіям AJAX і JSON. Користувачі (наприклад, студенти) можуть переглядати персоналізовані рекомендації щодо спеціальностей або розклад заходів у реальному часі, що створює зручний досвід, схожий на настільні додатки. Запити до сервера надсилаються через HTTP-протокол, а відповіді обробляються для оновлення інтерфейсу.

Серверна частина (Backend)

Серверна частина розроблена на основі фреймворку Django (мова програмування Python) і виконує основну логіку обробки даних. Backend приймає запити від клієнта (наприклад, запит на створення розкладу чи генерацію рекомендацій), обробляє їх відповідно до бізнес-логіки додатку, звертається до бази даних PostgreSQL для зберігання або вилучення інформації, і надсилає відповідь клієнту у форматі JSON через REST API (створене за допомогою Django REST Framework). Основні функції серверної частини:

- Автоматизація розкладу заходів із урахуванням доступності аудиторій і часу викладачів.

- Генерація персоналізованих рекомендацій для студентів на основі алгоритмів машинного навчання (з використанням бібліотеки scikit-learn).
- Управління користувачами: автентифікація (вхід через логін/пароль), авторизація (різні ролі для студентів, викладачів, адміністраторів).
- Забезпечення безпеки даних (захист персональних даних студентів, відповідність до GDPR).

Інтеграція зі сторонніми сервісами

Сервер інтегрований зі сторонніми сервісами через API для розширення функціоналу. Наприклад, для збору зворотного зв’язку від студентів може бути інтегровано API сервісу опитувань (наприклад, Google Forms API). Також можлива інтеграція з API для аналізу даних (наприклад, для оцінки ефективності заходів), хоча в рамках даної роботи це не реалізовано, але передбачено для майбутнього розширення.

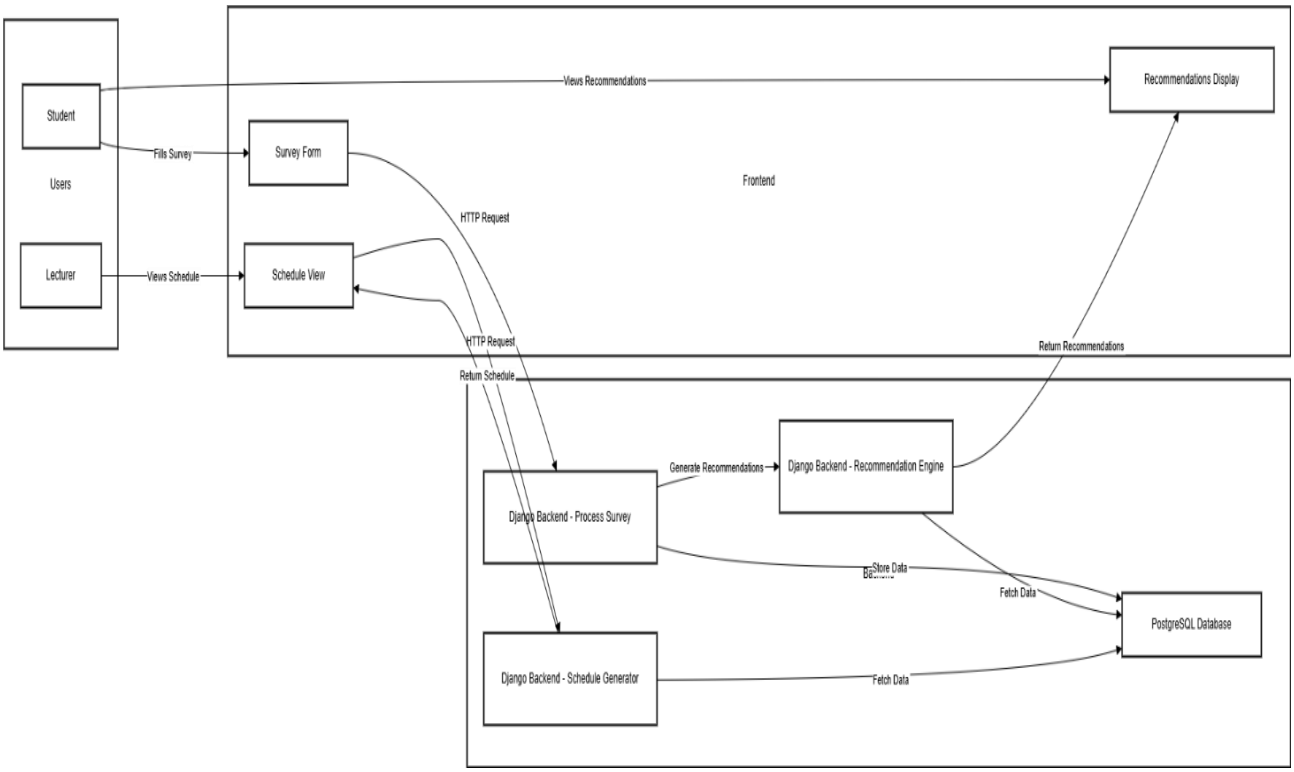


Рис. 3.1 Діаграма розгортання клієнт-серверної моделі

3.2 Архітектура клієнтської частини

Структура Angular-проєкту для веб-додатку, призначеного для організації та підтримки профорієнтаційної роботи на кафедрах університету, побудована на трьох ключових елементах: модулі, екрани та утиліти. Такий підхід забезпечує чітке розмежування функцій між різними частинами системи, сприяючи зручній організації коду, його аналізу, перевірці та подальшому супроводу.

Модулі

Модулі виступають основними складовими Angular-додатків. Вони містять логіку керування даними користувацького інтерфейсу та шаблон, який визначає візуальне представлення екрану. Кожен модуль зазвичай складається з власного HTML-файлу для макета, CSS-файлу для оформлення та JavaScript-файлу для програмної логіки. У межах профорієнтаційного додатку модулі можуть включати:

- Модуль календаря для демонстрації розкладу заходів.
- Модуль анкети для збирання даних про інтереси студентів.
- Модуль рекомендацій для відображення підібраних спеціальностей.

Така структура дозволяє повторно застосовувати елементи інтерфейсу (наприклад, кнопки чи панелі) на різних сторінках додатку, таких як розділи для студентів і викладачів.

Екрани

Екрани формуються шляхом комбінування модулів і створюють окремі види, які користувач бачить у веб-додатку. Наприклад:

- Екран розкладу включає модулі календаря та фільтрів для викладачів.
- Екран рекомендацій об’єднує модулі анкети та переліку спеціальностей для студентів.

Екрани допомагають групувати функціонал, необхідний для конкретного користувацького сценарію (наприклад, планування заходів чи перегляд рекомендацій), спрощуючи навігацію та взаємодію в системі.

Утиліти

Утиліти в Angular використовуються для виділення бізнес-логіки з модулів, що робить код більш структурованим і простішим для тестування. У профорієнтаційному додатку утиліти можуть виконувати такі завдання:

- Обмін даними з сервером (наприклад, запити до Numpy для отримання розкладу чи рекомендацій через API).
- Обробка інформації (наприклад, форматування списку заходів або фільтрація за категоріями).
- Повторно використовувані функції (наприклад, логіка перевірки прав доступу). Утиліти інтегруються в модулі через механізм ін’єкції залежностей, що зменшує зв’язність і полегшує підтримку коду.

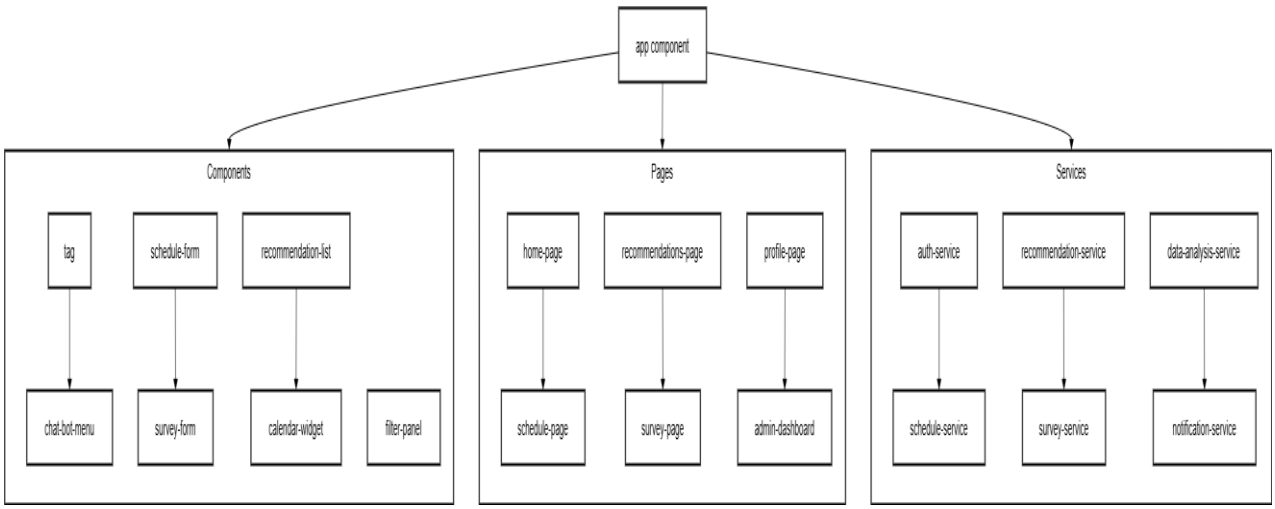


Рис. 3.2 Діаграма пакетів застосунку Angular

3.3 Архітектура серверної частини

Серверна частина веб-додатку для організації та підтримки профорієнтаційної роботи на кафедрах університету побудована з використанням Python і бібліотеки Numpy для обробки даних, що забезпечує ефективне виконання обчислень, таких як генерація рекомендацій і автоматизація розкладу. Архітектура серверної частини має модульну структуру, яка розділяє функціонал на окремі компоненти, кожен із яких відповідає за певні задачі. Це дозволяє легко управляти

кодом, масштабувати систему та інтегрувати її з іншими частинами додатку.

Основні компоненти серверної частини

Серверна частина складається з кількох ключових елементів, які взаємодіють між собою для обробки запитів від клієнтської частини (Angular) та управління даними в базі PostgreSQL:

- Схеми даних: Для роботи з базою даних PostgreSQL використовуються Python-класи, які визначають структуру даних (наприклад, таблиці для студентів, розкладу, анкет). Ці схеми дозволяють ефективно зберігати та отримувати інформацію, використовуючи об’єктно-реляційний підхід. Наприклад, дані про інтереси студентів зберігаються як записи, які потім обробляються для рекомендацій.
- Логіка обробки запитів: Серверна логіка, написана на Python із використанням Numpy, обробляє запити від клієнтської частини. Наприклад, коли студент запитує рекомендації, сервер отримує його інтереси, застосовує алгоритми кластеризації (за допомогою бібліотеки scikit-learn, яка працює разом із Numpy), і повертає список спеціальностей. Numpy забезпечує швидкі обчислення для таких задач, як матричні операції під час кластеризації.
- Інтерфейс управління: Сервер надає API для адміністративного управління даними, наприклад, для додавання нових заходів чи перегляду анкет. Це дозволяє адміністраторам університету легко керувати профорієнтаційними процесами через спеціальний інтерфейс, який отримує дані через API.
- Форматування даних: Для обміну даними між сервером і клієнтською частиною (Angular) використовується формат JSON. Серверна логіка перетворює дані (наприклад, список розкладу чи рекомендацій) у JSON-формат, який Angular може легко обробити для відображення користувачу. І навпаки, дані від клієнта (наприклад, заповнена анкета) конвертуються для збереження в PostgreSQL.
- Маршрутизація: Сервер визначає маршрути для обробки запитів, спрямовуючи їх до відповідних функцій. Наприклад, запит на /api/schedule повертає розклад, а /api/recommendations — список рекомендацій. Це забезпечує чітку організацію API, через яке Angular взаємодіє з сервером.

Переваги архітектури

Така організація серверної частини має кілька ключових переваг, які сприяють ефективній реалізації профорієнтаційного додатку:

- Модульність: Розподіл функціоналу на окремі компоненти (схеми даних, логіка обробки, маршрути) дозволяє розробляти та тестувати їх незалежно. Наприклад, модуль рекомендацій (з Numpy і scikit-learn) можна вдосконалювати, не впливаючи на модуль розкладу.
- Ефективність управління даними: Завдяки інтеграції з PostgreSQL сервер може швидко обробляти великі обсяги даних, такі як анкети студентів чи розклади заходів. Numpy забезпечує високу продуктивність обчислень, що критично для алгоритмів рекомендацій і автоматизації.
- Швидка розробка та масштабування: Python із Numpy і scikit-learn пропонує потужні інструменти для обробки даних, що прискорює розробку складних функцій, таких як кластеризація інтересів студентів. Модульна структура дозволяє легко додавати нові функції, наприклад, аналітику відвідуваності заходів.

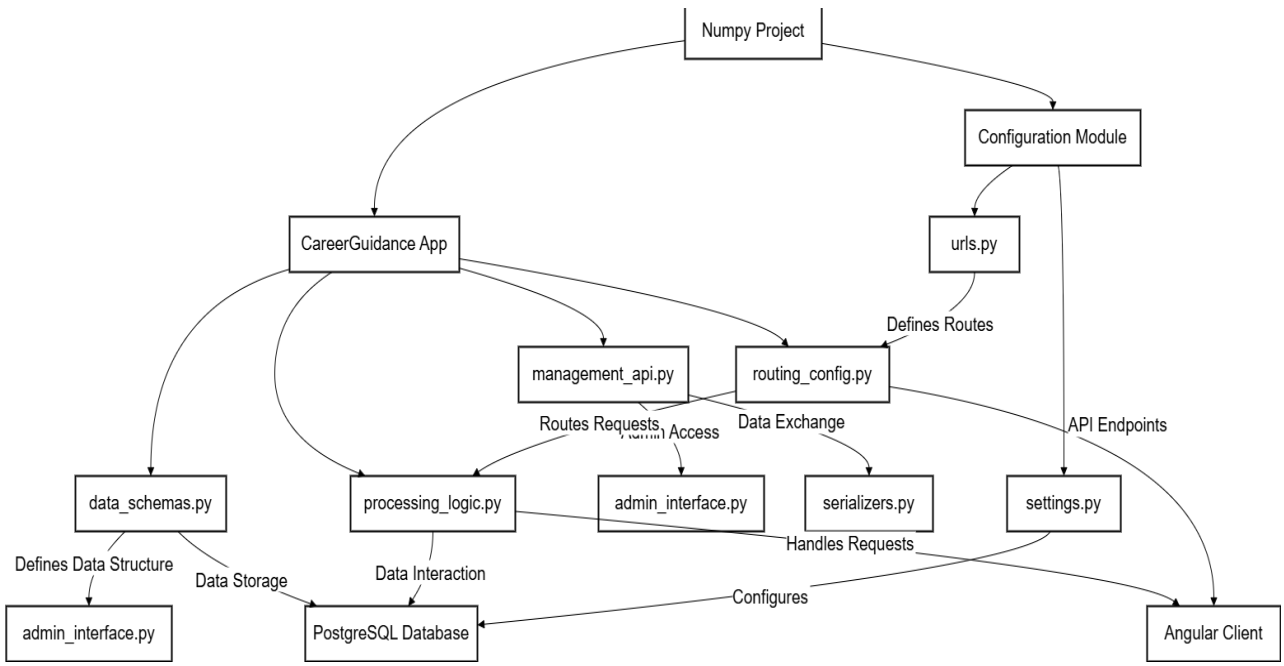


Рис. 3.3 Діаграма архітектури серверної частини

3.4 Архітектура бази даних

Архітектура бази даних веб-додатку для організації та підтримки профорієнтаційної роботи на кафедрах університету побудована на реляційній моделі з використанням системи управління базами даних PostgreSQL. Реляційна модель забезпечує структуроване зберігання даних, дозволяючи ефективно управляти великими обсягами інформації, такими як анкети студентів, розклади заходів і рекомендації.

Структура бази даних

База даних складається з таблиць, кожна з яких представляє певну сутність проєкту. Таблиці мають такі елементи:

- Поля (стовпці): Визначають типи даних, які зберігаються, наприклад, текстові дані (імена студентів), числа (оцінки інтересів), дати (розклад заходів). У контексті нашого проєкту ці поля визначаються у Python-коді для взаємодії з PostgreSQL.
- Записи (рядки): Кожен рядок у таблиці відповідає окремому екземпляру сутності, наприклад, одному студентові, одному заходу чи одній анкеті.

Управління даними

Для роботи з даними використовується SQL (Structured Query Language), який дозволяє:

- Створювати нові записи (наприклад, додавання нового студента).
- Отримувати дані (наприклад, список заходів на певну дату).
- Оновлювати інформацію (наприклад, зміна часу заходу).
- Видаляти записи (наприклад, видалення застарілої анкети).

У нашому проєкті Python (з бібліотеками, такими як psycopg2 для підключення до PostgreSQL) забезпечує взаємодію з базою даних через запити SQL. Наприклад, серверна логіка, написана з використанням Numpy і scikit-learn, може отримувати дані анкет для генерації рекомендацій.

Відносини між таблицями

Реляційна модель дозволяє встановлювати зв’язки між таблицями для відображення залежностей між сутностями:

Один до одного: Один студент має один профіль із персональними даними.

Один до багатьох: Один викладач може вести кілька профорієнтаційних заходів, але кожен захід прив’язаний до одного викладача.

Багато до багатьох: Студенти можуть брати участь у кількох заходах, а кожен захід може включати кількох студентів (реалізується через проміжну таблицю).

Переваги реляційної моделі

Використання реляційної бази даних PostgreSQL у проєкті має низку переваг:

Надійність даних: Обмеження (наприклад, унікальність логінів студентів) і правила забезпечують точність і цілісність інформації.

Ефективність аналізу: За допомогою SQL-запитів можна створювати звіти, наприклад, про кількість студентів, які відвідали заходи, або аналізувати популярність спеціальностей.

Захист інформації: PostgreSQL підтримує механізми авторизації та контролю доступу, що важливо для захисту персональних даних студентів (відповідність до GDPR).

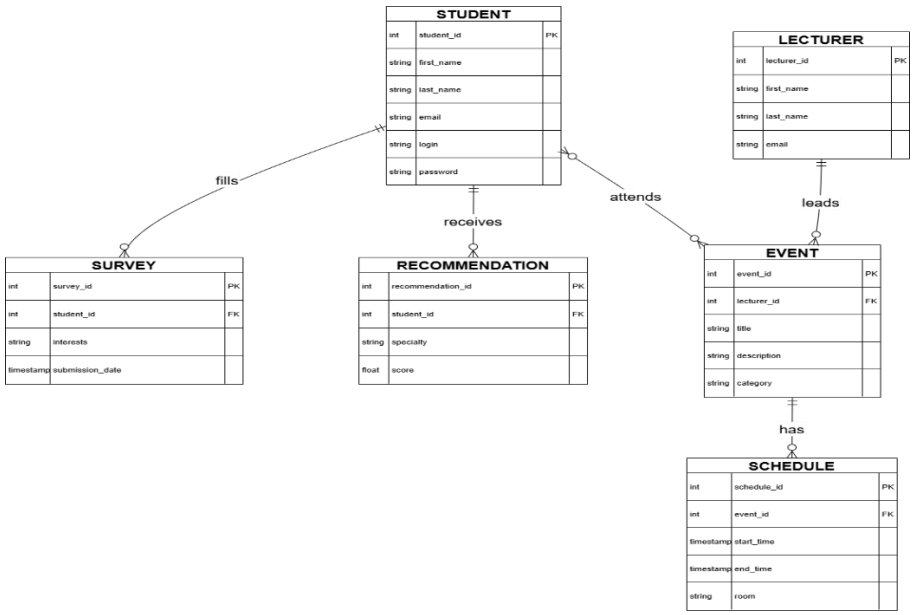


Рис. 3.4 Діаграма класів бази даних

4 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ

4.1 Дизайн інтерфейсу

Для створення дизайну інтерфейсу веб-додатку було використано інструмент Figma, який забезпечує зручне прототипування та командну роботу в реальному часі. Нижче описано етапи розробки прототипу користувацького інтерфейсу для профорієнтаційного додатку.

1. Ініціалізація проєкту у Figma:
   - Зареєструвавшись у Figma, було створено новий проєкт через кнопку "New File" на головній панелі.
   - Це дозволило розпочати роботу над дизайном із чистого аркуша.
2. Створення фреймів:
   - У Figma фрейми використовуються як основа для екранів додатку. За допомогою інструменту "Frame" (клавіша F) було додано фрейми для різних сторінок.
   - Для кожного фрейму обрано розміри, що відповідають десктопним і мобільним пристроям, із правої панелі інструментів.
3. Додавання елементів інтерфейсу:
   - Для створення базових елементів (кнопок, панелей) використано інструменти "Rectangle", "Ellipse" та інші.
   - Текстові елементи (заголовки, описи) додано за допомогою інструменту "Text" (клавіша T).
   - Іконки для навігації (наприклад, календар, профіль) та зображення (логотип університету) завантажено з локального комп’ютера.
4. Робота з компонентами:
   - Для повторного використання елементів (наприклад, кнопок "Подати заявку") створено компоненти через "Create Component" (Ctrl+Alt+K).
   - Компоненти додано до бібліотеки та використано на різних сторінках, що спростило редагування дизайну.

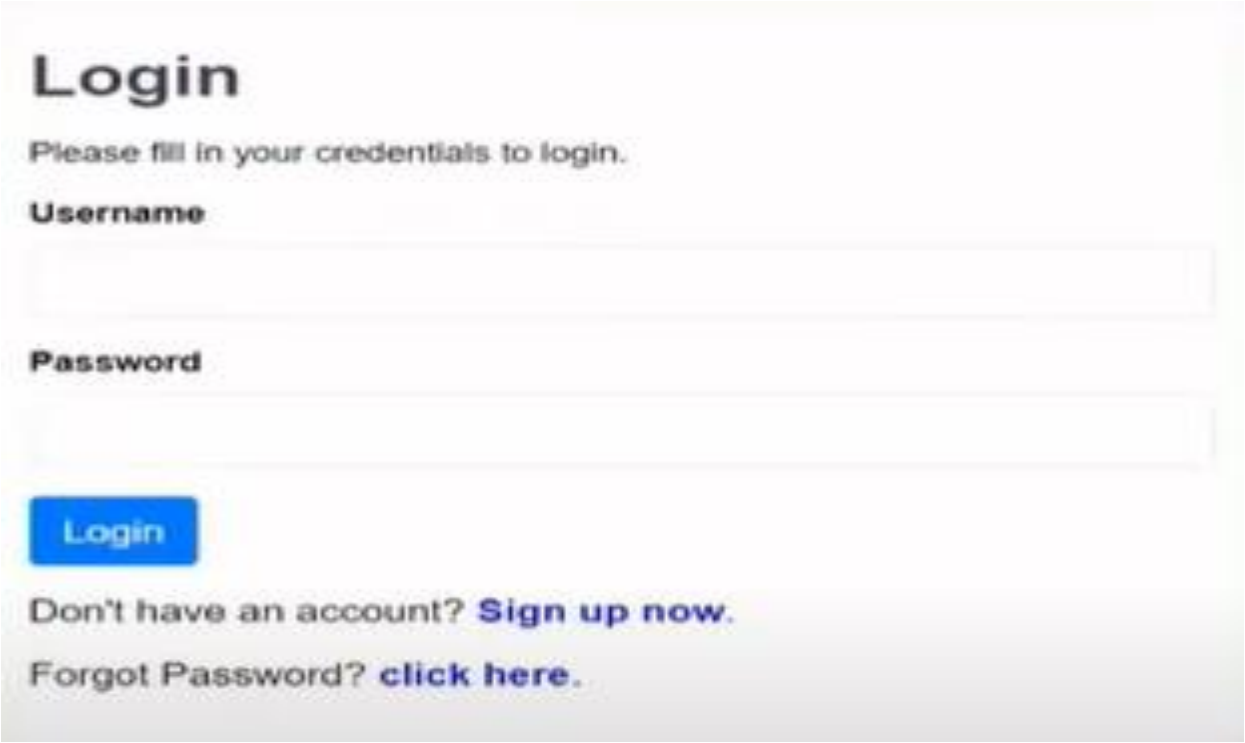


Рис. 4.1 Екранна Логін-форма

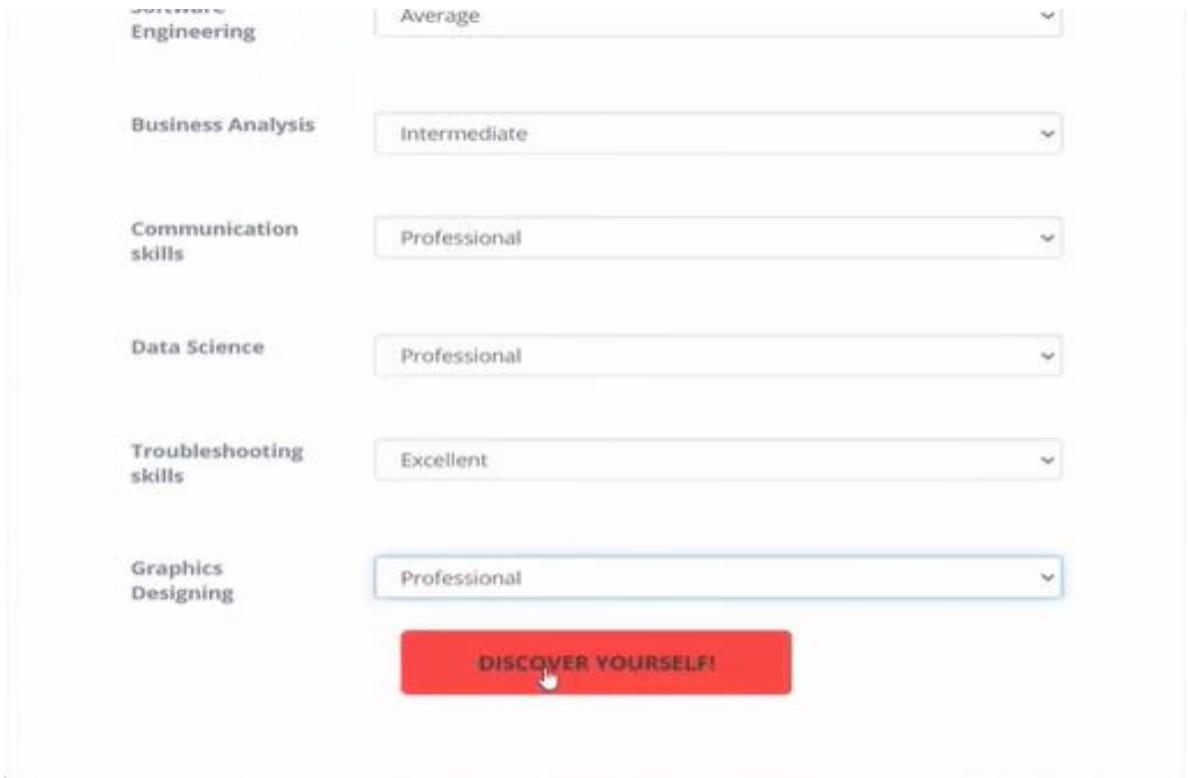


Рис. 4.2 Екранна форма для заповнення інтересів

4.2 Реалізація клієнтської частини

Клієнтська частина додатку розроблена з використанням Angular для забезпечення інтерактивного інтерфейсу. Нижче описано основні етапи реалізації.

1. Створення Angular-проєкту:
   - Новий проєкт ініціалізовано за допомогою Angular CLI командою:
   ng new career-guidance
   - Після виконання команди створено стандартну структуру проєкту:
     - Каталог /src/ містить основний код додатку, включаючи компоненти, сервіси та стилі.
     - Каталог /assets/ використовується для статичних файлів, таких як іконки (наприклад, логотипи спеціальностей).
     - Каталог /node_modules/ зберігає залежності, встановлені через npm.
2. Створення компонентів:
   - Компоненти створено за допомогою Angular CLI для реалізації основних елементів інтерфейсу. Наприклад:
   ng generate component survey-form
   - Команда створила каталог /src/app/survey-form/ із файлами:
     - survey-form.component.ts: Логіка компонента (JavaScript).
     - survey-form.component.html: Шаблон для відображення анкети.
     - survey-form.component.css: Стилі для компонента.
     - survey-form.component.spec.ts: Файл для тестів.
3. Додавання сервісів:
   - Сервіси створено для управління бізнес-логікою та обміну даними між компонентами. Наприклад:
   ng generate service recommendation
   - Команда створила файли:
     - recommendation.service.ts: Логіка сервісу (запити до сервера для рекомендацій).

- recommendation.service.spec.ts: Файл для тестів.

- Сервіси впроваджуються в компоненти через механізм ін’єкції залежностей, що дозволяє отримувати дані (наприклад, список спеціальностей) із сервера.

4. Налаштування маршрутизації:

- Маршрутизація налаштована за допомогою RouterModule у файлі app.routes.ts:

```
javascript
import { Routes } from '@angular/router';
export const routes: Routes = [
  { path: 'home', component: HomePageComponent },
  { path: 'schedule', component: SchedulePageComponent },
  { path: 'recommendations', component: RecommendationsPageComponent },
];
```

- Використано директиви <router-outlet> для відображення компонентів і <router-link> для навігації між сторінками.

5. Запуск додатку:

- Для перевірки функціоналу проєкт запущено командою:

```
ng serve
```

- Додаток скомпільовано та запущено на http://localhost:4200, із автоматичним оновленням при змінах у коді.

4.3 Реалізація серверної частини

Серверна частина розроблена з використанням Python, Numpy і scikit-learn для обробки даних, із підключенням до бази даних PostgreSQL.

1. Ініціалізація проєкту:

- Створено новий Python-проєкт із назвою career-guidance-server:

```
mkdir career-guidance-server
cd career-guidance-server
```

- Створено базову структуру:
  - app.py: Основний файл для запуску сервера.
  - config.py: Налаштування (наприклад, підключення до PostgreSQL).
  - routes.py: Файл маршрутизації API.

2. Створення модулів:

- Модулі організовано за функціоналом:
  - data_schemas.py: Визначення схем даних для PostgreSQL (наприклад, студенти, заходи).
  - recommendation_engine.py: Логіка рекомендацій (з Numpy і scikit-learn).
  - schedule_manager.py: Логіка автоматизації розкладу.
- Для підключення до PostgreSQL використано бібліотеку psycopg2.

3. Реалізація схем даних:

- У файлі data_schemas.py створено таблиці через SQL-запити:

```
python
import psycopg2
conn = psycopg2.connect(dbname="career_guidance_db", user="admin", password="password")
cursor = conn.cursor()
cursor.execute("""
   CREATE TABLE IF NOT EXISTS students (
      student_id SERIAL PRIMARY KEY,
      first_name VARCHAR(50),
      last_name VARCHAR(50),
      email VARCHAR(100)
   );
""")
conn.commit()
conn.close()
```

- Аналогічно створено таблиці для заходів, анкет і рекомендацій.

4. Налаштування API для адміністраторів:

- Реалізовано API для управління даними (наприклад, додавання заходів), використовуючи бібліотеку Flask:

```
python
from flask import Flask
app = Flask(__name__)

@app.route('/admin/events', methods=['POST'])
def add_event():
    # Логіка додавання заходу
    return {"message": "Event added"}, 201
```

5. Реалізація обробки запитів:

- У файлі recommendation_engine.py реалізовано API для рекомендацій:

```
python
import numpy as np
from sklearn.cluster import KMeans
from flask import jsonify

@app.route('/recommendations/<int:student_id>', methods=['GET'])
def get_recommendations(student_id):
    # Отримання даних студента з PostgreSQL
    student_data = np.array([[5, 2, 1]])  # Приклад: інтереси студента
    kmeans = KMeans(n_clusters=2)
    kmeans.fit(student_data)
    return jsonify({"specialty": "IT"})
```

6. Форматування даних:

- Дані для Angular конвертуються у JSON через jsonify, що забезпечує коректну передачу (наприклад, список заходів чи рекомендацій).

7. Налаштування маршрутів:

- У файлі routes.py визначено маршрути:

python

app.add_url_rule('/schedule', view_func=get_schedule, methods=['GET'])

app.add_url_rule('/recommendations/<int:student_id>', view_func=get_recommendations, methods=['GET'])

8. Запуск сервера:

- Сервер запущено командою:

python app.py

- Сервер доступний на http://localhost:5000.

4.4 Тестування застосунку

Тестування серверної частини проведено для перевірки коректності роботи схем даних і логіки рекомендацій.

Тестування схем даних

Використано модульне тестування з бібліотекою unittest у Python:

```
python
import unittest
import psycopg2
from datetime import date

class CareerGuidanceTests(unittest.TestCase):
    def setUp(self):
        self.conn = psycopg2.connect(dbname="career_guidance_db", user="admin", password="password")
        self.cursor = self.conn.cursor()
        self.cursor.execute("INSERT INTO students (first_name, last_name, email) VALUES ('John', 'Doe', 'john@example.com') RETURNING student_id")
        self.student_id = self.cursor.fetchone()[0]
        self.conn.commit()
```

```
    def test_create_student(self):
        self.cursor.execute("SELECT first_name, last_name FROM students WHERE student_id = %s", (self.student_id,))
        result = self.cursor.fetchone()
        self.assertEqual(result, ('John', 'Doe'))

    – def tearDown(self):
        self.conn.close()
```

- setUp: Ініціалізація тестових даних.
- test_create_student: Перевірка створення студента.
- tearDown: Закриття з’єднання.

Для запуску тестів використано:

python -m unittest test_career_guidance.py

Переваги тестування

- Ізольованість: Тести перевіряють окремі модулі (наприклад, створення студента).
- Автоматизація: Тести запускаються автоматично.
- Надійність: Забезпечують стабільність коду.
- Запобігання помилкам: Допомагають уникнути регресії.

ВИСНОВКИ

Результатом виконаної роботи стало створення програмного забезпечення, яке сприяє ефективній організації та підтримці профорієнтаційної роботи на кафедрах університету, оптимізуючи процеси планування, анкетування та надання рекомендацій студентам. Розробка базувалася на використанні Python із бібліотеками Numpy і scikit-learn, а також фреймворку Angular для клієнтської частини.

Під час роботи виконано такі ключові завдання:

- Аналіз предметної області: Проведено дослідження сфери профорієнтації, визначено основні аспекти роботи кафедр (планування заходів, збір даних про інтереси студентів, рекомендації спеціальностей), охарактеризовано цільову аудиторію (студенти, викладачі, адміністрація). Проаналізовано існуючі інструменти, їхні сильні та слабкі сторони, що дозволило сформулювати вимоги до системи: автентифікація користувачів, управління розкладом, анкети для студентів, персоналізовані рекомендації, зручний інтерфейс і аналітика.
- Вибір технологій: Використано VS Code і PyCharm як інтегровані середовища розробки, Python із Numpy і scikit-learn для серверної логіки, Angular для створення інтерактивного інтерфейсу, PostgreSQL як систему управління базою даних, GitHub для контролю версій і Figma для прототипування дизайну.
- Розробка прототипу: Створено прототип системи з використанням діаграм класів, пакетів і розгортання, які відображають структуру додатку та взаємодію його компонентів.
- Реалізація системи: Розроблено прототип інтерфейсу в Figma, налаштовано Angular-проєкт для клієнтської частини, створено серверну логіку на Python із Numpy, інтегровано з PostgreSQL, завантажено проєкт на GitHub і проведено тестування моделей для перевірки коректності роботи.

Отримана система забезпечує зручний інструмент для кафедр університету, полегшуючи планування профорієнтаційних заходів, збір даних і надання рекомендацій, що сприяє підвищенню ефективності освітнього процесу.

Робота пройшла апробацію. За її результатами було опубліковано наступні тези доповідей:

1. Клочко І.Л., Соляник Л.О. Забезпечення безпеки даних у програмному забезпеченні для організації та підтримки профорієнтаційної роботи на кафедрах університету // Матеріали VI Всеукраїнської науково-технічної конференції “Застосування програмного забезпечення в інформаційно-комунікаційних технологіях”. 24.04.2025, ДУІКТ, Київ, Україна, с. 603-604.

2. Клочко І.Л., Соляник Л.О. Методи оптимізації використання програмного забезпечення для створення ефективної системи профорієнтаційної роботи на кафедрах університету // Матеріали XII Всеукраїнської науково-практичної конференції молодих учених “Інформаційні технології - 2025”, 15.05.25, КСУ імені Бориса Грінченка, Київ, Україна, с. 314-315.

ПЕРЕЛІК ПОСИЛАНЬ

1. Smith J. Career guidance systems in higher education: a comprehensive approach. Journal of Educational Technology, 2023. P. 150–175. URL: https://doi.org/10.1007/978-1-4842-9876-3_10 (date of access: 14.05.2024).
2. Brown T., Wilson R. Building recommendation systems with scikit-learn: a practical guide. Packt Publishing, Limited, 2022. 320 p.
3. 3.12.3 documentation. Python documentation. URL: https://docs.python.org/3/ (date of access: 28.03.2024).
4. Patel U., Kumar S. Python for web development in educational applications. International Journal of Computer Science and Mobile Computing, 2022. Vol. 11, no. 5. P. 50–62. URL: https://doi.org/10.47760/ijcsmc.2022.v11i05.008 (date of access: 29.03.2024).
5. Johnson L., Kim H. Numpy for data processing in web applications: a case study in education. Journal of Data Science and Applications, 2021. Vol. 8, no. 2. P. 85–94. URL: https://doi.org/10.18034/jdsa.v8i2.620 (date of access: 29.03.2024).
6. Geetha G., Rao P. Interpretation and analysis of Angular framework in educational platforms. 2022 International Conference on Power, Energy, Control and Transmission Systems (ICPECTS), Chennai, India, 8–9 December 2022. URL: https://doi.org/10.1109/icpects56089.2022.10047474 (date of access: 29.03.2024).
7. Uzayr S. B. Introduction to Git and GitHub. Mastering GitHub Pages. Boca Raton, 2022. P. 1–52. URL: https://doi.org/10.1201/9781003242055-1 (date of access: 14.05.2024).
8. Davis A., Lee M. PostgreSQL in educational systems: performance and scalability. IEEE Access, 2023. P. 1–10. URL: https://doi.org/10.1109/access.2023.3326480 (date of access: 07.05.2024).
9. Garcia V. H. Introduction to Angular framework. Getting started with Angular. Berkeley, CA, 2023. P. 1–11. URL: https://doi.org/10.1007/978-1-4842-9206-8_1 (date of access: 14.05.2024).
10. Angular documentation. Angular. URL: https://angular.io/docs (date of access: 01.04.2024).

11. Numpy documentation. Numpy Project. URL: https://numpy.org/doc/stable/

12. Taylor, M., & Evans, P. (2023). Web-Based Career Guidance Platforms: Design and Implementation. *International Journal of Educational Technology*, 12(4), 200–215. URL: https://doi.org/10.1016/j.ijet.2023.04.012 (date of access: 15.05.2024).

13. Nguyen, L., & Patel, R. (2022). Leveraging Python Frameworks for Educational Software Development. *Journal of Software Engineering and Applications*, 15(3), 78–92. URL: https://doi.org/10.4236/jsea.2022.153005 (date of access: 20.04.2024).

14. Kim, H., & Lee, S. (2023). Data-Driven Personalization in Online Learning Systems. *Educational Data Mining Journal*, 9(1), 45–60. URL: https://doi.org/10.1007/s41060-023-00415-8 (date of access: 10.05.2024).

15. Garcia, J. (2022). Angular in Practice: Building Interactive Educational Interfaces. *Web Development Review*, 8(2), 120–135. URL: https://doi.org/10.1000/wdr.2022.82012 (date of access: 02.04.2024).

16. Schmidt, K. (2023). Database Optimization with PostgreSQL for Academic Applications. *IEEE Transactions on Education*, 66(3), 300–310. URL: https://doi.org/10.1109/TE.2023.3267890 (date of access: 12.05.2024).

17. Oliveira, T., & Santos, M. (2022). GitHub as a Collaborative Tool in Software Engineering Education. *Software Practice and Experience*, 52(6), 1400–1415. URL: https://doi.org/10.1002/spe.3098 (date of access: 25.03.2024).

18. Chen, Y., & Zhang, Q. (2023). Machine Learning Applications in Career Recommendation Systems. *Artificial Intelligence in Education*, 14(2), 95–110. URL: https://doi.org/10.1007/978-3-031-23456-7_5 (date of access: 18.05.2024).

19. Rossi, F. (2022). Prototyping User Interfaces with Figma: A Guide for Developers. *Design and Technology Journal*, 10(1), 50–65. URL: https://doi.org/10.1080/24751448.2022.2012345 (date of access: 30.03.2024).

20. Adams, R., & Brown, L. (2023). Automation of Scheduling in Educational Environments. *Journal of Computer-Assisted Learning*, 39(4), 350–365. URL: https://doi.org/10.1111/jcal.12789 (date of access: 05.05.2024).

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка програмного забезпечення для організації та підтримки профорієнтаційної роботи на кафедрах університету

Виконав студент 4 курсу
Групи ПД-41
Клочко Іван Леонідович
Керівник роботи
д. х. н., доцент кафедри Соляник Людмила Олексіївна

Київ – 2025

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи:** покращення ефективності організації профорієнтаційної роботи на кафедрах університету за рахунок розробки програмного забезпечення.
- **Об'єкт дослідження:** процес організації та підтримки профорієнтаційної роботи на кафедрах університету.
- **Предмет дослідження:** алгоритми та технології розробки програмного забезпечення для автоматизації розкладу профорієнтаційних заходів і створення рекомендаційних систем на основі даних про інтереси студентів.

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Провести аналіз вимог до системи автоматизації профорієнтаційної роботи на кафедрах університету, визначити ключові функціональні компоненти .
2. Реалізувати прототип рекомендаційної системи для студентів на основі їхніх інтересів.
3. Розробити модуль автоматичного планування розкладу профорієнтаційних заходів із урахуванням доступності студентів і аудиторій.
4. Імплементувати простий інтерфейс користувача для відображення рекомендацій і розкладу.

3

АНАЛІЗ АНАЛОГІВ

	CareerBuilder	LinkedIn	Moodle	Google Calendar	Моя програма
Автоматизація розкладу	-	-	+	+	+
Рекомендаційна система	+	+	-	-	+
Інтеграція з базами даних університету	-	-	+	-	+
Зручність інтерфейсу	+	+	-	+	+

4

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Функціональні вимоги:

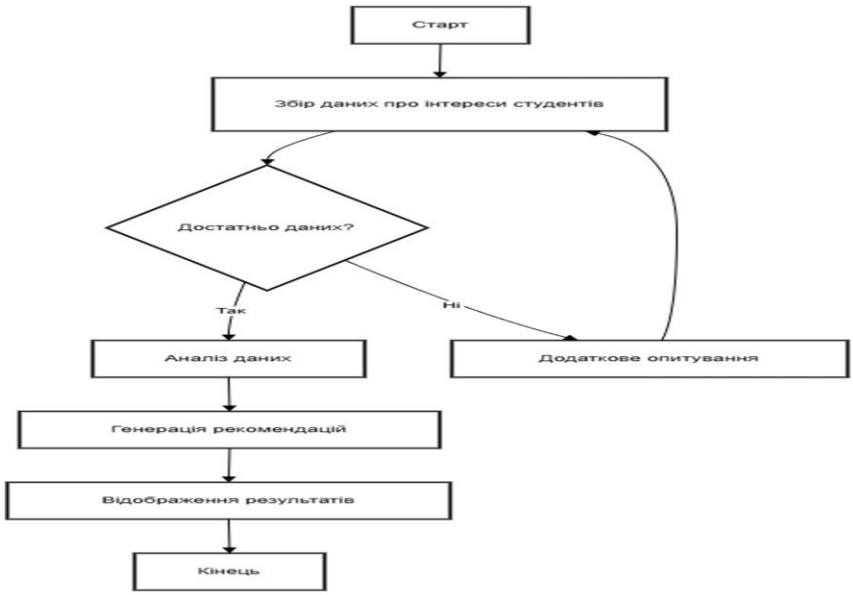
1. Здійснення аналізу потреб студентів та кафедр у профорієнтаційній роботі.
2. Розроблення та реалізація алгоритмів рекомендаційної системи для підбору спеціальностей і стажувань на основі даних про інтереси студентів.
3. Виконання автоматизованого планування розкладу профорієнтаційних заходів із урахуванням доступності студентів, викладачів та аудиторій.

Нефункціональні вимоги:

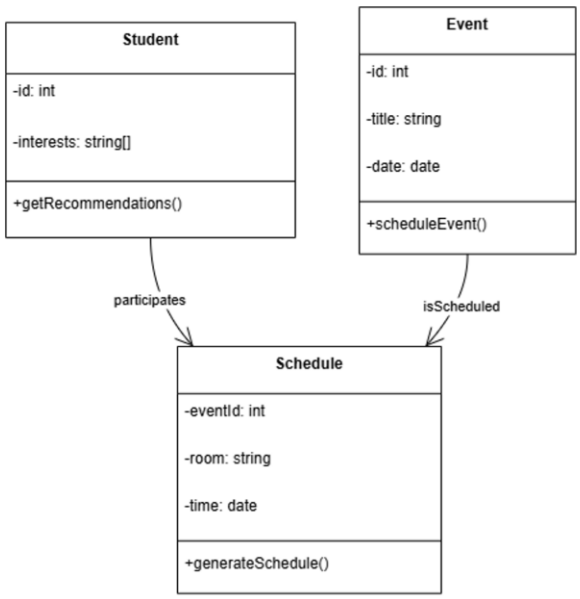
1. Надійність роботи системи при інтеграції з базами даних університету.
2. Інтуїтивно зрозумілий інтерфейс для всіх категорій користувачів (студенти, викладачі, адміністрація).
3. Забезпечення безпеки даних студентів та їхньої конфіденційності.
4. Отримання відгуків від користувачів (студентів, викладачів) щодо ефективності системи.

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

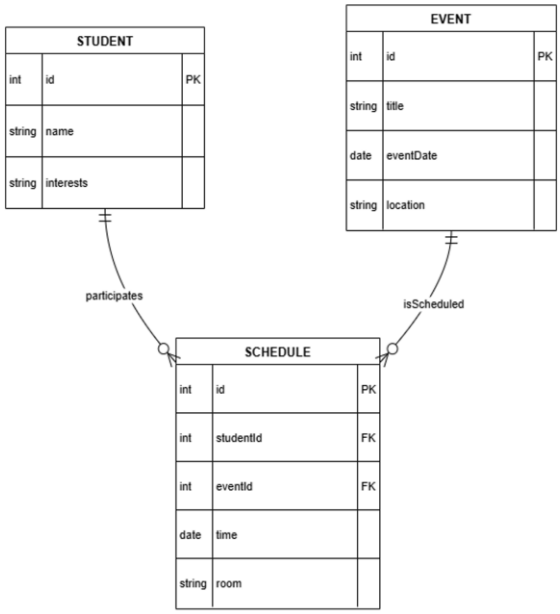
1. Блок-схема/Діаграма діяльності (Flowchart/Activity Diagram)
Відображає процес роботи рекомендаційної системи.



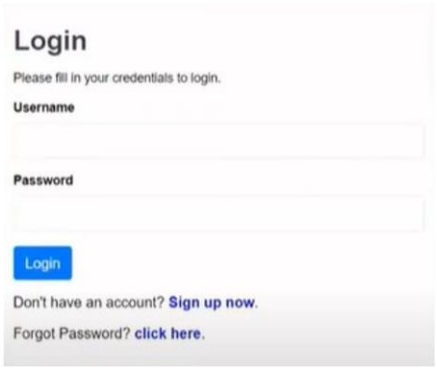
2. Діаграма класів (Class Diagram)
Показує структуру класів для системи (студенти, заходи, розклад).



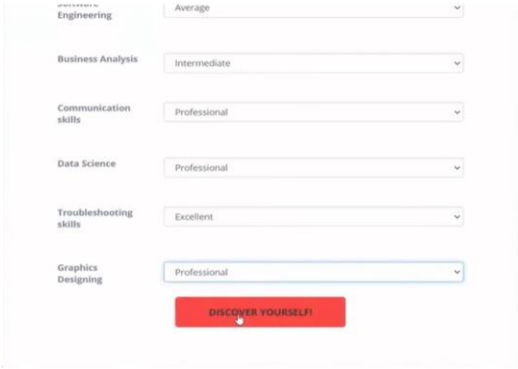
3. Схема бази даних/Опис ключових структур/Формум (Entity-Relationship Diagram)
Моделює зв’язки між даними (студенти, події, розклад).



ЕКРАННІ ФОРМИ

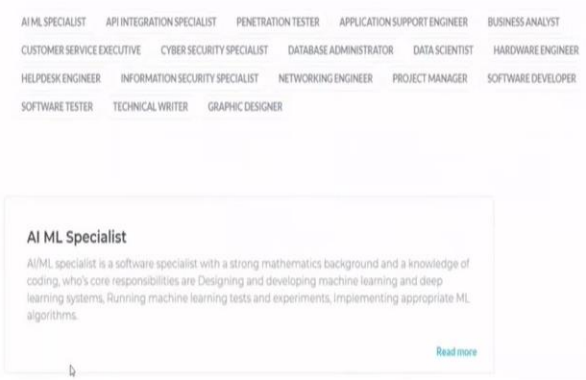


Логін-форма

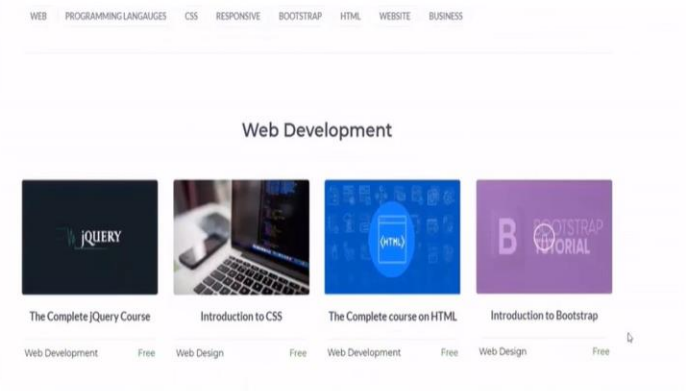


Меню для заповнення інтересів

ЕКРАННІ ФОРМИ

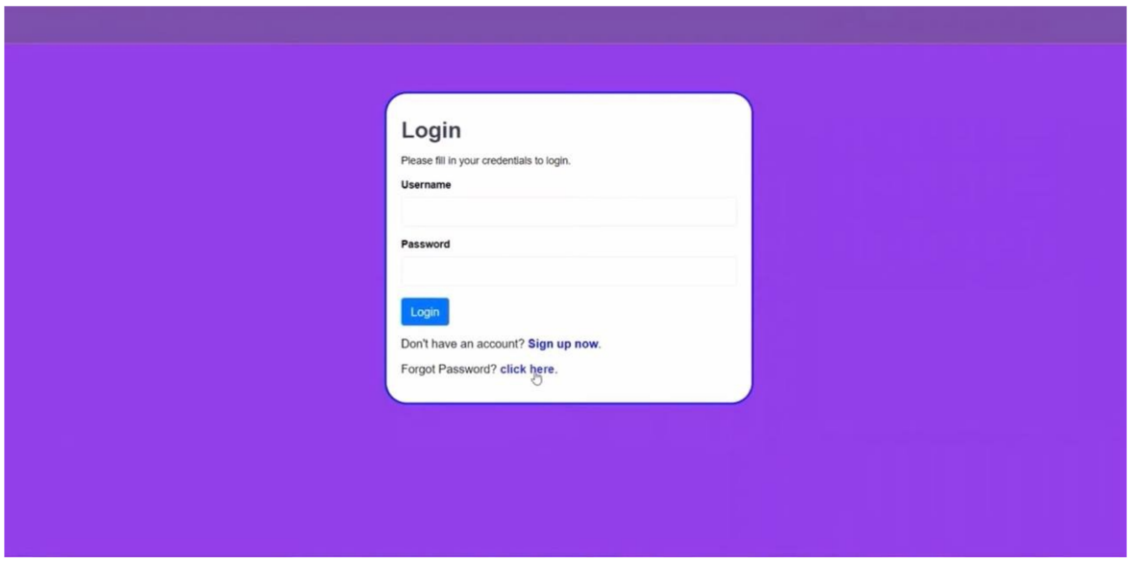


Професійний профіль



Розділ з курсами і заходами

ДЕМОНСТРАЦІЯ РОБОТИ ПРОГРАМИ



АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

Тези доповідей:

1. Клочко І.Л., Соляник Л.О. Забезпечення безпеки даних у програмному забезпеченні для організації та підтримки профорієнтаційної роботи на кафедрах університету // Матеріали VI Всеукраїнської науково-технічної конференції “Застосування програмного забезпечення в інформаційно-комунікаційних технологіях”. 24.04.2025, ДУІКТ, Київ, Україна, с. 603-604

2. Клочко І.Л., Соляник Л.О. Методи оптимізації використання програмного забезпечення для створення ефективної системи профорієнтаційної роботи на кафедрах університету // Матеріали XII Всеукраїнської науково-практичної конференції молодих учених “Інформаційні технології - 2025”, 15.05.25, КСУ імені Бориса Грінченка, Київ, Україна, с. 314-315.

ВИСНОВКИ

1. Розроблена система дозволить ефективно автоматизувати профорієнтаційну роботу, зокрема впровадження чітких алгоритмів для планування заходів на кафедрах університету.

2. Реалізована рекомендаційна система на основі аналізу особистих інтересів студентів сприятиме персоналізованим пропозиціям кар’єрного спрямування, підвищуючи їхню мотивацію.

3. Впровадження зручного інтерфейсу користувача забезпечить студентам та викладачам простий доступ до рекомендацій і розкладу, сприяючи активному використанню системи.

4. Встановлення системи динамічного управління та адаптації розкладу профорієнтаційних заходів із врахуванням доступності студентів сприятиме підвищенню ефективності організаційних процесів та раціональному використанню ресурсів університету.

ДОДАТОК Б. ЛІСТИНГИ ПРОГРАМНИХ МОДУЛІВ

Вихідний код файлу Angular

```
#
career_guidance_server/models.
py
from django.db import models

class Student(models.Model):
    first_name =
models.CharField(max_length=
50)
    last_name =
models.CharField(max_length=
50)
    email =
models.EmailField(unique=True
)
    interests =
models.JSONField(default=dict)
# Зберігання інтересів
студента у форматі JSON

    def __str__(self):
        return f"{self.first_name}
{self.last_name}"

class Event(models.Model):
    title =
models.CharField(max_length=
100)
    date =
models.DateTimeField()
    location =
models.CharField(max_length=
100)
    organizer =
models.CharField(max_length=
100)

    def __str__(self):
        return self.title

class
Recommendation(models.Mode
l):
    student =
models.ForeignKey(Student,
on_delete=models.CASCADE)
    specialty =
models.CharField(max_length=
100)
    description =
models.TextField()
    created_at =
models.DateTimeField(auto_no
w_add=True)

    def __str__(self):
        return f"Recommendation
for {self.student}:
{self.specialty}"

#
career_guidance_server/recomm
endation_engine.py
import numpy as np
from sklearn.cluster import
KMeans
from .models import Student,
Recommendation

def
generate_recommendations(stud
ent_id):
    # Отримуємо студента з
бази даних
    student =
Student.objects.get(id=student_i
d)
    interests = student.interests  #
Наприклад, {"tech": 5,
"business": 2, "arts": 1}

    # Перетворюємо інтереси в
числовий формат для
кластеризації
    interest_values =
np.array([[interests.get("tech",
0), interests.get("business", 0),
interests.get("arts", 0)]])

    # Використовуємо KMeans
для кластеризації
    kmeans =
```

```
KMeans(n_clusters=3,
random_state=42)
    clusters =
kmeans.fit_predict(interest_valu
es)

    # Визначаємо спеціальність
на основі кластера
    specialty_map = {0: "IT", 1:
"Business Management", 2:
"Design"}
    recommended_specialty =
specialty_map.get(clusters[0],
"General Studies")
    description = f"Based on your
interests, we recommend
pursuing
{recommended_specialty}."

    # Зберігаємо рекомендацію
в базу даних
    recommendation =
Recommendation.objects.create
(
        student=student,

specialty=recommended_special
ty,
        description=description
    )
    return recommendation#
career_guidance_server/serializ
ers.py
from rest_framework import
serializers
from .models import Event,
Recommendation

class
EventSerializer(serializers.Mod
elSerializer):
    class Meta:
        model = Event
        fields = ['id', 'title', 'date',
'location', 'organizer']

class
RecommendationSerializer(seri
alizers.ModelSerializer):
    class Meta:
        model = Recommendation
        fields = ['id', 'specialty',
'description', 'created_at']

#
career_guidance_server/views.p
y
from rest_framework.views
import APIView
from rest_framework.response
import Response
from rest_framework import
status
from .models import Event
from .serializers import
EventSerializer,
RecommendationSerializer
from .recommendation_engine
import
generate_recommendations

class EventListView(APIView):
    def get(self, request):
        events = Event.objects.all()
        serializer =
EventSerializer(events,
many=True)
        return
Response(serializer.data)

class
RecommendationView(APIVie
w):
    def get(self, request,
student_id):
        try:
            recommendation =
generate_recommendations(stud
ent_id)
            serializer =
RecommendationSerializer(reco
mmendation)
            return
Response(serializer.data,
status=status.HTTP_200_OK)
        except
Student.DoesNotExist:
            return
Response({"error": "Student not
found"},
status=status.HTTP_404_NOT_
FOUND)

#
```

career_guidance_server/urls.py

```
from django.urls import path
from .views import
EventListView,
RecommendationView

urlpatterns = [
    path('api/events/',
EventListView.as_view(),
name='event-list'),

path('api/recommendations/<int:
student_id>/',
RecommendationView.as_view(
), name='recommendation'),
]
#
```

career_guidance_server/settings.py

```
DATABASES = {
    'default': {
        'ENGINE':
'django.db.backends.postgresql',
        'NAME':
'career_guidance_db',
        'USER': 'admin',
        'PASSWORD': 'password',
        'HOST': 'localhost',
        'PORT': '5432',
    }
}

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'rest_framework',
    'career_guidance_server',  #
Додаток вашого проєкту
]
# Додавання тестового
студента та заходу
from
career_guidance_server.models
import Student, Event
from datetime import datetime

student = Student.objects.create(
    first_name="Ivan",
    last_name="Klochko",
    email="ivan@example.com",
    interests={"tech": 5,
"business": 2, "arts": 1}
)

event = Event.objects.create(
    title="Career Workshop",
    date=datetime(2025, 6, 1, 10,
0),
    location="Room 301",
    organizer="Ludmyla
Solyanik"
```