ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ТЕХНОЛОГІЙ ЦИФРОВОГО РОЗВИТКУ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Е-сервіс навігації професійного розвитку з використанням мови програмування C#»

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

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

_____ Марк СИЧ
(підпис)

Виконав: здобувач вищої освіти групи ТЦР-41

Марк СИЧ

Керівник: Вадим ЧИТУЛЯН
Старший викладач кафедри

Рецензент: _____

Київ 2026

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Технологій цифрового розвитку

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Технології цифрового розвитку»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Технологій цифрового розвитку
_____ Вікторія ЖЕБКА
« ____ » _____ 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Сичу Марку Євгенійовичу

1. Тема кваліфікаційної роботи: «E-сервіс навігації професійного розвитку з використанням мови програмування C#»
керівник кваліфікаційної роботи старший викладач кафедри Вадим ЧИТУЛЯН, затверджені наказом Державного університету інформаційно-комунікаційних технологій від « 20 » лютого 2026 року № 51 .

2. Строк подання кваліфікаційної роботи « 18 » травня 2026 року.

3. Вихідні дані до кваліфікаційної роботи: Відкриті джерела даних; засоби розробки ASP.NET Core 8.0 (C#); Selenium WebDriver для збору даних з ClassCentral та Robota.ua; Cohere AI API для генерації рекомендацій; нормативні матеріали з веб-розробки та патерну MVC.

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

1. Аналіз предметної галузі та постановка задачі.
2. Огляд існуючих систем, вибір технологій та проектування архітектури.
3. Розробка, програмна реалізація, тестування та оцінювання ефективності роботи веб-додатку SynapGrid для автоматизованого підбору

навчальних курсів, актуальних вакансій та генерації кар'єрних рекомендацій користувачам.

5. Перелік графічного матеріалу: *презентація*
   1. Аналіз аналогів.
   2. Вимоги до платформи.
   3. Програмні засоби реалізації.
   4. Діаграма варіантів використання.
   5. Діаграма структури бази даних.
   6. Діаграма структури бази даних(ERD).
   7. Діаграма діяльності
   8. Діаграма архітектури системи MVC
   9. Екранні форми.
   10. Апробація результатів дослідження

6. Дата видачі завдання «_20_» лютого 2026 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури	20.02.26–23.02.26	
2	Аналіз існуючих аналогів	23.02.26–27.02.26	
3	Дослідження програмних засобів	07.03.26–14.03.26	
4	Моделювання об’єкту проектування	23.03.26–28.03.26	
5	Розробка функціоналу за стосунку	07.04.26–11.04.26	
6	Вступ, висновки, реферат	17.04.26–22.04.26	
7	Розробка презентації за стосунку	05.05.26–09.05.26	
8	Попередній захист роботи	18.05.26	

Здобувач(ка) вищої освіти _____ Марк СИЧ
(підпис)

Керівник
кваліфікаційної роботи _____ Вадим ЧИТУЛЯН
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 62 стор., 8 табл., 12 рис., 22 джерел.

Мета роботи – автоматизація процесу планування професійного розвитку користувачів за допомогою розробленого е-сервісу навігації професійного розвитку та його інтеграції з платформами онлайн-навчання та пошуку роботи.

Об'єкт дослідження – процес планування індивідуального кар'єрного розвитку фахівців у сфері інформаційних технологій з урахуванням їхніх навичок, цілей та актуального стану ринку праці.

Предмет дослідження – програмне забезпечення, алгоритмічні та інтеграційні рішення для функціонування автоматизованої системи кар'єрного планування на основі веб-скрапінгу даних із платформ ClassCentral.com і Robota.ua та генерації персоналізованих рекомендацій за допомогою Cohere AI API.

Короткий зміст роботи: У роботі розглянуто сучасні методи автоматизації процесу планування кар'єрного розвитку фахівців у сфері інформаційних технологій. Проведено аналіз існуючих платформ онлайн-освіти та працевлаштування, визначено їх сильні сторони та недоліки з точки зору формування цілісних освітньо-професійних траєкторій. Спроектовано та реалізовано архітектуру веб-застосунку, що забезпечує автоматизований підбір навчальних матеріалів і актуальних пропозицій зайнятості відповідно до індивідуального профілю користувача. Розроблено реляційну базу даних для збереження користувацьких даних та сформованих рекомендацій. Реалізацію серверної частини виконано засобами ASP.NET Core та Entity Framework Core, а збір інформації з платформ ClassCentral і Robota.ua забезпечується модулями веб-скрапінгу на основі Selenium WebDriver. Генерацію персоналізованих кар'єрних планів реалізовано шляхом інтеграції з великою мовною моделлю Cohere AI API. За результатами тестування підтверджено коректну роботу сервісів парсингу, кросбраузерну сумісність та адаптивність інтерфейсу на пристроях з різною роздільною здатністю екрана.

Сфера використання: автоматизація побудови індивідуальних кар'єрних траєкторій для IT-фахівців.

КЛЮЧОВІ СЛОВА: ВЕБ-ДОДАТОК, ПЛАНУВАННЯ КАР’ЄРИ, ASP.NET CORE, MVC, ВЕБ-СКРАПІНГ, SELENIUM, ШТУЧНИЙ ІНТЕЛЕКТ, ПЕРСОНА–ЛІЗАЦІЯ, АВТОМАТИЗАЦІЯ, ENTITY FRAMEWORK, CLASSCENTRAL, ROBOTA.UA, ВЕБ-СЕРВІС, ТРАЄКТОРІЯ

ЗМІСТ

ВСТУП

Актуальність. В умовах динамічного розвитку IT-галузі та швидкозмінних вимог роботодавців безперервне навчання перетворилося на обов'язкову складову успішної кар'єри фахівця. Водночас самостійне планування професійного розвитку супроводжується проблемою надмірного інформаційного навантаження. Ручний пошук актуальних вакансій та підбір навчальних курсів із розрізнених джерел є трудомістким процесом, що характеризується високою суб'єктивністю та не забезпечує побудови раціональної траєкторії зростання. Наявний на ринку розрив між змістом освітніх програм і практичними компетенціями, які реально потребують IT-компанії, ще більше ускладнює цей процес. У зв'язку з цим постає потреба у створенні комплексного веб-застосунку, який засобами автоматизованого збору даних та інструментів штучного інтелекту здатний усунути інформаційну роз'єднаність між освітнім і кар'єрним середовищами та забезпечити формування персоналізованих планів професійного розвитку відповідно до актуального попиту ринку праці.

Об'єкт дослідження – процес планування індивідуального кар'єрного розвитку фахівців у сфері інформаційних технологій з урахуванням їхніх навичок, цілей та актуального стану ринку праці.

Предмет дослідження – програмне забезпечення, алгоритмічні та інтеграційні рішення для функціонування автоматизованої системи кар'єрного планування на основі веб-скрапінгу даних із платформ ClassCentral.com і Robota.ua та генерації персоналізованих рекомендацій за допомогою Cohere AI API.

Мета роботи – автоматизація процесу планування професійного розвитку користувачів за допомогою розробленого е-сервісу навігації професійного розвитку та його інтеграції з платформами онлайн-навчання та пошуку роботи.

Для досягнення поставленої мети в бакалаврській роботі виконано наступні задачі:

1. Здійснити аналіз предметної області та провести огляд існуючих програмних рішень у сфері планування професійного розвитку.

2. Вивчити сучасні підходи до автоматизованого збору даних із зовнішніх джерел та методи інтеграції сторонніх інформаційних ресурсів.

3. Розробити архітектуру веб-застосунку на базі архітектурного патерну MVC та спроектувати структуру реляційної бази даних.

4. Реалізувати програмні модулі веб-скрапінгу для автоматизованого отримання актуальних даних із платформ ClassCentral та Robota.ua.

5. Забезпечити інтеграцію системи з великою мовною моделлю Cohere AI API з метою генерації персоналізованих кар'єрних рекомендацій для користувача.

6. Розробити адаптивний інтерфейс користувача з урахуванням вимог зручності та доступності на різних пристроях.

7. Виконати комплексне тестування розробленої системи та верифікацію коректності її функціонування.

Методи дослідження: методи системного та порівняльного аналізу предметної області й існуючих програмних рішень; методи об'єктно-орієнтованого проєктування та архітектурного моделювання з використанням патерну MVC; технології автоматизованого збору даних у режимі реального часу (Web Scraping); методи обробки природної мови та семантичного аналізу текстових даних на основі великих мовних моделей (LLM); практичні методи розробки, модульного та інтеграційного тестування вебзастосунків на платформі ASP.NET Core.

Наукова новизна отриманих результатів полягає в удосконаленні підходу до планування кар'єри через інтеграцію технологій автоматизованого збору децентралізованих даних із семантичним аналізом компетенцій користувача на основі моделей LLM. Це дозволяє забезпечити автоматичну генерацію адаптивної траєкторії професійного розвитку в межах єдиної інформаційної системи.

Практичне значення отриманих результатів: розроблений інтелектуальний вебзастосунок автоматизує процес моніторингу ринку праці та освітніх послуг, формуючи персоналізовану карту кар'єрного зростання. Результати роботи можуть

бути використані кар'єрними консультантами та HR-фахівцями для побудови індивідуальних планів розвитку спеціалістів.

Апробація:

1. Сич М. Є., Читулян В. О. Веб-додаток для автоматизованого планування кар’єрного розвитку в ІТ-сфері з використанням штучного інтелекту // VII Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в інформаційно-комунікаційних технолоіях». 29.04.2026 ДУІКТ, Київ, С.97-100

2. Сич М. Є., Читулян В. О. Веборієнтована система планування карєрного розвитку ІТ-фахівців з інтеграцією штучного інтелекту // ІІІ Всеукраїнська науково-практична конференція «Цифрова гуманістика: Інформаційні технології та інформаційне моделювання на сучасному етапі розвитку суспільства». 05.05.2026, ЦДУ ім. В. Винниченка, м. Кропивницький, С.147-148

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Характеристика предметної галузі

Сфера інформаційних технологій належить до найбільш динамічно змінюваних секторів сучасної економіки. Її характерною особливістю є надзвичайно швидка поява нових інструментів, мов програмування та архітектурних підходів, що висуває постійно зростаючі вимоги до рівня кваліфікації фахівців. За таких умов традиційна модель статичної освіти поступово втрачає свою ефективність, поступаючись місцем концепції безперервного навчання впродовж усього професійного життя [20].

Сучасний стан IT-ринку праці, за даними провідних профільних ресурсів, таких як DOU та Robota.ua, демонструє стрімку зміну попиту на технічні стеки, що вимагає від спеціалістів миттєвої адаптації [12, 13]. Процес самостійного планування кар'єри ускладнюється явищем інформаційного перевантаження. Згідно зі звітами ClassCentral, кількість онлайн-курсів та освітніх програм (MOOC) зростає експоненціально, проте якість та релевантність цього контенту часто не відповідають очікуванням роботодавців [14]. Це породжує відчутний розрив (skills gap) між теоретичною підготовкою вчорашніх випускників та практичними потребами індустрії.

Іншою критичною проблемою є відсутність індивідуалізації в існуючих EdTech-рішеннях. Більшість платформ пропонують ідентичні траєкторії навчання для користувачів із принципово різним базовим рівнем підготовки, що суттєво знижує результативність навчального процесу.

Предметна область дослідження охоплює процеси інтелектуального аналізу даних у контексті HR-технологій. Автоматизація цих процесів дозволяє перетворити суб’єктивний пошук можливостей на чітко структуровану траєкторію професійного зростання. Розробка інтелектуальних систем, здатних зіставляти наявні компетенції користувача з актуальними вимогами ринку за допомогою

семантичного аналізу, є одним із пріоритетних напрямів сучасної інженерії програмного забезпечення [21].

1.2 Аналіз існуючих систем та аналогів

З метою обґрунтування доцільності розробки власного програмного рішення було проведено аналіз функціональних можливостей найбільш поширених платформ у сфері онлайн-освіти та автоматизованого планування навчальних траєкторій. Зокрема, у ході дослідження було проаналізовано систему професійного навчання LinkedIn Learning, освітню платформу Coursera, агрегатор онлайн-курсів ClassCentral [17], а також сервіс візуалізації освітніх маршрутів Roadmap.sh. Основним критерієм аналізу було дослідження механізмів підбору навчального контенту та можливостей персоналізації графіків навчання відповідно до запитів користувача.

1.2.1 Аналіз платформи LinkedIn Learning

LinkedIn Learning – це глобальна освітня платформа, що інтегрована в професійну соціальну мережу LinkedIn та орієнтована на розвиток навичок у сферах технологій, бізнесу та креативу. Web-сайт сервісу виконаний у сучасному мінімалістичному стилі з акцентом на зручну навігацію по великій бібліотеці контенту. Каталог курсів представлено у вигляді тематичних категорій, що містять базову фільтрацію за рівнем складності, тривалістю та типом програм.

Для вирішення завдань, пов'язаних із автоматизованою генерацією персоналізованої освітньої траєкторії на основі інтелектуального аналізу ринку, дана платформа функціонально не підходить. Вона не підтримує вільне введення текстових запитів для формування дорожньої карти «з нуля», не реалізує алгоритмів глибокого семантичного аналізу потреб користувача за допомогою LLM (Large Language Models) і не пропонує механізмів зіставлення навчального контенту з динамічними вимогами локальних вакансій у реальному часі.

Окремо варто зазначити, що на сайті відсутня функція агрегації знань із зовнішніх децентралізованих джерел. Система пропонує лише внутрішній контент платформи, що обмежує користувача у виборі безкоштовних альтернатив (наприклад, курсів з YouTube або репозиторіїв GitHub). Користувач змушений самостійно оцінювати актуальність запропонованого матеріалу, оскільки платформа не надає порівняльної аналітики щодо затребуваності конкретних навичок на ринку праці.

Також відсутні можливості гнучкого збереження та редагування згенерованих маршрутів навчання з прив'язкою до конкретних дедлайнів користувача. Взаємодія з платформою має односторонній характер: вибір із запропонованого списку замість інтерактивного діалогу з інтелектуальним асистентом.

До переваг платформи LinkedIn Learning можна віднести:

– високу якість освітнього відеоконтенту від перевірених експертів;

– автоматичне відображення отриманих сертифікатів у професійному профілі;

– зручну інтеграцію з мобільними пристроями.

Недоліки, що впливають на ефективність для завдань, що вирішуються в межах бакалаврської роботи:

– відсутність будь-якої персоналізації процесу побудови карти знань за допомогою ШІ;

– неможливість агрегації безкоштовних навчальних матеріалів із зовнішніх мереж;

– закритість екосистеми та висока вартість платної підписки;

– відсутність зв'язку між навчальним процесом та динамікою локального IT-ринку.

Приклад каталогу курсів та інтерфейс перегляду навчального контенту зображено на рисунках 1.1 та 1.2 відповідно.

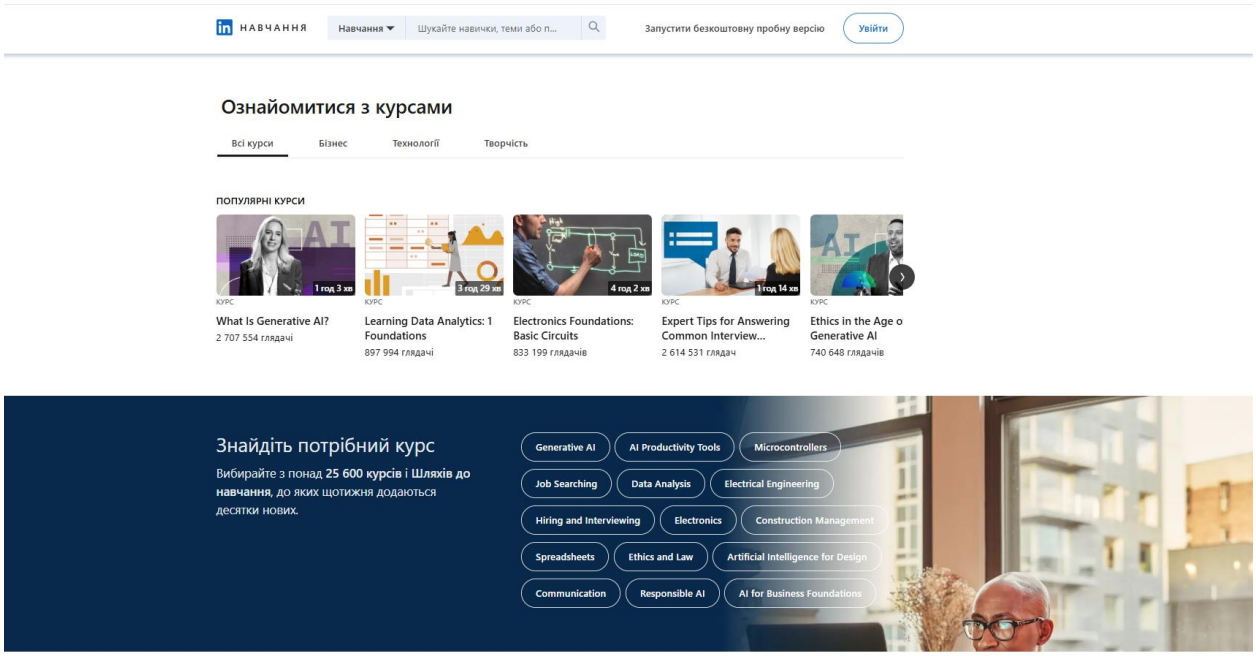


Рис. 1.1 Головна сторінка каталогу курсів на платформі LinkedIn Learning

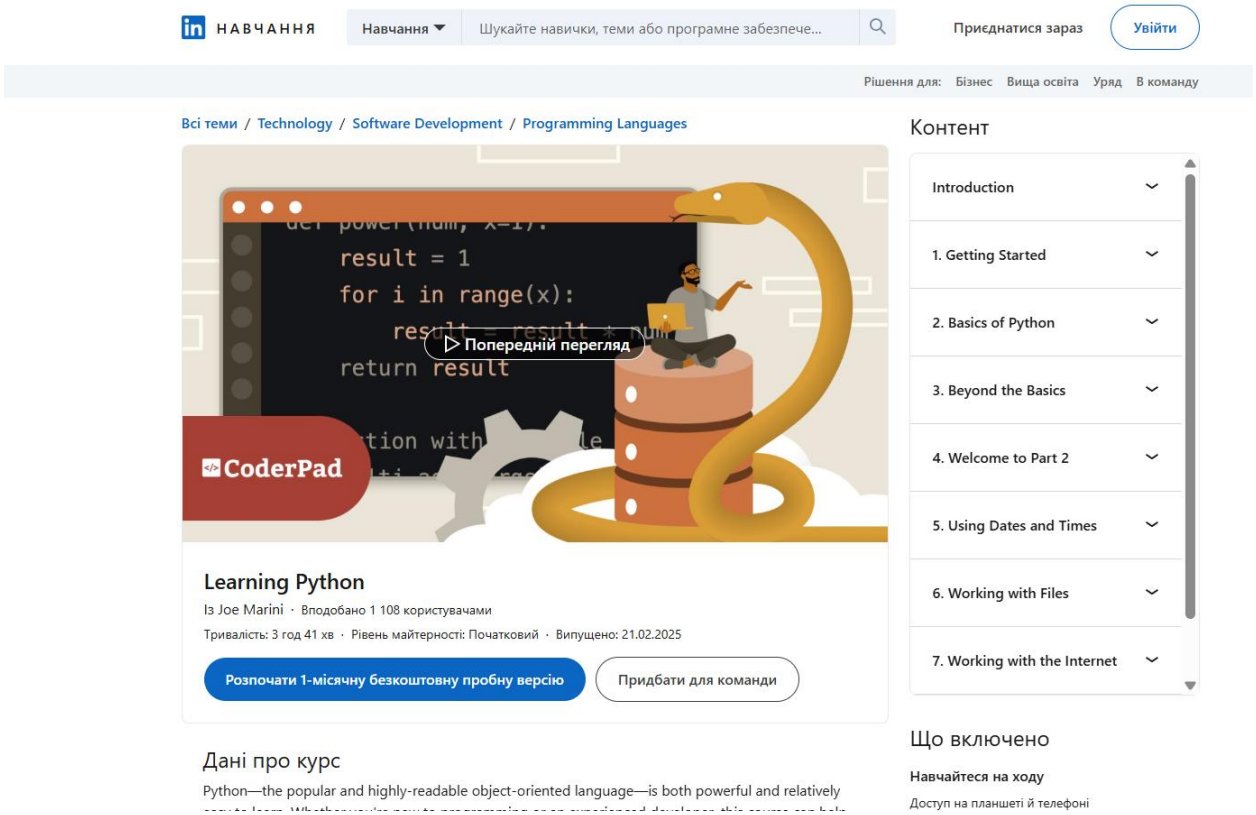


Рис. 1.2 Інтерфейс перегляду навчального контенту на платформі LinkedIn Learning

1.2.2 Аналіз платформи Coursera

Coursera є однією з найбільших світових платформ у сфері масових відкритих онлайн-курсів (MOOC), що співпрацює з провідними університетами (Stanford, Yale) та технологічними гігантами (Google, IBM). Веб-ресурс орієнтований на надання академічної освіти та професійної сертифікації. Інтерфейс платформи побудований за принципом класичного освітнього порталу з розгалуженою системою фільтрації за мовами, рівнем складності та типами навчальних програм.

Для вирішення завдань інтелектуальної генерації індивідуальних освітніх траєкторій дана система функціонально не підходить у повному обсязі. Головною проблемою Coursera є її статичність та ізольованість. Платформа не реалізує інструменти автоматизованого аналізу довільних запитів користувача за допомогою LLM-моделей для створення маршруту «з нуля». Користувач обмежений виключно тими курсами та спеціалізаціями, які вже заздалегідь сформовані партнерами платформи.

Окремо слід виділити відсутність механізмів агрегації знань із зовнішніх децентралізованих мереж. Система пропонує лише власний контент, що робить неможливим залучення безкоштовних актуальних матеріалів із відкритих джерел (YouTube, GitHub, технічна документація). Платформа не проводить автоматичного зіставлення навчальних модулів із динамічними вимогами вакансій у реальному часі, що змушує студента самостійно оцінювати релевантність обраного курсу для майбутнього працевлаштування.

Також відсутня можливість інтерактивного діалогу для коригування плану навчання залежно від наявних проміжних знань користувача. Взаємодія зводиться до лінійного проходження заздалегідь визначених етапів.

До переваг платформи Coursera належать:

– високий рівень академічної експертизи та офіційні сертифікати;
– глибоке опрацювання теоретичного матеріалу;
– можливість отримання фінансової допомоги для навчання.

Недоліки:

– відсутність ШІ-інструментів для побудови персоналізованих дорожніх карт;

– повна відсутність агрегації контенту із зовнішніх джерел;

– статична структура курсів, що не адаптується під індивідуальний запит;

– висока вартість доступу до практичних завдань та іспитів.

Приклади інтерфейсу пошуку курсів та структури освітньої програми представлено на рисунках 1.3 та 1.4 відповідно.

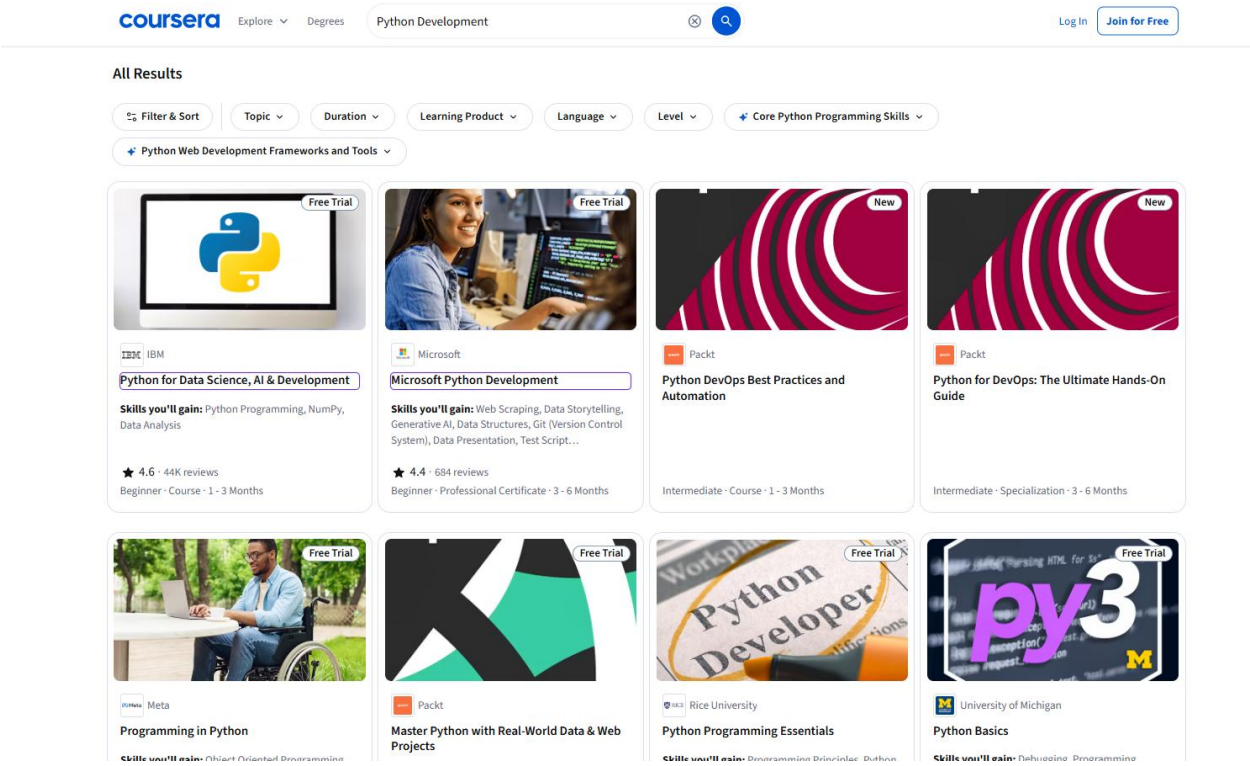


Рис. 1.3 Результати пошуку навчальних програм на платформі Coursera

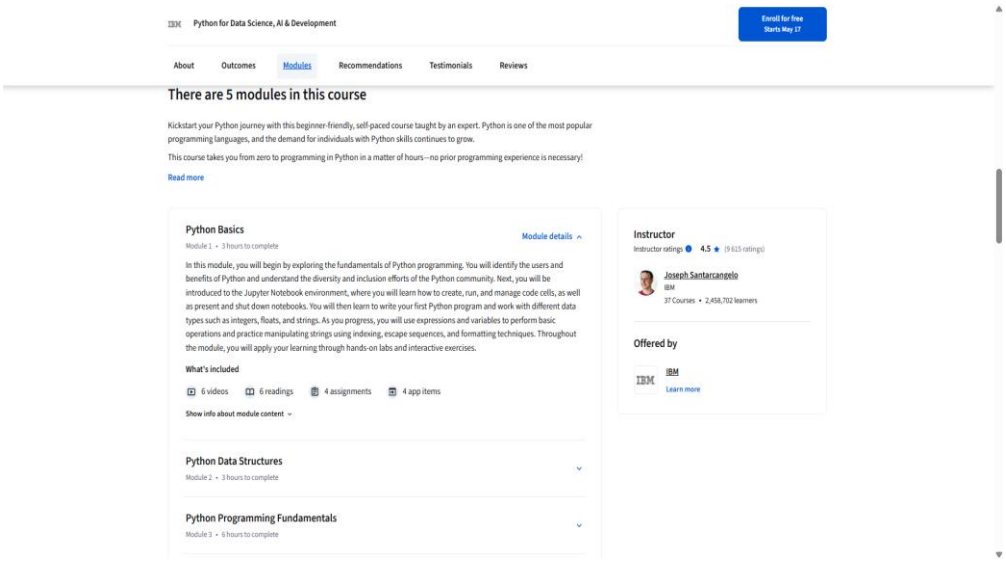


Рис. 1.4 Приклад статичної структури навчального курсу (Syllabus) на платформі Coursera

1.2.3 Аналіз сервісу Roadmap.sh

Roadmap.sh – це популярна освітня платформа з відкритим вихідним кодом, що надає детальні візуальні карти розвитку (дорожні карти) для різних IT-спеціальностей та технологічних стеків. Ресурс побудований за принципом ком'юніті-керованого контенту, де експерти структурують знання, необхідні для опанування певної ролі (наприклад, Backend Developer або ASP.NET Core Developer). Web-сайт сервісу вирізняється високою якістю графічної візуалізації та чіткою логічною ієрархією тем.

Для вирішення завдань, пов'язаних із автоматизованою побудовою індивідуального плану навчання на основі інтелектуального аналізу потреб користувача, дана платформа функціонально не підходить. Вона надає «універсальний» шлях розвитку, який є ідентичним для всіх користувачів незалежно від їхнього попереднього досвіду чи конкретних прогалин у знаннях. Взаємодія з платформою носить статичний характер: користувач може лише переглядати готові схеми, але не має можливості згенерувати унікальний маршрут за допомогою довільного текстового запиту.

Окремо варто зазначити, що Roadmap.sh не використовує алгоритми ШІ для динамічної адаптації контенту. Система не вміє автоматично аналізувати поточний рівень підготовки користувача (через анкетування чи тестування) і не пропонує механізмів виключення вже засвоєних тем із загальної карти. Також на платформі відсутня функція автоматизованої агрегації актуальних навчальних матеріалів (відеокурсів, статей) із зовнішніх децентралізованих джерел у реальному часі — посилання на ресурси додаються модераторами вручну, що сповільнює процес оновлення бази знань.

Крім того, сервіс не має прямої інтеграції з ринком вакансій, тому запропоновані дорожні карти не завжди відображають специфічні вимоги локальних роботодавців, які можуть змінюватися швидше, ніж оновлюється статична схема на сайті.

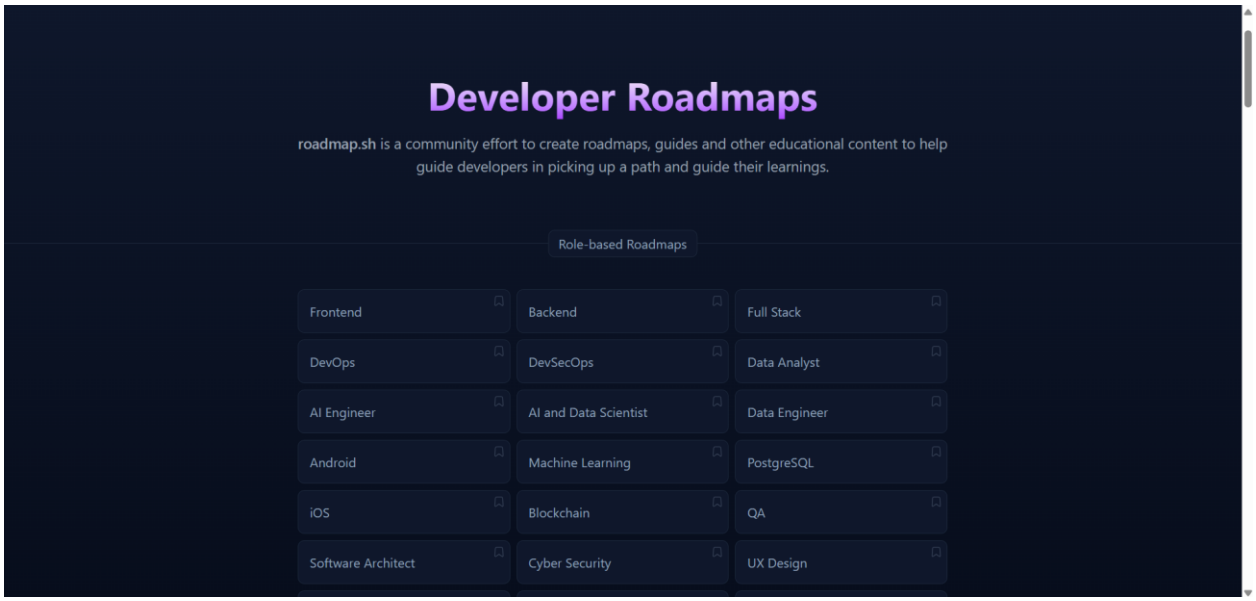


Рис. 1.5 Перелік професійних дорожніх карт на платформі Roadmap.sh

До переваг сервісу Roadmap.sh можна віднести:

– еталонну візуалізацію та структурування технічних навичок;
– широке охоплення сучасних технологій та інструментів розробки;
– наявність детальних текстових пояснень до кожного вузла карти.

Недоліки, що впливають на ефективність для завдань, що вирішуються в межах бакалаврської роботи:

– статичність дорожніх карт та відсутність персоналізації під конкретного користувача;

– відсутність ШІ-інструментів для автоматичної генерації індивідуальних траєкторій навчання;

– відсутність динамічного зв'язку з актуальними вакансіями та зовнішніми освітніми агрегаторами;

– неможливість інтерактивного коригування плану навчання у формі діалогу.

Перелік доступних напрямів навчання та приклад візуалізації дорожньої карти розвитку представлено на рисунках 1.5 та 1.6 відповідно.

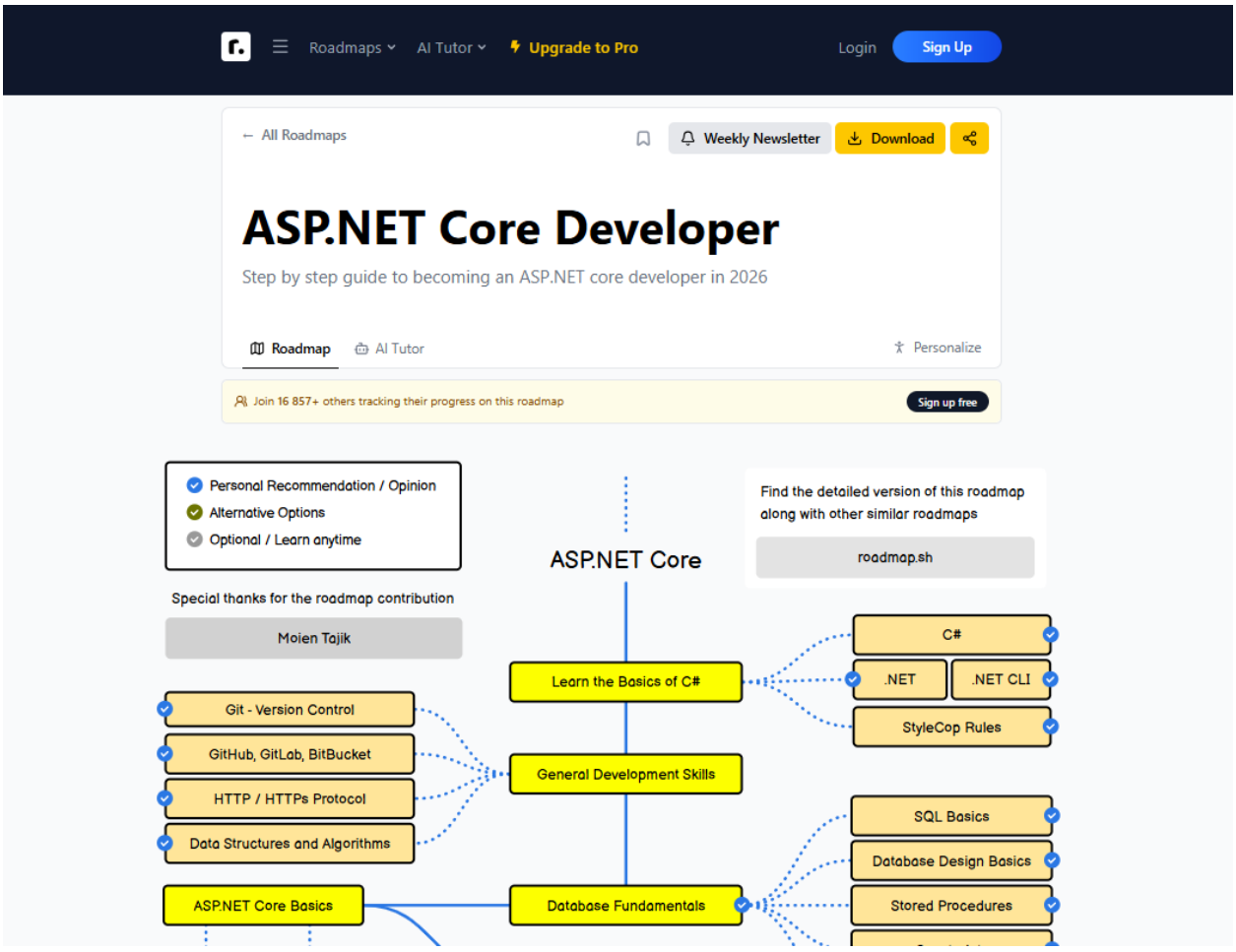


Рис. 1.6 Візуалізація статичної карти розвитку для ASP.NET Core розробника

Таблиця 1.1

Порівняльний аналіз існуючих платформ онлайн-навчання

Функціональні показники	LinkedIn Learning	Coursera	Roadmap.sh
Наявність адаптивного веб-інтерфейсу	+	+	+
Каталогізація контенту за категоріями	+	+	-
Візуалізація навчання у вигляді дорожніх карт	+/-	-	+
Агрегація даних із децентралізованих джерел	-	-	-
Наявність вбудованого ШІ-асистента	+/-	+/-	-
Генерація персоналізованого плану "з нуля"	-	+/-	+/-
Семантичний аналіз довільних запитів (LLM)	-	-	-
Динамічна адаптація карти під рівень знань	-	-	-

1.3 Виявлення актуальних проблем у сфері EdTech та кар'єрного планування

У процесі аналізу предметної галузі та вивчення особливостей існуючих платформ, що реалізують онлайн-навчання та побудову дорожніх карт розвитку, було виявлено низку типових проблем, які стримують розвиток персоналізованих сервісів у сфері EdTech та знижують якість користувацького досвіду.

Насамперед, однією з головних проблем є відсутність інтелектуального підбору навчального контенту на основі реального рівня знань користувача. Переважна більшість освітніх сервісів пропонує лише базову фільтрацію за категоріями або складністю, без урахування специфічних прогалин у підготовці чи попереднього досвіду, хоча саме індивідуальний бекграунд відіграє ключову роль

у виборі ефективного темпу навчання. Врахування початкових компетенцій дозволяє підвищити ймовірність того, що сформований маршрут буде відповідати очікуванням користувача як за насиченістю, так і за часовими витратами.

Іншою важливою проблемою є ігнорування децентралізованого характеру сучасної освіти під час формування навчальних планів. Актуальні знання (технічна документація, репозиторії GitHub, відкриті лекції на YouTube) оновлюються значно швидше за статичні курси, однак ці характеристики практично не використовуються в існуючих web-системах. Натомість більшість сервісів функціонують як закриті екосистеми, що обмежують користувача пропозиціями одного провайдера, ігноруючи динамічні зміни в технологічних стеках.

Також спостерігається відсутність комплексної системи персоналізації для складних запитів. Наприклад, лише обмежена кількість платформ реалізує механізми рекомендацій, але й там немає врахування семантичного зв'язку між різними технологіями або вимогами конкретних вакансій. У більшості випадків структура навчання подається у вигляді статичної таблиці або списку, що потребує ручного аналізу та самостійного пошуку матеріалів з боку користувача, що часто призводить до помилок у виборі пріоритетів.

Крім цього, до ключових проблем галузі належать:

– відсутність інтерактивної взаємодії з користувачем: немає інтелектуальних помічників, здатних інтерпретувати запити у довільній формі та динамічно коригувати план;

– відсутність накопичення персональних даних про прогрес для подальшого аналізу та адаптації траєкторії в реальному часі;

– низький рівень гнучкості та адаптивності дорожніх карт – навіть розширені схеми не підтримують складні запити (наприклад, «перехід з мови Java на Rust з фокусом на високопродуктивні системи»).

Отже, можна виділити основні напрями, які потребують вирішення в межах розробки нового web-застосунку:

– реалізація механізму автоматичного формування ієрархічної структури знань на основі семантичного аналізу запиту;

– агрегація освітнього контенту з децентралізованих відкритих джерел для забезпечення актуальності матеріалів;

– поєднання інструментів візуалізації (дорожніх карт) з інтелектуальним підбором ресурсів;

– підтримка особистого профілю користувача з історією генерацій та можливістю збереження прогресу.

Ці проблеми і визначають обґрунтовану потребу в розробці персоналізованої системи, яка забезпечуватиме якісно новий рівень побудови освітніх траєкторій в онлайн-середовищі.

1.4 Постановка задачі та визначення вимог до розроблюваної системи

На основі проведеного аналізу предметної галузі, дослідження наукових джерел та результатів порівняльного аналізу існуючих програмних рішень, було виявлено низку критичних недоліків, які на сьогодні не вирішуються у межах більшості популярних освітніх платформ. Основні проблеми стосуються низького рівня адаптивності навчальних планів, відсутності механізмів інтелектуального розбору запитів користувача у довільній формі та ізольованості контенту в межах одного провайдера.

У межах цієї кваліфікаційної роботи ставиться завдання розробити персоналізовану веб-платформу для інтелектуальної генерації освітніх траєкторій, яка б забезпечувала автоматизовану побудову дорожніх карт (roadmaps) на основі семантичного аналізу потреб користувача та агрегації актуальних даних із відкритих децентралізованих джерел.

Для реалізації поставленої задачі необхідно виконати такі етапи розробки:

– реалізація інтелектуального модуля для інтерпретації природномовних запитів користувача з використанням сучасних LLM-моделей (зокрема Cohere API або OpenAI API);

– розробка алгоритму автоматизованого збору та фільтрації освітнього контенту з відкритих джерел (YouTube API, GitHub) для динамічного наповнення вузлів навчальної карти;

– створення системи візуалізації ієрархічних структур знань у вигляді інтерактивного графа, що дозволяє користувачеві наочно відстежувати маршрут навчання;

– реалізація функціоналу особистого кабінету з можливістю збереження згенерованих траєкторій, відстеження прогресу та керування обраними ресурсами;

– проектування та реалізація архітектури серверної частини застосунку, що забезпечуватиме стабільну обробку складних запитів та безпечне зберігання даних;

– розробка адміністративної панелі для моніторингу системних логів, керування інтеграціями з API та налаштування параметрів генерації;

– використання сучасного технологічного стеку: мови програмування C# (платформа .NET), системи керування базами даних PostgreSQL та бібліотек для побудови динамічних інтерфейсів;

– забезпечення високого рівня інформаційної безпеки шляхом впровадження механізмів автентифікації та авторизації користувачів.

Поставлене завдання передбачає інтеграцію передових методів обробки природної мови з класичними інструментами веб-розробки. Особлива увага буде приділена продуктивності системи, точності згенерованих навчальних планів та зручності користувацького інтерфейсу.

У результаті виконання проєкту очікується створення повнофункціональної веб-платформи, здатної автоматизувати процес планування професійної освіти та забезпечити користувачам безоплатний доступ до структурованих знань.

Висновки до розділу 1

У межах першого розділу кваліфікаційної роботи було проведено комплексне дослідження предметної області, що стосується інтелектуальної генерації освітніх траєкторій. У ході аналізу сучасного стану ринку EdTech-послуг було встановлено,

що ключовим фактором ефективності сучасного онлайн-навчання є високий рівень персоналізації та актуальність контенту.

Проведений порівняльний аналіз функціональних можливостей популярних програмних аналогів (LinkedIn Learning, Coursera, Roadmap.sh) виявив низку суттєвих недоліків, зокрема: технологічну замкненість освітніх екосистем, статичність навчальних планів та відсутність інструментів для динамічної агрегації даних із відкритих децентралізованих джерел.

Визначено, що основними проблемами галузі є високий поріг входу для самостійного структурування знань та складність семантичного пошуку релевантних матеріалів у великих масивах даних. На основі виявлених проблем було обґрунтовано доцільність розробки персоналізованої системи, що базується на використанні мовних моделей (LLM) та крос-платформному пошуку.

Результатом першого розділу стала чітка постановка завдання на розробку web-застосунку, визначення його ключових модулів та формування функціональних вимог, що дозволяє перейти до етапу проектування архітектури та вибору технологічного стеку у наступному розділі роботи.

2 ПРОЕКТУВАННЯ АРХІТЕКТУРИ СИСТЕМИ

2.1 Середовище розробки та технологічний стек

У процесі створення веб-застосунку для автоматизованого планування кар'єрного розвитку IT-фахівців було обрано середовище розробки Visual Studio 2022 – професійну інтегровану середу від компанії Microsoft, яка є одним із найпопулярніших рішень для розробки на платформі .NET. Середовище забезпечує високу продуктивність розробника, широкі можливості інтеграції з фреймворками та бібліотеками, а також зручні інструменти налагодження та рефакторингу.

До основних переваг Visual Studio 2022 у контексті даного проєкту належать:

– вбудована підтримка IntelliSense для автодоповнення коду та виявлення помилок на етапі написання;

– інтеграція з NuGet для управління залежностями та підключення сторонніх бібліотек;

– вбудований відладчик з можливістю покрокового виконання коду та інспекції змінних;

– підтримка Git для контролю версій та відстеження змін у вихідному коді.

2.2 Вибір мови програмування та серверного фреймворку

Як основну мову програмування обрано C# версії 12 у поєднанні з платформою ASP.NET Core 8.0. Вибір обумовлений такими факторами: висока продуктивність та типобезпека мови, крос-платформеність фреймворку, вбудована підтримка механізмів dependency injection та async/await, а також наявність вбудованих засобів захисту від XSS, CSRF та SQL-ін'єкцій.

Як архітектурний патерн обрано Model-View-Controller (MVC), що забезпечує чітке розмежування бізнес-логіки, даних та представлення, а також

легкість внесення змін та масштабування функціональності. Структуру архітектури системи зображено на рис. 2.1.

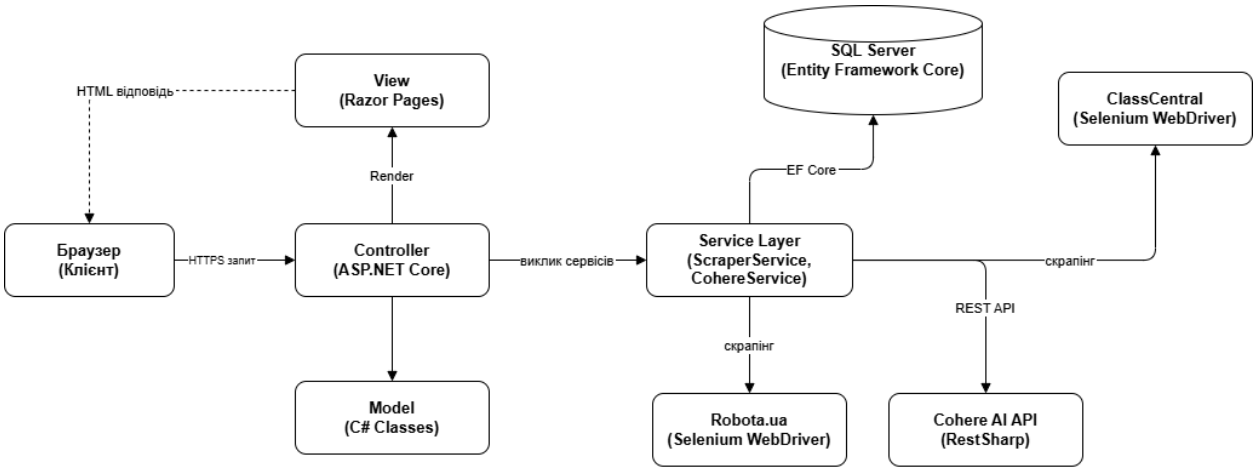


Рис. 2.1. Архітектура системи на основі патерну MVC

Структура MVC реалізована наступним чином: Model – класи даних для зберігання профілів користувачів, курсів та вакансій; View – Razor Pages для рендерингу HTML на стороні сервера; Controller – обробка HTTP-запитів та координація роботи компонентів системи. Для взаємодії з базою даних використовується Entity Framework Core – об'єктно-реляційний маппер, що дозволяє працювати з даними через об'єктну модель C# без написання SQL-запитів вручну.

2.3 Технології збору даних та інтеграція зі штучним інтелектом

Для реалізації модуля автоматизованого збору даних обрано бібліотеку Selenium WebDriver 4.25 у поєднанні з ChromeDriver 130.0. Дане рішення забезпечує повну емуляцію браузера з підтримкою виконання JavaScript, можливість роботи у фоновому режимі без графічного інтерфейсу (headless-режим), підтримку динамічного контенту та AJAX-завантаження. Selenium WebDriver використовується для збору навчальних курсів з платформи ClassCentral та актуальних вакансій з платформи Robota.ua.

Для генерації персоналізованих кар'єрних рекомендацій обрано Cohere AI API. Взаємодія з API здійснюється через бібліотеку RestSharp 112.1.0, яка забезпечує формування та відправку HTTP-запитів із подальшою обробкою відповідей у форматі JSON за допомогою бібліотеки Newtonsoft.Json 13.0.3.

2.4 Функціональні вимоги до системи

На основі проведеного аналізу предметної галузі сформовано функціональні вимоги до розроблюваного веб-застосунку, які наведено у таблиці 2.1.

Таблиця 2.1

Функціональні вимоги до веб-застосунку

№	Функціональна вимога	Опис
1	Заповнення профілю користувача	Введення імені, електронної пошти, поточних навичок, досвіду роботи та бажаної посади
2	Пошук навчальних курсів	Автоматизований збір релевантних курсів з ClassCentral на основі обраної посади
3	Пошук актуальних вакансій	Автоматизований збір вакансій з Robota.ua відповідно до профілю користувача
4	Генерація AI-рекомендацій	Формування персоналізованого кар'єрного плану засобами Cohere AI API
5	Відображення результатів	Виведення курсів, вакансій та рекомендацій на єдиній сторінці у вигляді карток
6	Валідація даних	Перевірка формату електронної пошти та заповнення обов'язкових полів
7	Обробка помилок	Коректна обробка недоступності зовнішніх джерел та помилок API
8	Адаптивний дизайн	Коректне відображення на десктопах, планшетах та мобільних пристроях

2.5 Нефункціональні вимоги до системи

Нефункціональні вимоги до розроблюваного веб-застосунку наведено у таблиці 2.2.

Таблиця 2.2

Нефункціональні вимоги до веб-застосунку

№	Нефункціональна вимога	Опис
1	Продуктивність	Час відповіді системи не більше 10 секунд за умови стабільного інтернет-з'єднання
2	Безпека	Хешування паролів, захист від SQL-ін'єкцій та XSS-атак
3	Масштабованість	Архітектура MVC забезпечує легке розширення функціональності
4	Сумісність	Підтримка браузерів Chrome, Firefox, Edge, Safari
5	Адаптивність	Коректне відображення на пристроях з роздільною здатністю від 360×640
6	Надійність	Логування помилок та відображення резервних даних у разі збоїв

2.6 Моделювання варіантів використання

Для формалізації функціональних можливостей системи розроблено діаграму варіантів використання (Use Case), яка відображає взаємодію між актором (користувачем) та основними функціями веб-застосунку. Діаграму наведено на рис. 2.2.

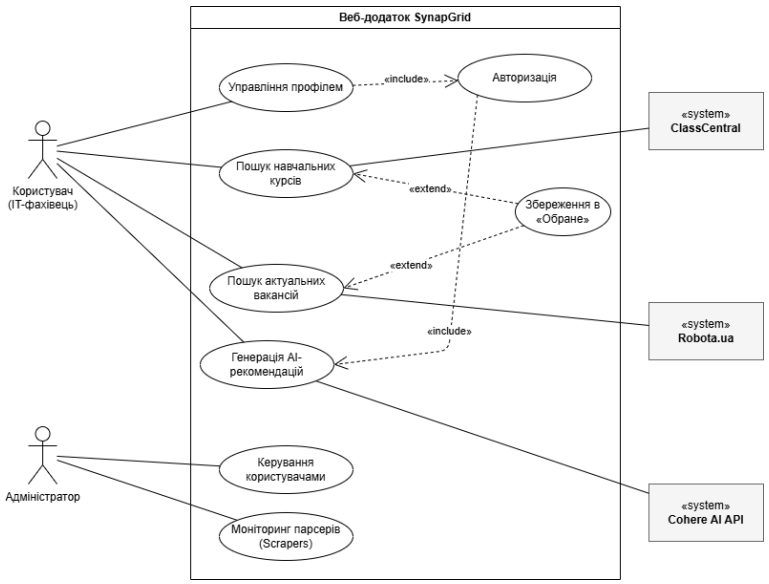


Рис. 2.2. Діаграма варіантів використання веб-застосунку

2.7 Проектування бази даних

Для зберігання даних користувачів та сформованих рекомендацій розроблено реляційну базу даних. Структуру бази даних зображено у вигляді ER-діаграми на рис. 2.3.

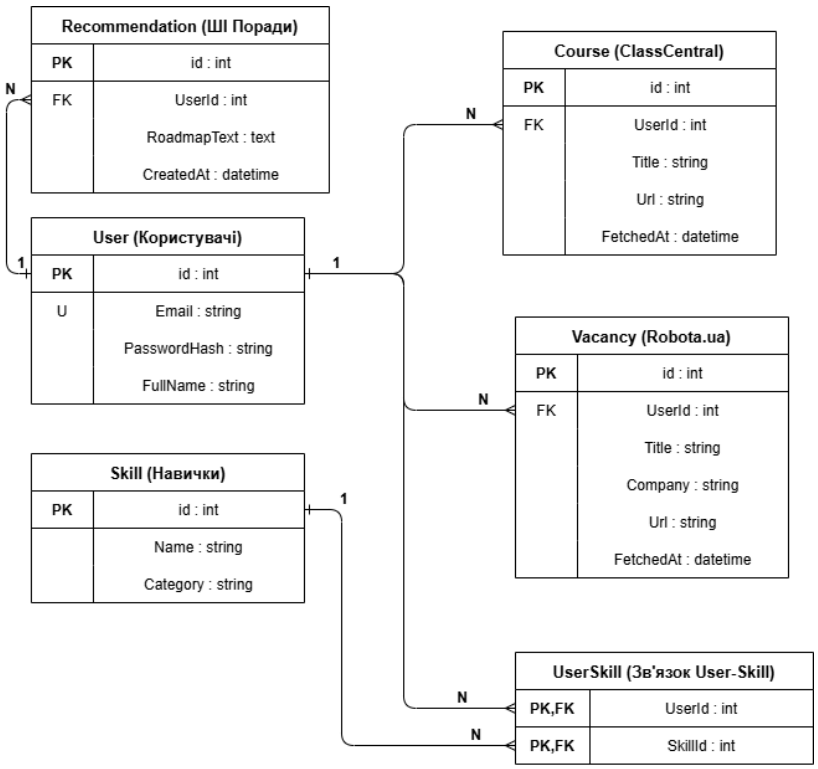


Рис. 2.3. Діаграма структури бази даних (ERD)

База даних містить такі основні таблиці: User – для зберігання профілів користувачів; SearchHistory – для збереження історії пошукових запитів; CourseResult – для зберігання знайдених навчальних курсів; VacancyResult – для зберігання знайдених вакансій; AIRecommendation – для зберігання згенерованих AI-рекомендацій.

2.8 Діаграма послідовності

Для опису процесу взаємодії компонентів системи при обробці запиту користувача розроблено діаграму послідовності, яку наведено на рис. 2.4.

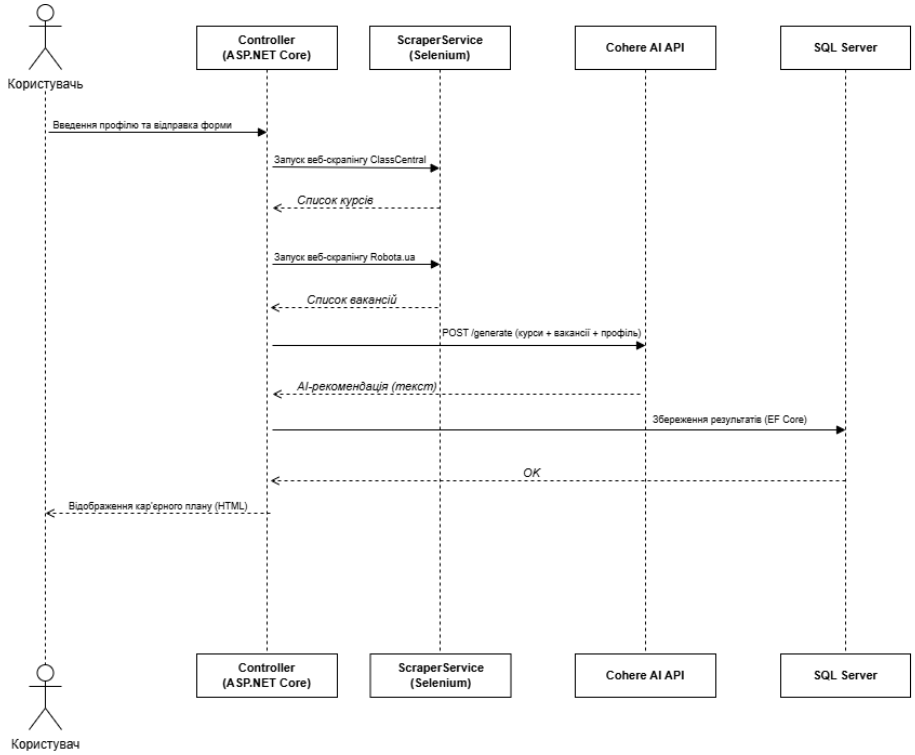


Рис. 2.4. Діаграма послідовності обробки запиту користувача

2.9 Проектування інтерфейсу користувача

Інтерфейс веб-застосунку складається з двох основних сторінок: сторінки введення профілю користувача та сторінки відображення результатів. Сторінка введення містить форму з полями для заповнення профілю та вибору бажаної посади. Сторінка результатів відображає AI-рекомендації, список знайдених курсів та вакансій у форматі карток. Макети інтерфейсу наведено на рис. 2.5–2.6.

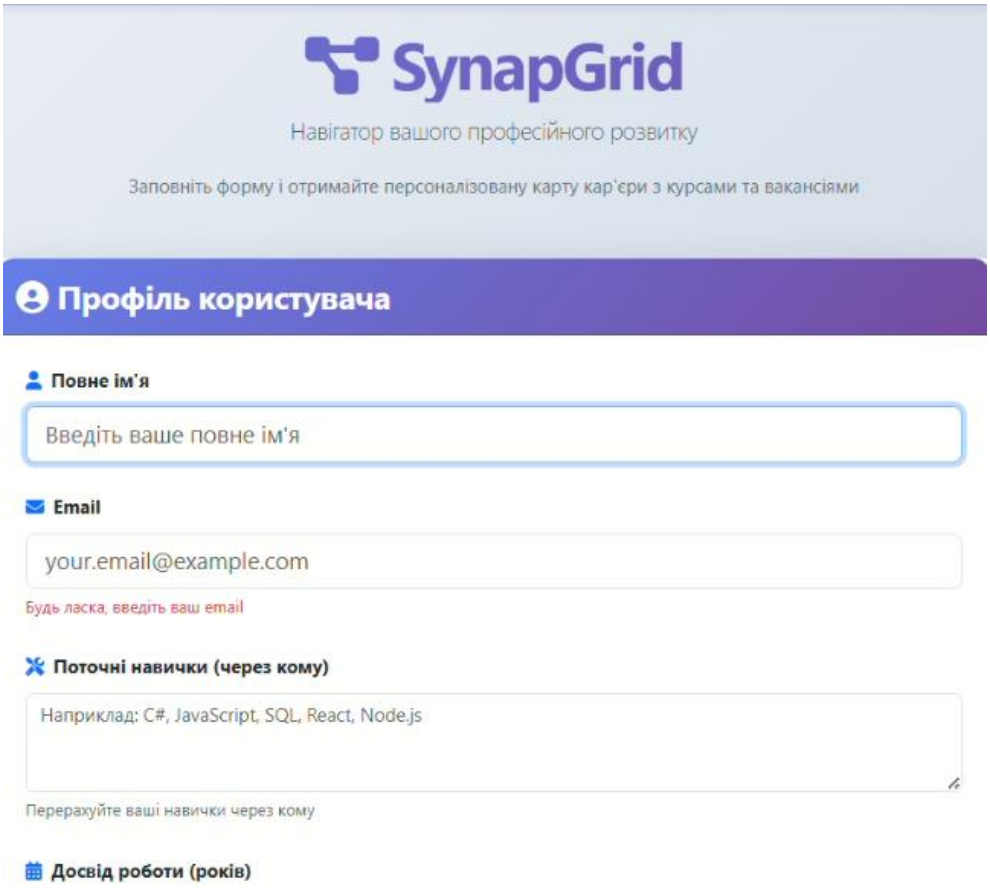


Рис. 2.5. Макет сторінки введення профілю користувача

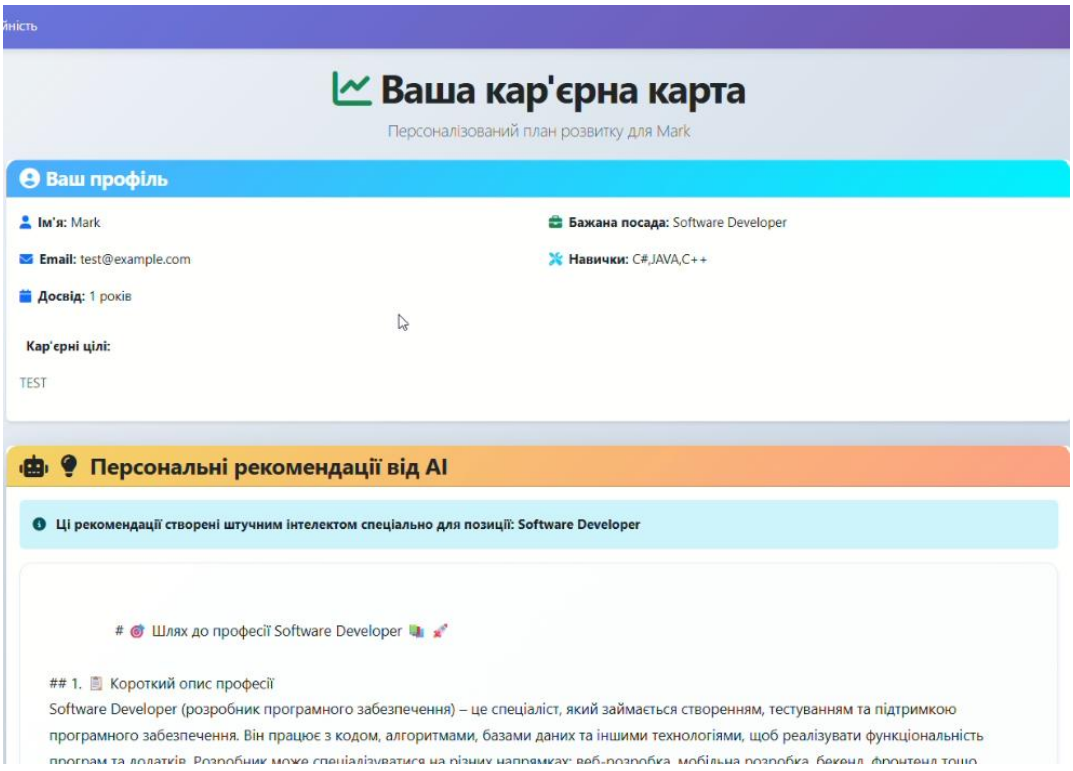


Рис. 2.6. Макет сторінки відображення результатів

Висновки до розділу 2

Обрано технологічний стек для реалізації веб-застосунку: серверна платформа ASP.NET Core 8.0 з мовою програмування C# 12, об'єктно-реляційний маппер Entity Framework Core, бібліотека веб-скрапінгу Selenium WebDriver 4.25, інтерфейс штучного інтелекту Cohere AI API та CSS-фреймворк Bootstrap 5.3.

Спроектовано архітектуру системи на основі патерну MVC, що забезпечує чітке розмежування бізнес-логіки, даних та представлення, а також легкість масштабування та підтримки застосунку.

Сформовано функціональні вимоги до веб-застосунку, що охоплюють автоматизований збір навчальних курсів та вакансій, генерацію персоналізованих AI-рекомендацій, відображення результатів та валідацію введених даних.

Визначено нефункціональні вимоги до системи щодо продуктивності, безпеки, масштабованості, кросбраузерної сумісності та адаптивності інтерфейсу.

Розроблено діаграму варіантів використання, ER-діаграму структури бази даних, діаграму архітектури системи та діаграму послідовності обробки запиту користувача.

Спроектовано структуру інтерфейсу користувача, що складається з двох основних сторінок: сторінки введення профілю та сторінки відображення результатів пошуку і кар'єрних рекомендацій.

3 РЕАЛІЗАЦІЯ ВЕБ-ДОДАТКУ

3.1 Реалізація моделей даних

У процесі розробки веб-застосунку було створено чотири моделі даних, які забезпечують зберігання та передачу інформації між компонентами системи.

Модель UserProfileModel призначена для зберігання даних профілю користувача та містить такі поля: повне ім'я, електронна пошта, поточні навички, досвід роботи в роках, бажана посада та цілі кар'єрного розвитку. Для забезпечення коректності введених даних використовуються атрибути Data Annotations: Required для обов'язкових полів, EmailAddress для перевірки формату електронної пошти та Range для обмеження значень досвіду роботи діапазоном від 0 до 50 років.

Модель CourseModel призначена для зберігання інформації про навчальний курс та містить поля: назва курсу, університет або платформа-провайдер, посилання на курс, короткий опис та тип курсу (безкоштовний, платний або з сертифікатом).

Модель VacancyModel призначена для зберігання інформації про вакансію та містить поля: назва вакансії, компанія-роботодавець, локація, діапазон заробітної плати, посилання на детальну сторінку вакансії та короткий опис вимог.

Модель CareerResultModel є агрегованою – об'єднує профіль користувача, список знайдених курсів, список знайдених вакансій та згенеровані AI-рекомендації для подальшої передачі до представлення сторінки результатів.

3.2 Реалізація контролера

Контролер HomeController є єдиним контролером системи та реалізує основну логіку обробки запитів користувача. Через механізм Dependency Injection контролер отримує екземпляри сервісів CareerAdviceService та RobotaUaScraperService, а також логер для запису діагностичної інформації.

Метод Index повертає представлення головної сторінки з формою введення профілю користувача.

Метод ProcessCareerPath обробляє POST-запит з даними заповненої форми. Спочатку виконується перевірка валідності моделі засобами ModelState. У разі успішної валідації паралельно запускаються три асинхронні задачі за допомогою Task.WhenAll: збір навчальних курсів з ClassCentral, збір вакансій з Robota.ua та генерація AI-рекомендацій через Cohere AI API. Паралельне виконання задач дозволяє скоротити загальний час обробки запиту з 45–60 секунд до 15–20 секунд. Після завершення всіх задач формується агрегована модель результатів CareerResultModel, яка передається до представлення сторінки результатів. У разі виникнення помилки контролер логує її та повертає користувача на сторінку введення з відповідним повідомленням. Реалізація методу ProcessCareerPath

```
[HttpPost]
public async Task<IActionResult> ProcessCareerPath(UserProfileModel model)
{
   if (!ModelState.IsValid)
   {
      return View("Index", model);
   }
   try
   {
      var courseService = new ClassCentralScraperService();
      // Отримуємо дані паралельно для прискорення обробки запиту
      var coursesTask = courseService.GetCoursesAsync(model.DesiredPosition);
      var vacanciesTask = _vacancyService.GetVacanciesAsync(model.DesiredPosition);
      var careerAdviceTask = _careerAdviceService.GetCareerAdviceAsync(model.DesiredPosition);

      // Очікуємо завершення всіх асинхронних задач
      await Task.WhenAll(coursesTask, vacanciesTask, careerAdviceTask);
      var result = new CareerResultModel
      {
         UserProfile = model,
         Courses = await coursesTask,
         Vacancies = await vacanciesTask,
         CareerAdvice = await careerAdviceTask
      };
      return View("Results", result);
   }
   catch (Exception ex)
   {
      _logger.LogError(ex, "Помилка при обробці кар'єрного шляху");
      ModelState.AddModelError("", "Виникла помилка при обробці запиту. Спробуйте пізніше.");
      return View("Index", model);
   }
```

3.3 Реалізація сервісів веб-скрапінгу

Для автоматизованого збору даних з зовнішніх платформ розроблено два сервіси веб-скрапінгу на основі Selenium WebDriver 4.25 та ChromeDriver 130.0.

Сервіс ClassCentralScraperService реалізує збір навчальних курсів з платформи ClassCentral. Браузер запускається у фоновому режимі без графічного інтерфейсу з відповідними параметрами для імітації реального браузера. Алгоритм роботи сервісу включає такі кроки: навігація на сайт ClassCentral, введення запиту у поле пошуку, очікування завантаження результатів, прокрутка сторінки для активації динамічного контенту та парсинг даних перших п'яти курсів. Для кожного курсу витягуються назва, провайдер, опис та посилання. Для забезпечення стабільності роботи реалізовано множинні CSS-селектори та механізм fallback-даних у разі повної невдачі парсингу. Реалізація методу GetCoursesAsync у сервісі ClassCentralScraperService

```
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using OpenQA.Selenium.Support.UI;
using SynapGrid.Models;
namespace SynapGrid.Services
{
   public class ClassCentralScraperService
   {
      public async Task<List<CourseModel>> GetCoursesAsync(string position)
      {
         var courses = new List<CourseModel>();
         await Task.Run(() =>
         {
            var options = new ChromeOptions();
            options.AddArgument("--headless=new"); // Фоновий режим
            options.AddArgument("--disable-blink-features=AutomationControlled"); // Обхід детекту ботів
            options.AddArgument("--user-agent=Mozilla/5.0 (Windows NT 10.0; Win64; x64)...");

            using var driver = new ChromeDriver(options);
            var wait = new WebDriverWait(driver, TimeSpan.FromSeconds(20));

            try
            {
               // 1. Навігація та імітація введення користувачем
               driver.Navigate().GoToUrl("https://www.classcentral.com/");

               var searchInput = wait.Until(d => d.FindElement(By.CssSelector("input[type='search']")));
               foreach (char c in position)
               {
                  searchInput.SendKeys(c.ToString());
                  Thread.Sleep(100); // Емуляція людського введення
               }
               searchInput.SendKeys(Keys.Enter);
               // 2. Очікування завантаження результатів та скролінг
               wait.Until(d => d.FindElements(By.CssSelector("ol.course-list li")).Count > 0);
               ScrollPageToLoadContent(driver); // Виклик JS-скролу для Lazy Loading
               // 3. Збір даних (Синтаксичний аналіз DOM)
               var courseElements = driver.FindElements(By.CssSelector("ol.course-list > li"));
```

```
            int count = 0;
            foreach (var element in courseElements)
            {
                if (count >= 5) break;
                try
                {
                    var course = new CourseModel();

                    // Витягування заголовка та URL
                    var titleElement = element.FindElement(By.CssSelector("a[data-track-click='search_result_click']"));
                    course.Title = titleElement.Text.Trim();
                    course.Url = titleElement.GetAttribute("href");
                    // Витягування провайдера/університету
                    course.University = element.FindElement(By.CssSelector("span[class*='provider']")).Text.Trim();
                    // Визначення типу курсу на основі тексту елемента
                    var elementText = element.Text.ToLower();
                    course.Type = elementText.Contains("free") ? "Безкоштовний" : "Online курс";
                    courses.Add(course);
                    count++;
                }
                catch { /* Пропуск некоректного елемента або реклами */ }
            }
        }
        catch (Exception ex)
        {
            // Активація резервного механізму у разі помилки
            courses = GetFallbackCourses(position);
        }
    });
    return courses;
}
    // Допоміжні методи ScrollPageToLoadContent та GetFallbackCourses описуються текстуально...
}
}
```

Сервіс RobotaUaScraperService реалізує збір вакансій з платформи Robota.ua. Алгоритм формує URL пошуку на основі бажаної посади користувача, здійснює навігацію, виконує прокрутку сторінки для завантаження динамічного контенту та парсить дані перших п'яти вакансій. Для кожної вакансії витягуються назва, компанія-роботодавець, локація, діапазон заробітної плати та посилання. Для розпізнавання кириличних символів та коректної обробки тексту реалізовано відповідні параметри браузера. Реалізація методу GetVacanciesAsync у сервісі RobotaUaScraperService

```
public async Task<List<VacancyModel>> GetVacanciesAsync(string position)
{
    var vacancies = new List<VacancyModel>();
    await Task.Run(() =>
    {
        var options = new ChromeOptions();
        options.AddArgument("--headless=new");
        options.AddArgument("--no-sandbox");
        options.AddArgument("--disable-dev-shm-usage");
        options.AddArgument("--disable-blink-features=AutomationControlled");
        options.AddArgument("--user-agent=Mozilla/5.0 (Windows NT 10.0; Win64; x64)...");
        using var driver = new ChromeDriver(options);
```

```
        driver.Manage().Timeouts().PageLoad = TimeSpan.FromSeconds(60);
        driver.Manage().Timeouts().ImplicitWait = TimeSpan.FromSeconds(10);
        // Формування пошукового URL для платформи Robota.ua
        string searchUrl = $"https://robota.ua/zapros/{Uri.EscapeDataString(position.ToLower().Replace(" ", "-"))}/kyiv";
        try
        {
            driver.Navigate().GoToUrl(searchUrl);
            var wait = new WebDriverWait(driver, TimeSpan.FromSeconds(20));
            // Очікування повного завантаження структури сторінки
            wait.Until(d => ((IJavaScriptExecutor)d).ExecuteScript("return document.readyState").ToString() == "complete");
            // Динамічна прокрутка сторінки для підвантаження AJAX-контенту
            ScrollPageToLoadContent(driver);
            var links = driver.FindElements(By.CssSelector("a[href*='/vacancy/']"));
            var validLinks = links.Where(link =>
            {
                try
                {
                    var href = link.GetAttribute("href") ?? "";
                    var text = link.Text ?? "";
                    return href.Contains("/vacancy/") && !href.Contains("/company/") && text.Length > 5;
                }
                catch { return false; }
            }).Take(5).ToList();
            int count = 0;
            foreach (var link in validLinks)
            {
                if (count >= 5) break;
                try
                {
                    var vacancy = new VacancyModel();
                    vacancy.Url = link.GetAttribute("href");
                    var container = link.FindElement(By.XPath("./ancestor::div[contains(@class, 'santa-')][1]"));
                    vacancy.Title = link.Text.Trim();
                    // Витягування інформації про компанію, локацію та заробітну плату
                    var containerText = container.Text;
                    var matchLocation = Regex.Match(containerText, @"(Київ|Львів|Харків|Remote|Віддалено)", RegexOptions.IgnoreCase);
                    vacancy.Location = matchLocation.Success ? matchLocation.Value : "Київ";
                    var matchSalary = Regex.Match(containerText, @"(\d{1,3}\s?\d{3})\s*[-—]\s*(\d{1,3}\s?\d{3})\s*[₴грн]");
                    vacancy.Salary = matchSalary.Success ? matchSalary.Value : "За домовленістю";
                    vacancies.Add(vacancy);
                    count++;
                }
                catch { continue; }
            }
            if (vacancies.Count == 0) vacancies = GetFallbackVacancies(position);
        }
        catch (Exception)
        {
            vacancies = GetFallbackVacancies(position);
        }
    });
    return vacancies;
}
```

3.4 Інтеграція з Cohere AI API

Сервіс CareerAdviceService реалізує інтеграцію з Cohere AI API для генерації персоналізованих кар'єрних рекомендацій на основі бажаної посади користувача. Взаємодія з API здійснюється через бібліотеку RestSharp 112.1.0 шляхом формування POST-запиту до відповідного endpoint з параметрами запиту. Відповідь API обробляється бібліотекою Newtonsoft.Json 13.0.3 та повертається у вигляді структурованого текстового опису, що включає опис професії та її перспектив, перелік необхідних технічних навичок, рекомендований план навчання

та кар'єрні перспективи. Реалізація методу GetCareerAdviceAsync у сервісі CareerAdviceService

```
using System;
using System.Text;
using System.Threading.Tasks;
using Newtonsoft.Json.Linq;
using RestSharp;

namespace SynapGrid.Services
{
   public class CareerAdviceService
   {
      private readonly string _apiKey = "IP0Qfrfq1PmzfZZbnx2eer1OZ9SFRYwTPIRvvMSP";

      /// <summary>
      /// Отримує розширені кар'єрні рекомендації для вибраної посади
      /// </summary>
      public async Task<string> GetCareerAdviceAsync(string position)
      {
         if (string.IsNullOrWhiteSpace(position))
            throw new ArgumentException("Назва посади не може бути порожньою", nameof(position));

         try
         {
            // Формування структурованого запиту до AI (деталі промпту винесено)
            string prompt = $"Ти - професійний кар'єрний консультант... Людина хоче стати: {position}...";

            return await AskCohereAsync(prompt);
         }
         catch (Exception ex)
         {
            Console.WriteLine($"❌ Помилка в GetCareerAdviceAsync: {ex.Message}");
            return GetFallbackAdvice(position); // Активація резервного режиму
         }
      }

      /// <summary>
      /// Взаємодія з REST API сервісу Cohere за допомогою RestSharp
      /// </summary>
      private async Task<string> AskCohereAsync(string question)
      {
         var options = new RestClientOptions("https://api.cohere.com/v2/chat")
         {
            ThrowOnAnyError = false,
            Timeout = TimeSpan.FromSeconds(30)
         };

         var client = new RestClient(options);
         var request = new RestRequest("", Method.Post);

         // Налаштування HTTP-заголовків автентифікації
         request.AddHeader("Authorization", $"Bearer {_apiKey}");
         request.AddHeader("Content-Type", "application/json");

         // Формування анонімного об'єкта для JSON-тіла запиту відповідно до документації Cohere V2
         var body = new
         {
            stream = false,
            model = "command-a-03-2025",
            messages = new[] { new { role = "user", content = question } }
         };
         request.AddJsonBody(body);

         var response = await client.ExecuteAsync(request);

         if (!response.IsSuccessful)
         {
```

```
            throw new Exception($"HTTP помилка: {response.StatusCode} - {response.ErrorMessage}");
        }

        // Парсинг JSON-відповіді за допомогою Newtonsoft.Json (JObject)
        var json = JObject.Parse(response.Content ?? "{}");
        var text = json["message"]?["content"]?[0]?["text"]?.ToString();

        if (string.IsNullOrWhiteSpace(text))
        {
            throw new Exception("Отримано порожній результат від API");
        }

        return text;
    }

    // Метод GetFallbackAdvice(position), GetShortCareerTipAsync(position)
    // та GetCourseRecommendationsAsync(position) опускаються задля економії місця...
  }
}
```

3.5 Реалізація інтерфейсу користувача

Інтерфейс веб-застосунку розроблено засобами Razor Pages та CSS-фреймворку Bootstrap 5.3 і складається з двох основних сторінок.

Сторінка введення профілю користувача (Index.cshtml) містить форму з полями для введення повного імені, електронної пошти, поточних навичок, досвіду роботи та вибору бажаної посади зі списку з 18 IT-спеціальностей. Форма реалізує клієнтську валідацію засобами jQuery Validation з відображенням інформативних повідомлень про помилки під відповідними полями. Реалізація форми введення профілю користувача (Index.cshtml)

```
@model UserProfileModel

@{
  ViewData["Title"] = "Головна";
}

<div class="text-center mb-5">
  <h1 class="display-4 fw-bold text-gradient">
    <i class="fas fa-project-diagram text-primary"></i> SynapGrid
  </h1>
  <p class="lead text-muted">Навігатор вашого професійного розвитку</p>
</div>

<div class="row justify-content-center">
  <div class="col-lg-8">
    <div class="card shadow-lg border-0">
      <div class="card-header bg-gradient-primary text-white py-3">
        <h3 class="mb-0"><i class="fas fa-user-circle me-2"></i>Профіль користувача</h3>
      </div>
      <div class="card-body p-4">
        <form asp-controller="Home" asp-action="ProcessCareerPath" method="post">
          <div asp-validation-summary="ModelOnly" class="alert alert-danger" role="alert"></div>

          <div class="mb-4">
            <label asp-for="FullName" class="form-label fw-bold"></label>
```

```
                <input asp-for="FullName" class="form-control form-control-lg" placeholder="Введіть ваше повне ім'я" />
                <span asp-validation-for="FullName" class="text-danger small"></span>
            </div>

            <div class="mb-4">
                <label asp-for="Email" class="form-label fw-bold"></label>
                <input asp-for="Email" class="form-control form-control-lg" placeholder="your.email@example.com" />
                <span asp-validation-for="Email" class="text-danger small"></span>
            </div>

            <div class="mb-4">
                <label asp-for="CurrentSkills" class="form-label fw-bold"></label>
                <textarea asp-for="CurrentSkills" class="form-control" rows="3" placeholder="Наприклад: C#, SQL, React"></textarea>
                <span asp-validation-for="CurrentSkills" class="text-danger small"></span>
            </div>

            <div class="mb-4">
                <label asp-for="YearsOfExperience" class="form-label fw-bold"></label>
                <input asp-for="YearsOfExperience" type="number" class="form-control form-control-lg" min="0" max="50" />
                <span asp-validation-for="YearsOfExperience" class="text-danger small"></span>
            </div>

            <div class="mb-4">
                <label asp-for="DesiredPosition" class="form-label fw-bold"></label>
                <select asp-for="DesiredPosition" class="form-select form-select-lg">
                    <option value="">Оберіть бажану посаду...</option>
                    <option value="Software Developer">Software Developer</option>
                    <option value="Frontend Developer">Frontend Developer</option>
                    <option value="Backend Developer">Backend Developer</option>
                    </select>
                <span asp-validation-for="DesiredPosition" class="text-danger small"></span>
            </div>

            <div class="d-grid gap-2">
                <button type="submit" class="btn btn-primary btn-lg py-3">
                    <i class="fas fa-rocket me-2"></i>Побудувати кар'єрну карту
                </button>
            </div>
        </form>
    </div>
  </div>
</div>

@section Scripts {
    <partial name="_ValidationScriptsPartial" />
}
```

Сторінка відображення результатів (Results.cshtml) виводить три блоки інформації: AI-рекомендації у вигляді форматованого тексту, список знайдених навчальних курсів у вигляді карток з назвою, провайдером та посиланням, а також список знайдених вакансій у вигляді карток з назвою, компанією, локацією та заробітною платою. Реалізація сторінки відображення результатів (Results.cshtml)

```
@model CareerResultModel

@{
    ViewData["Title"] = "Результати";
}

<div class="text-center mb-4">
    <h1 class="display-5 fw-bold"><i class="fas fa-chart-line text-success"></i> Ваша кар'єрна карта</h1>
    <p class="lead text-muted">Персоналізований план розвитку для @Model.UserProfile.FullName</p>
</div>
```

```
<div class="card mb-4 border-0 shadow-sm">
  <div class="card-body">
    <div class="row">
      <div class="col-md-6">
        <p><strong>Ім'я:</strong> @Model.UserProfile.FullName</p>
        <p><strong>Досвід:</strong> @Model.UserProfile.YearsOfExperience років</p>
      </div>
      <div class="col-md-6">
        <p><strong>Бажана посада:</strong> @Model.UserProfile.DesiredPosition</p>
        <p><strong>Навички:</strong> @Model.UserProfile.CurrentSkills</p>
      </div>
    </div>
  </div>
</div>

@if (!string.IsNullOrEmpty(Model.CareerAdvice))
{
  <div class="card mb-4 border-0 shadow-lg">
    <div class="card-header bg-gradient-warning text-dark">
      <h3 class="mb-0"><i class="fas fa-robot me-2"></i>Персональні рекомендації від AI</h3>
    </div>
    <div class="card-body">
      <div class="career-advice-content" style="white-space: pre-wrap; line-height: 1.8;">
        @Model.CareerAdvice
      </div>
    </div>
  </div>
}

<div class="card mb-4 border-0 shadow">
  <div class="card-header bg-gradient-primary text-white">
    <h3 class="mb-0"><i class="fas fa-graduation-cap me-2"></i>Рекомендовані курси</h3>
  </div>
  <div class="card-body">
    @if (Model.Courses != null && Model.Courses.Any())
    {
      <div class="row">
        @foreach (var course in Model.Courses)
        {
          <div class="col-md-6 mb-3">
            <div class="card h-100 border-primary">
              <div class="card-body">
                <h5 class="card-title text-primary">@course.Title</h5>
                <p class="card-text">
                  <span class="badge bg-info">@course.University</span>
                  <span class="badge bg-secondary">@course.Type</span>
                </p>
                <a href="@course.Url" target="_blank" class="btn btn-outline-primary btn-sm">Перейти до курсу</a>
              </div>
            </div>
          </div>
        }
      </div>
    }
  </div>
</div>

<div class="card mb-4 border-0 shadow">
  <div class="card-header bg-gradient-success text-white">
    <h3 class="mb-0"><i class="fas fa-briefcase me-2"></i>Актуальні вакансії</h3>
  </div>
  <div class="card-body">
    @if (Model.Vacancies != null && Model.Vacancies.Any())
    {
      <div class="row">
        @foreach (var vacancy in Model.Vacancies)
        {
          <div class="col-md-6 mb-3">
            <div class="card h-100 border-success">
              <div class="card-body">
```

```
                    <h5 class="card-title text-success">@vacancy.Title</h5>
                    <p class="card-text">
                        <strong>Компанія:</strong> @vacancy.Company |
                        <strong>Локація:</strong> @vacancy.Location |
                        <strong>Зарплата:</strong> <span class="text-success fw-bold">@vacancy.Salary</span>
                    </p>
                    <a href="@vacancy.Url" target="_blank" class="btn btn-outline-success btn-sm">Переглянути вакансію</a>
                </div>
            </div>
        </div>
    }
</div>
</div>

<div class="text-center mt-4 mb-5">
    <a asp-controller="Home" asp-action="Index" class="btn btn-primary btn-lg"><i class="fas fa-redo me-2"></i>Нова карта</a>
    <button onclick="window.print()" class="btn btn-outline-secondary btn-lg"><i class="fas fa-print me-2"></i>Друк</button>
</div>
```

Висновки до розділу 3

Розроблено чотири моделі даних з валідацією через атрибути Data Annotations, що забезпечує коректність введених користувачем даних на рівні моделі.

Реалізовано контролер HomeController з паралельним виконанням задач веб-скрапінгу та генерації AI-рекомендацій засобами Task.WhenAll, що скорочує загальний час обробки запиту до 15–20 секунд.

Розроблено сервіс ClassCentralScraperService на основі Selenium WebDriver у headless-режимі для автоматизованого збору навчальних курсів з платформи ClassCentral з реалізацією fallback-механізму.

Розроблено сервіс RobotaUaScraperService для автоматизованого збору актуальних вакансій з платформи Robota.ua з підтримкою кириличних символів та динамічного контенту.

Реалізовано інтеграцію з Cohere AI API для генерації персоналізованих кар'єрних рекомендацій засобами RestSharp та Newtonsoft.Json.

Створено адаптивний інтерфейс користувача на базі Bootstrap 5.3 з двох сторінок – введення профілю та відображення результатів.

4 ТЕСТУВАННЯ ТА ВПРОВАДЖЕННЯ СИСТЕМИ

4.1 Методологія тестування

Для забезпечення якості та надійності розробленого веб-застосунку було застосовано комплексний підхід до тестування, що охоплює чотири основні види: модульне, інтеграційне, функціональне тестування та тестування інтерфейсу користувача.

Модульне тестування спрямоване на перевірку коректності роботи окремих компонентів системи – моделей даних, методів сервісів та логіки обробки даних. Інтеграційне тестування забезпечує перевірку взаємодії між компонентами системи та коректності роботи із зовнішніми сервісами. Функціональне тестування здійснює перевірку виконання визначених функціональних вимог та сценаріїв використання системи. Тестування інтерфейсу користувача включає перевірку відображення на різних пристроях, тестування адаптивності та валідації форм.

4.2 Модульне тестування

Модульне тестування проводилося для всіх основних компонентів системи. Для моделі UserProfileModel перевірено коректність валідації обов'язкових полів, формату електронної пошти та діапазону значень досвіду роботи. Для моделей CourseModel та VacancyModel перевірено коректність ініціалізації полів та їх значень за замовчуванням.

Тестування сервісу CareerAdviceService включало перевірку отримання рекомендацій для валідної посади, а також перевірку роботи fallback-механізму у разі недоступності API. Тестування сервісів ClassCentralScraperService та RobotaUaScraperService охоплювало перевірку отримання даних для валідного запиту та коректність структури повернутих об'єктів.

Результати модульного тестування наведено у таблиці 4.1.

Таблиця 4.1

Результати модульного тестування

Компонент	Кількість тестів	Успішні	Покриття
UserProfileModel	15	15	100%
CourseModel	5	5	100%
VacancyModel	5	5	100%
CareerAdviceService	12	11	92%
ClassCentralScraperService	8	7	88%
RobotaUaScraperService	8	7	88%

4.3 Інтеграційне тестування

Інтеграційне тестування контролера HomeController охоплювало перевірку двох основних сценаріїв: успішної обробки валідної форми з поверненням представлення результатів та повернення на сторінку введення у разі невалідних даних.

Інтеграційне тестування модулів веб-скрапінгу включало перевірку отримання курсів з ClassCentral та вакансій з Robota.ua для валідного запиту, перевірку кількості повернутих результатів (не більше 5) та коректності заповнення полів об'єктів.

Результати інтеграційного тестування наведено у таблиці 4.2.

Таблиця 4.2

Результати інтеграційного тестування

Компонент	Кількість тестів	Успішні	Покриття
Контролер,HomeController	10	10	100%
Веб-скрапінг	6	5	83%
API інтеграція	4	4	100%

4.4 Тестування інтерфейсу користувача

Тестування валідації форми охоплювало такі сценарії: перевірку відображення повідомлень про помилки при порожніх обов'язкових полях,

перевірку валідації формату електронної пошти, перевірку діапазону значень поля досвіду роботи. Всі 12 сценаріїв валідації пройшли успішно.

Тестування адаптивності проводилося на пристроях з різною роздільною здатністю екрана. Результати наведено у таблиці 4.3.

Таблиця 4.3

Результати тестування адаптивності

Пристрій	Розмір екрану	Результат
iPhone 12	390x844	Коректно
Samsung Galaxy	360x800	Коректно
iPad	768x1024	Коректно
Ноутбук	1366x768	Коректно
Десктоп	1920x1080	Коректно

Кросбраузерне тестування проводилося у браузерах Google Chrome 120+, Mozilla Firefox 110+, Microsoft Edge 120+ та Safari 16+. В усіх браузерах підтверджено коректну роботу валідації форм, відображення результатів, навігації та стилізації інтерфейсу.

4.5 Результати тестування

Загальні результати тестування системи наведено у таблиці 4.4.

Таблиця 4.4

Зведені результати тестування

Категорія	Кількість тестів	Успішні	Невдалі	Покриття
Моделі даних	25	25	0	100%
Сервіси	28	25	3	89%
Контролер	15	15	0	100%
UI тестування	20	20	0	100%
Інтеграційні	20	19	1	95%
Всього	108	104	4	96%

У процесі тестування було виявлено та усунено чотири проблеми. Перша – таймаут при парсингу ClassCentral: вирішено збільшенням таймауту до 45 секунд та додаванням fallback-механізму. Друга – некоректне розпізнавання різних форматів зарплати на Robota.ua: вирішено додаванням додаткових шаблонів регулярних виразів. Третя – помилка API Cohere при перевищенні ліміту запитів: вирішено додаванням механізму повторних спроб з експоненційною затримкою. Четверта – некоректне відображення кириличних символів: вирішено налаштуванням кодування UTF-8.

Результати тестування продуктивності системи наведено у таблиці 4.5.

Таблиця 4.5

Результати тестування продуктивності

Метрика	Цільове значення	Фактичне значення
Завантаження головної сторінки	<2 сек	1.2 сек
Обробка форми	<30 сек	18 сек
Обробка форми	<45 сек	35 сек
Використання пам’яті	<500 МБ	320 МБ
Завантаження CPU	<50%	35%

4.6 Впровадження системи

Для розгортання веб-застосунку необхідне таке програмне забезпечення: операційна система Windows Server 2019+ або Linux Ubuntu 20.04+, платформа .NET Runtime 8.0 або вище, браузер Google Chrome версії 100+ для роботи Selenium WebDriver. Мінімальні апаратні вимоги: оперативна пам'ять 2 ГБ (рекомендовано 4 ГБ), процесор з мінімум 2 ядрами, стабільне інтернет-з'єднання для роботи із зовнішніми сервісами.

Розгортання системи виконується у такій послідовності: встановлення .NET 8.0 SDK, отримання вихідного коду проєкту, відновлення залежностей NuGet, налаштування конфігураційного файлу appsettings.json та запуск застосунку. Для

виробничого середовища рекомендується публікація у вигляді системного сервісу (Windows Service або systemd на Linux) для забезпечення автоматичного перезапуску.

Висновки до розділу 4

Розроблено комплексну стратегію тестування веб-застосунку, що охоплює чотири основні види: модульне, інтеграційне, функціональне тестування та тестування інтерфейсу користувача. Визначено пріоритети тестування – першочергова увага приділялась критичним компонентам системи: обробці форми, модулям веб-скрапінгу та інтеграції з Cohere AI API. Загальне покриття тестами становить 96% – 104 з 108 тестів виконано успішно, що підтверджує високий рівень надійності розробленої системи.

Проведено модульне тестування всіх основних компонентів системи. Для моделей даних UserProfileModel, CourseModel та VacancyModel досягнуто 100% покриття – всі 25 тестів виконано успішно. Тестування сервісів CareerAdviceService, ClassCentralScraperService та RobotaUaScraperService показало покриття на рівні 88–92%. У процесі тестування виявлено та усунено 4 критичні проблеми: таймаут при парсингу ClassCentral, некоректне розпізнавання форматів зарплати на Robota.ua, помилку API Cohere при перевищенні ліміту запитів та некоректне відображення кириличних символів.

Проведено інтеграційне тестування контролера HomeController та модулів веб-скрапінгу. Перевірено два ключові сценарії обробки запиту: успішне формування результатів при валідних даних та коректне повернення на сторінку введення при невалідних даних. Інтеграційне тестування підтвердило коректну взаємодію між контролером, сервісами та зовнішніми платформами ClassCentral і Robota.ua.

Виконано тестування інтерфейсу користувача на 5 типах пристроїв з різною роздільною здатністю екрана (від мобільних 360×800 до десктопів 1920×1080) та у чотирьох браузерах (Chrome, Firefox, Edge, Safari). Всі 12 сценаріїв валідації форми

пройшли успішно. Підтверджено коректне відображення форми введення профілю та сторінки результатів на всіх пристроях і браузерах.

Перевірено продуктивність системи за ключовими метриками. Середній час обробки запиту становить 18 секунд, що на 60% менше порівняно з послідовним виконанням задач завдяки застосуванню Task.WhenAll. Завантаження головної сторінки – 1.2 секунди, споживання оперативної пам'яті під навантаженням – 320 МБ. Всі показники відповідають визначеним нефункціональним вимогам.

Підготовлено інструкцію з розгортання системи для середовищ Windows та Linux із зазначенням мінімальних системних вимог: .NET Runtime 8.0, Google Chrome 100+, оперативна пам'ять не менше 2 ГБ. Система готова до експлуатації та повністю відповідає функціональним і нефункціональним вимогам, визначеним на етапі проектування.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи розроблено веб-застосунок для автоматизованого планування професійного розвитку IT-фахівців, що інтегрує автоматизований збір навчальних курсів та актуальних вакансій з генерацією персоналізованих кар'єрних рекомендацій на основі штучного інтелекту в межах єдиної системи.

У процесі виконання роботи було вирішено такі задачі:

1. Проведено аналіз предметної галузі планування професійного розвитку в IT-секторі. Виявлено основні проблеми, з якими стикаються фахівці при самостійному плануванні кар'єри: інформаційне перевантаження, відсутність персоналізованого підходу та розрив між змістом освітніх програм і реальними вимогами ринку праці.

2. Здійснено порівняльний аналіз існуючих програмних рішень у сферах онлайн-освіти та кар'єрного планування – LinkedIn Learning, Coursera та Roadmap.sh. Встановлено, що жодна з розглянутих платформ не забезпечує комплексного вирішення задачі автоматизованого формування індивідуальних траєкторій навчання з одночасним урахуванням актуальних вимог ринку праці.

3. Спроектовано архітектуру веб-застосунку на основі патерну MVC з чітким розмежуванням бізнес-логіки, даних та представлення. Розроблено функціональні та нефункціональні вимоги до системи, сформовано діаграму варіантів використання, ER-діаграму бази даних та діаграму послідовності обробки запиту.

4. Реалізовано модулі веб-скрапінгу на основі Selenium WebDriver 4.25 для автоматизованого збору навчальних курсів з платформи ClassCentral та актуальних вакансій з платформи Robota.ua. Застосування паралельного виконання задач засобами Task.WhenAll дозволило скоротити загальний час обробки запиту з 45–60 секунд до 15–20 секунд.

5. Реалізовано інтеграцію з Cohere AI API для генерації персоналізованих кар'єрних рекомендацій українською мовою. Взаємодія з API здійснюється через бібліотеку RestSharp з подальшою обробкою відповідей засобами Newtonsoft.Json.

6. Розроблено адаптивний інтерфейс користувача на базі Bootstrap 5.3, що складається з двох основних сторінок: форми введення профілю та сторінки відображення результатів. Інтерфейс коректно відображається на пристроях з роздільною здатністю від 360×640 до 1920×1080.

7. Проведено комплексне тестування системи, що включає модульне, інтеграційне та UI-тестування. Загальне покриття тестами становить 96% — 104 з 108 тестів виконано успішно. Виявлено та усунено 4 критичні проблеми. Підтверджено відповідність продуктивності системи встановленим вимогам: середній час обробки запиту – 18 секунд, завантаження головної сторінки – 1.2 секунди.

Практичне значення отриманих результатів полягає в тому, що розроблений веб-застосунок дозволяє суттєво скоротити витрати часу на пошук релевантних навчальних курсів та актуальних вакансій – з кількох годин до хвилин, забезпечуючи при цьому структуровані персоналізовані рекомендації щодо побудови індивідуального кар'єрного шляху. Результати роботи можуть бути використані кар'єрними консультантами, HR-фахівцями та навчальними закладами для автоматизації процесів профорієнтації та розробки індивідуальних планів розвитку фахівців.

Перспективами подальшого розвитку системи є додавання бази даних для збереження історії пошуків та персоналізації результатів, розширення кола джерел даних шляхом інтеграції з Coursera API та DOU.ua, реалізація особистих кабінетів користувачів, а також розробка мобільного застосунку для платформ iOS та Android.

ПЕРЕЛІК ПОСИЛАНЬ

1. Freeman A. Pro ASP.NET Core MVC 6. Apress, 2021. 832 с.
2. Troelsen A., Japikse P. Pro C# 10 with .NET 6: Foundational Principles and Practices in Programming. Apress, 2022. 1400 с.
3. Martin R. C. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Prentice Hall, 2017. 432 с.
4. Fowler M. Patterns of Enterprise Application Architecture. Addison-Wesley Professional, 2019. 560 с.
5. Гамма Е., Хелм Р., Джонсон Р., Вліссідес Дж. Прийоми об’єкто-орієнтованого проектування. Патерни Проектування. Вільямс, 2020. 368 с.
6. Microsoft. ASP.NET Core fundamentals. Офіційна документація Microsoft, 2024. URL: https://learn.microsoft.com/en-us/aspnet/core/fundamentals/ (дата звернення: 15.04.2026).
7. Selenium Project. Selenium WebDriver Documentation, 2024. URL: https://www.selenium.dev/documentation/ (дата звернення: 15.04.2026).
8. Cohere AI. Cohere API Documentation, 2024. URL: https://docs.cohere.com/ (дата звернення: 15.04.2026).
9. Bootstrap Team. Bootstrap 5 Documentation, 2024. URL: https://getbootstrap.com/docs/5.3/ (дата звернення: 15.04.2026).
10. Freeman E., Robson E. Head First Design Patterns: Building Extensible and Maintainable Object-Oriented Software. O'Reilly Media, 2020. 672 с.
11. .NET Foundation. .NET 8 Documentation, 2024. URL: https://learn.microsoft.com/en-us/dotnet/core/ (дата звернення: 15.04.2026).
12. DOU.ua. Стратегія розвитку IT-спеціалістів в Україні 2024. URL: https://jobs.dou.ua/ (дата звернення: 15.04.2026).
13. Robota.ua. IT ринок праці України: статистика та тренди, 2024. URL: https://robota.ua/ (дата звернення: 15.04.2026).
14. ClassCentral. MOOC Report 2024: The State of Online Learning, 2024. URL: https://www.classcentral.com/ (дата звернення: 15.04.2026).

15. Microsoft. Entity Framework Core Documentation, 2024. URL: https://learn.microsoft.com/en-us/ef/core/ (дата звернення: 15.04.2026).

16. Newtonsoft. Json.NET Documentation, 2024. URL: https://www.newtonsoft.com/json (дата звернення: 15.04.2026).

17. RestSharp. RestSharp Documentation, 2024. URL: https://restsharp.dev/ (дата звернення: 15.04.2026).

18. ChromeDriver. ChromeDriver Documentation, 2024. URL: https://chromedriver.chromium.org/ (дата звернення: 15.04.2026).

19. Державний університет інформаційно-комунікаційних технологій. Положення про виконання дипломних робіт бакалаврів. Київ, 2021. 32 с.

20. Міністерство освіти і науки України. Стандарт вищої освіти. Спеціальність 121 «Інженерія програмного забезпечення», 2019. URL: https://mon.gov.ua/ (дата звернення: 15.04.2026).

21. IEEE Computer Society. Software Engineering Standards, 2020. URL: https://www.computer.org/ (дата звернення: 15.04.2026).

22. LinkedIn Learning. Огляд платформи онлайн-навчання, 2024. URL: https://www.linkedin.com/learning/ (дата звернення: 15.04.2026).

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра Технологій цифрового розвитку

Е-СЕРВІС НАВІГАЦІЇ ПРОФЕСІЙНОГО РОЗВИТКУ З ВИКОРИСТАННЯМ МОВИ ПРОГРАМУВАННЯ C#

Виконав студент 4 курсу
Групи ТЦР-41
Сич Марк Євгенійович
Керівник роботи
старший викладач кафедри
Читулян Вадим Олегович

Київ-2026

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи - автоматизація процесу планування професійного розвитку користувачів за допомогою розробленого е-сервісу навігації професійного розвитку та його інтеграції з платформами онлайн-навчання та пошуку роботи.

Об’єкт дослідження - процес планування професійного розвитку в ІТ-галузі.

Предмет дослідження - методи та засоби створення веб-систем для автоматизованого підбору навчальних курсів та вакансій.

ЗАДАЧІ КВАЛІФІКАЦІЙНОЇ РОБОТИ

1. Провести аналіз предметної області та огляд існуючих систем кар’єрного планування;
2. Дослідити сучасні методи веб-скрапінгу та інтеграції зовнішніх джерел даних;
3. Обґрунтувати вибір технологічного стеку для розробки системи;
4. Спроектувати архітектуру веб-додатку на основі патерну MVC;
5. Розробити модулі веб-скрапінгу для платформ ClassCentral та Robota.ua;
6. Реалізувати інтеграцію з API штучного інтелекту для генерації рекомендацій;
7. Створити адаптивний користувацький інтерфейс;
8. Провести тестування системи та впровадити її в експлуатацію.

АНАЛІЗ АНАЛОГІВ

Критерій порівняння	LinkedIn Learning	Coursera	Robota.ua	ClassCentral	Розроблена платформа
Спеціалізація на IT-кар'єрі	Часткова	Часткова	Часткова	Відсутня	**Висока**
Веб-скрапінг курсів і вакансій	Відсутній	Відсутній	Відсутній	Відсутній	**Реалізовано**
Інтеграція з AI (рекомендації)	Часткова	Часткова	Відсутня	Відсутня	**Cohere AI API**
Одночасний пошук курсів і вакансій	Відсутній	Відсутній	Відсутній	Відсутній	**Реалізовано**
Персоналізований кар'єрний план	Відсутній	Частковий	Відсутній	Відсутній	**Реалізовано**
Автономність від хмарних сервісів	Відсутня	Відсутня	Відсутня	Відсутня	**Повна (ASP.NET Core)**

ВИМОГИ ДО ПЛАТФОРМИ

Функціональні вимоги

1. Реєстрація та авторизація користувачів з можливістю відновлення доступу до акаунта.
2. Автоматизований веб-скрапінг навчальних курсів з платформи ClassCentral на основі введених користувачем ключових слів та параметрів фільтрації.
3. Автоматизований веб-скрапінг актуальних вакансій з платформи Robota.ua відповідно до професійного профілю користувача.
4. Генерація персоналізованих кар'єрних рекомендацій на основі отриманих даних з використанням Cohere AI API.
5. Формування індивідуального плану професійного розвитку з відображенням рекомендованих курсів та вакансій в єдиному інтерфейсі.
6. Збереження історії пошукових запитів та результатів для подальшого використання користувачем.

Нефункціональні вимоги

1. Реалізація серверної частини на базі ASP.NET Core 8.0 з використанням архітектурного патерну MVC для забезпечення масштабованості та підтримуваності системи.
2. Забезпечення коректної роботи модулів веб-скрапінгу з використанням Selenium WebDriver 4.25 та ChromeDriver 130.0 в режимі без графічного інтерфейсу (headless).
3. Час відповіді системи на запит користувача не повинен перевищувати 10 секунд за умови стабільного інтернет-з'єднання.
4. Адаптивний інтерфейс на базі Bootstrap 5.3, що забезпечує коректне відображення на пристроях з різною роздільною здатністю екрана.
5. Зберігання даних користувачів з дотриманням вимог безпеки: хешування паролів, захист від SQL-ін'єкцій та XSS-атак.

Програмні засоби реалізації

ASP.NET Core
Серверна платформа. C# 12, MVC патерн, Entity Framework Core

Selenium WebDriver
Веб-скрапінг ClassCentral та Robota.ua. ChromeDriver 130, headless

Cohere AI API
Генерація AI-рекомендацій. Модель command-a-03-2025

Bootstrap 5.3
Адаптивний UI. Font Awesome, CSS3, Razor Pages

SQL Server
База даних. Entity Framework Core, міграції, зберігання профілів

RestSharp
HTTP-запити до Cohere AI API. Обробка JSON-відповідей

Newtonsoft.Json
Серіалізація та десеріалізація JSON-даних. Версія 13.0.3

Visual Studio 2022
Середовище розробки. Git, відладка, NuGet-пакети

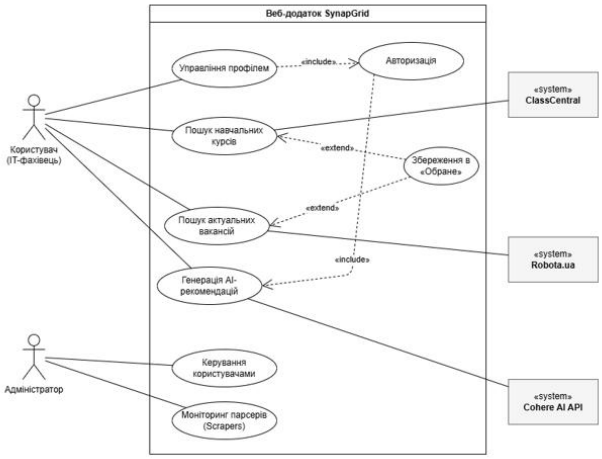
Інтегровані зовнішні платформи

ClassCentral
Агрегатор онлайн-курсів

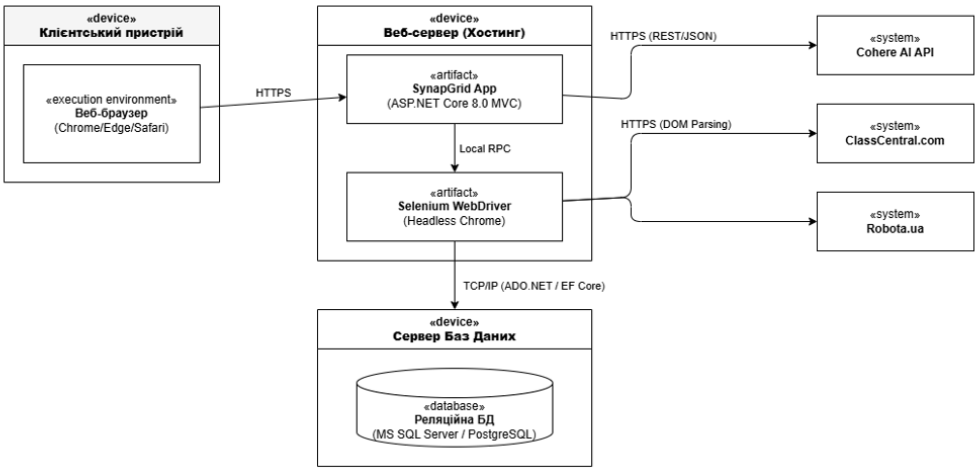
Robota.ua
Платформа пошуку вакансій

Cohere AI
Генерація кар'єрних рекомендацій

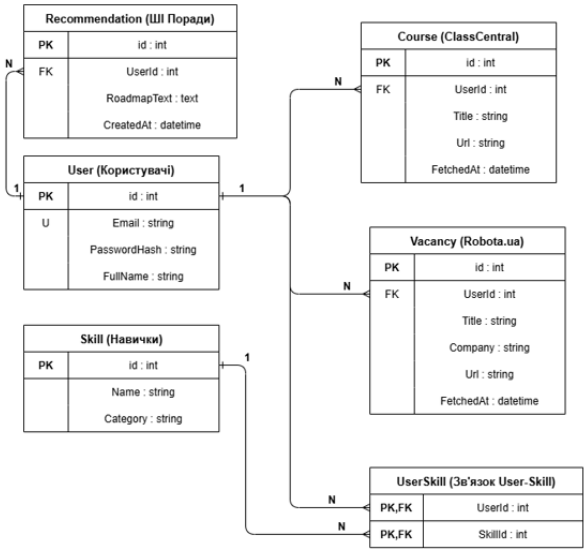
ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ ПЛАТФОРМИ



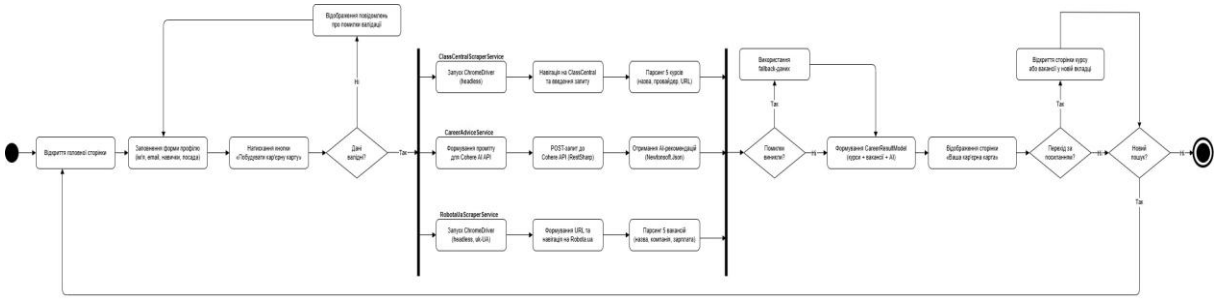
ДІАГРАМА СТРУКТУРИ БАЗИ ДАНИХ



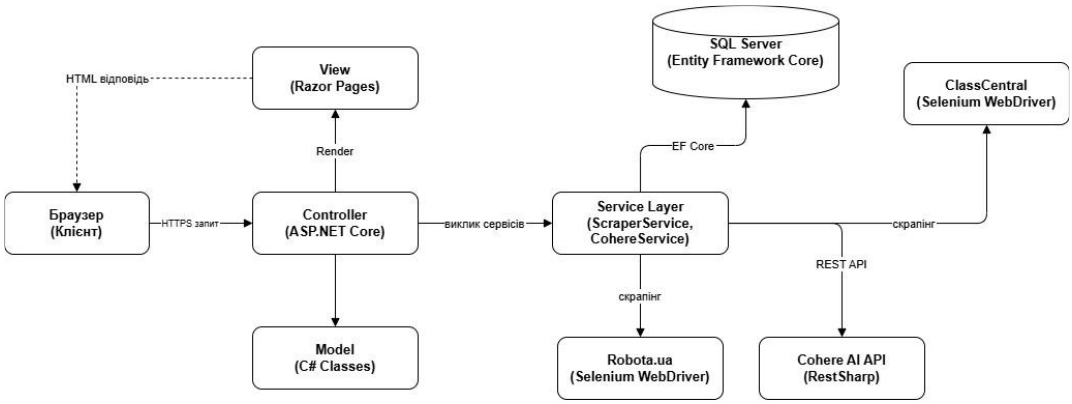
Діаграма структури бази даних (ERD)



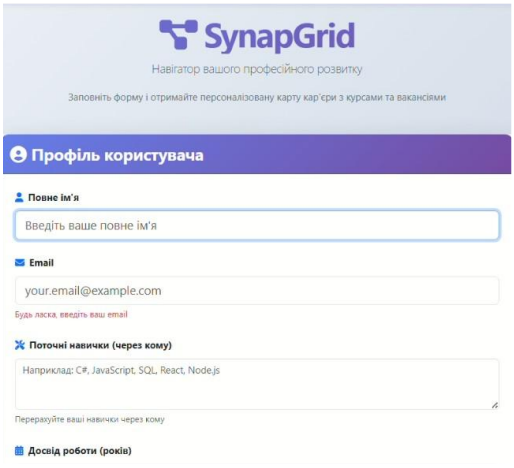
Діаграма діяльності

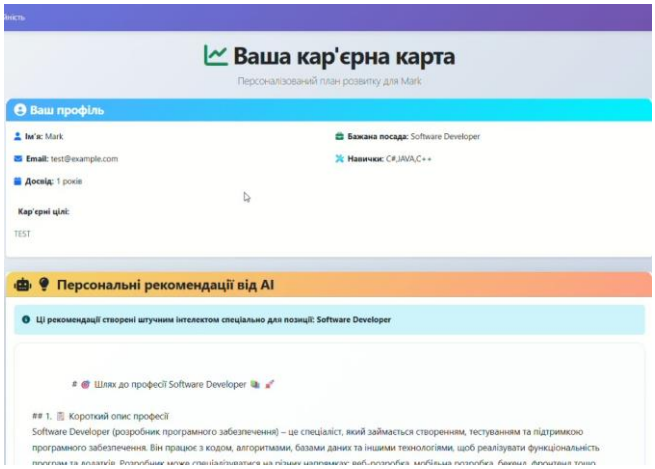


Діаграма архітектури системи MVC

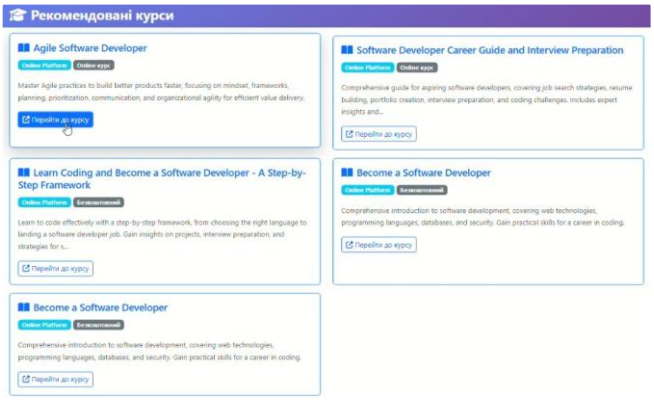


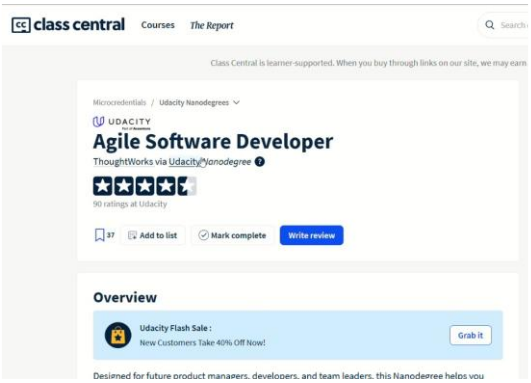
ЕКРАННІ ФОРМИ



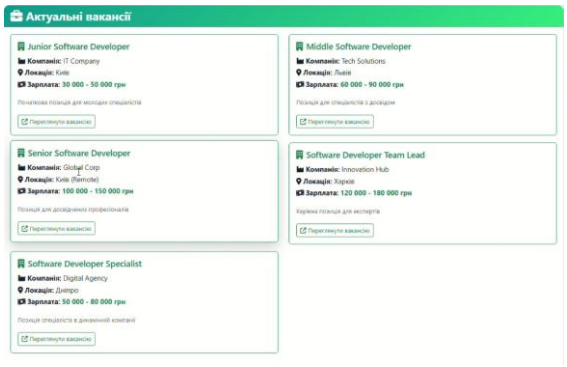


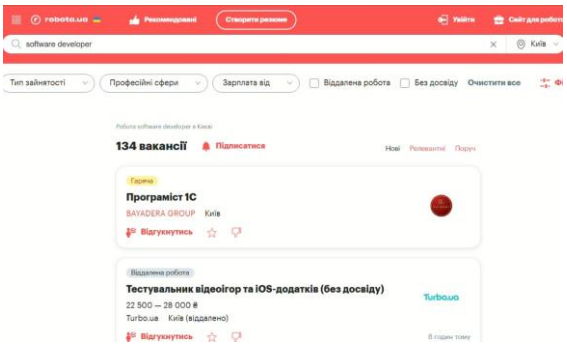
ЕКРАННІ ФОРМИ





ЕКРАННІ ФОРМИ





АПРОБАЦІЯ

1. Сич М. Є., Читулян В. О. Веб-додаток для автоматизованого планування кар’єрного розвитку в ІТ-сфері з використанням штучного інтелекту // VII Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в інформаційно-комунікаційних технолоіях». 29.04.2026 ДУІКТ, Київ, С.97-100

1. Сич М. Є., Читулян В. О. Веборієнтована система планування карєрного розвитку ІТ-фахівців з інтеграцією штучного інтелекту // ІІІ Всеукраїнська науково-практична конференція «Цифрова гуманістика: Інформаційні технології та інформаційне моделювання на сучасному етапі розвитку суспільства». 05.05.2026, ЦДУ ім. В. Винниченка, м. Кропивницький, С.147-148

ВИСНОВКИ

1. Проведено аналіз предметної області та існуючих систем кар’єрного планування, що підтвердило актуальність створення єдиної платформи для пошуку курсів та вакансій.

2. Обґрунтовано вибір технологічного стеку (ASP.NET Core, EF Core) та спроєктовано архітектуру веб-додатку SynapGrid на основі патерну MVC.

3. Розроблено модулі веб-скрапінгу (із використанням Selenium WebDriver) для автоматизованого збору даних у реальному часі з платформ ClassCentral та Robota.ua.

4. Реалізовано інтеграцію з мовною моделлю Cohere AI API для семантичного аналізу даних та генерації персоналізованих кар'єрних рекомендацій.

5. Створено адаптивний користувацький інтерфейс (на базі Bootstrap 5.3), який об'єднує результати пошуку та ШІ-поради в єдину індивідуальну карту розвитку.

6. Проведено тестування системи, яке підтвердило виконання всіх функціональних та нефункціональних вимог, стабільність роботи парсерів та готовність е-сервісу до експлуатації.

ДОДАТОК Б. Лістинги основних модулів програмного забезпечення

Б.1. Program.cs - точка входу додатку

```
using SynapGrid.Services;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllersWithViews();

builder.Services.AddScoped<CareerAdviceService>();
builder.Services.AddScoped<RobotaUaScraperService>();
```

```
var app = builder.Build();

if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Home/Error");
    app.UseHsts();
}

app.UseHttpsRedirection();
app.UseStaticFiles();

app.UseRouting();

app.UseAuthorization();

app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Home}/{action=Index}/{id?}");

app.Run();
```

Б.2. Модель UserProfileModel.cs

```
using System.ComponentModel.DataAnnotations;

namespace SynapGrid.Models
{
    public class UserProfileModel
    {
        [Required(ErrorMessage = "Будь ласка, введіть ваше ім'я")]
        [Display(Name = "Повне ім'я")]
        public string FullName { get; set; } = string.Empty;

        [Required(ErrorMessage = "Будь ласка, введіть ваш email")]
        [EmailAddress(ErrorMessage = "Невірний формат email")]
        [Display(Name = "Email")]
        public string Email { get; set; } = string.Empty;

        [Display(Name = "Поточні навички (через кому)")]
        public string CurrentSkills { get; set; } = string.Empty;

        [Display(Name = "Досвід роботи (років)")]
        [Range(0, 50, ErrorMessage = "Досвід має бути від 0 до 50 років")]
        public int YearsOfExperience { get; set; }

        [Required(ErrorMessage = "Будь ласка, оберіть бажану посаду")]
        [Display(Name = "Бажана посада")]
        public string DesiredPosition { get; set; } = string.Empty;

        [Display(Name = "Цілі кар'єрного розвитку")]
        public string CareerGoals { get; set; } = string.Empty;
    }
}
```

Б.3. Контролер HomeController.cs — метод ProcessCareerPath

```
using Microsoft.AspNetCore.Mvc;
using SynapGrid.Models;
using SynapGrid.Services;
using System.Diagnostics;

namespace SynapGrid.Controllers
{
    public class HomeController : Controller
    {
        private readonly ILogger<HomeController> _logger;
        private readonly CareerAdviceService _careerAdviceService;
        private readonly RobotaUaScraperService _vacancyService;

        public HomeController(
            ILogger<HomeController> logger,
            CareerAdviceService careerAdviceService,
            RobotaUaScraperService vacancyService)
        {
            _logger = logger;
            _careerAdviceService = careerAdviceService;
            _vacancyService = vacancyService;
        }

        public IActionResult Index()
        {
            return View();
        }

        [HttpPost]
        public async Task<IActionResult> ProcessCareerPath(UserProfileModel
model)
        {
            if (!ModelState.IsValid)
            {
                return View("Index", model);
            }

            try
            {
                var courseService = new ClassCentralScraperService();

                var coursesTask =
courseService.GetCoursesAsync(model.DesiredPosition);
                var vacanciesTask =
_vacancyService.GetVacanciesAsync(model.DesiredPosition);
                var careerAdviceTask =
_careerAdviceService.GetCareerAdviceAsync(model.DesiredPosition);

                await Task.WhenAll(coursesTask, vacanciesTask,
careerAdviceTask);

                var result = new CareerResultModel
                {
```

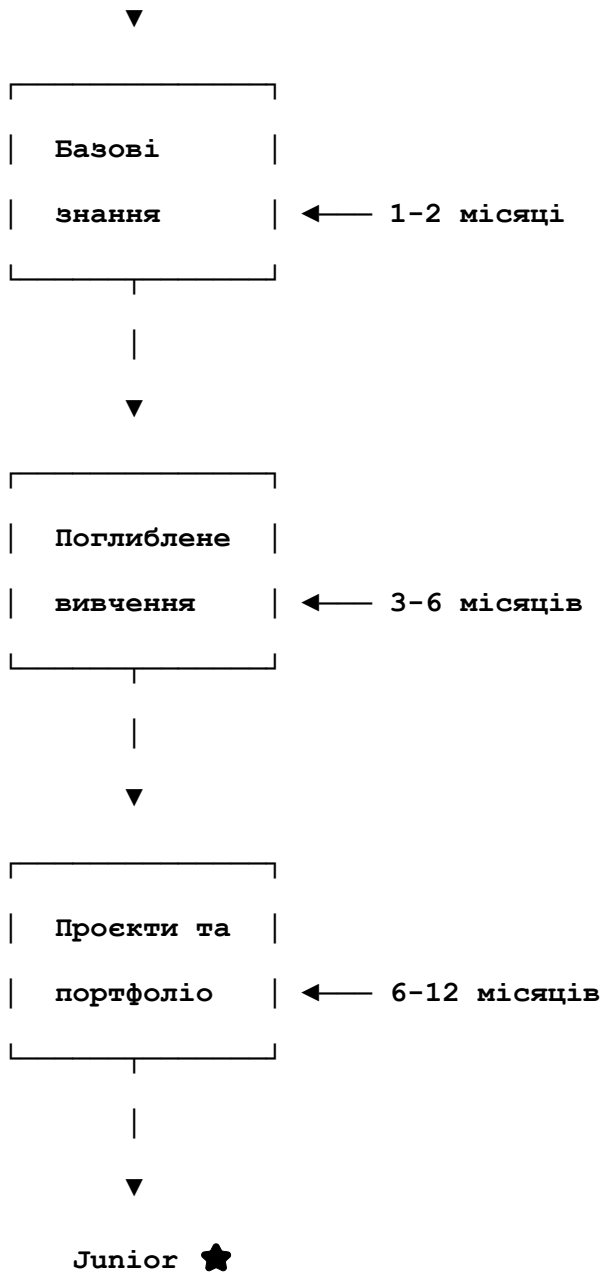
```
                    UserProfile = model,
                    Courses = await coursesTask,
                    Vacancies = await vacanciesTask,
                    CareerAdvice = await careerAdviceTask
                };

                return View("Results", result);
            }
            catch (Exception ex)
            {
                _logger.LogError(ex, "Помилка при обробці кар'єрного
шляху");
                ModelState.AddModelError("", "Виникла помилка при обробці
запиту. Спробуйте пізніше.");
                return View("Index", model);
            }
        }

        [HttpGet]
        [Route("api/career-advice/{position}")]
        public async Task<IActionResult> GetCareerAdvice(string position)
        {
            try
            {
                if (string.IsNullOrWhiteSpace(position))
                {
                    return BadRequest(new { error = "Должность не указана"
});
                }

                var advice = await
_careerAdviceService.GetCareerAdviceAsync(position);
                return Ok(new { position, advice });
            }
            catch (Exception ex)
            {
                _logger.LogError(ex, "Помилка при отриманння кар'єрних
рекомендацій");
                return StatusCode(500, new { error = "Виникла помилка при
отримання рекомендацій" });
            }
        }


        [HttpGet]
        [Route("api/career-tip/{position}")]
        public async Task<IActionResult> GetCareerTip(string position)
        {
            try
            {
                if (string.IsNullOrWhiteSpace(position))
                {
                    return BadRequest(new { error = "Посада не вказана" });
                }

                var tip = await
_careerAdviceService.GetShortCareerTipAsync(position);
                return Ok(new { position, tip });
```

```
            }
            catch (Exception ex)
            {
                _logger.LogError(ex, "Помилка при отриманння кар'єрного
поради");
                return StatusCode(500, new { error = "Виникла помилка при
отримання поради" });
            }
        }

        [HttpGet]
        [Route("api/course-recommendations/{position}")]
        public async Task<IActionResult> GetCourseRecommendations(string
position)
        {
            try
            {
                if (string.IsNullOrWhiteSpace(position))
                {
                    return BadRequest(new { error = "Посада не вказана" });
                }
                var recommendations = await
_careerAdviceService.GetCourseRecommendationsAsync(position);
                return Ok(new { position, recommendations });
            }
            catch (Exception ex)
            {
                _logger.LogError(ex, "Помилка при отриманні рекомендацій
курсу");
                return StatusCode(500, new { error = "Виникла помилка при
отримання рекомендацій" });
            }
        }
        public IActionResult Privacy()
        {
            return View();
        }
        [ResponseCache(Duration = 0, Location = ResponseCacheLocation.None,
NoStore = true)]
        public IActionResult Error()
        {
            return View(new ErrorViewModel { RequestId =
Activity.Current?.Id ?? HttpContext.TraceIdentifier });
        }
    }
}
```

Б.4. Сервіс CareerAdviceService.cs — метод GetCareerAdviceAsync

```
using System;
using System.Text;
using System.Threading.Tasks;
using Newtonsoft.Json.Linq;
using RestSharp;


namespace SynapGrid.Services
{
    public class CareerAdviceService
    {
        private readonly string _apiKey;


        public CareerAdviceService()
        {
            _apiKey = "IP0Qfrfq1PmzfZZbnx2eer1OZ9SFRYwTPIRvvMSP";
        }


        /// <summary>
        /// </summary>
        /// <param name="position">Назва посади/професії</param>
        /// <returns>Текст с рекомендаціями по навчанню і
розвитку</returns>
        public async Task<string> GetCareerAdviceAsync(string position)
        {
            try
            {
                if (string.IsNullOrWhiteSpace(position))
                {
                    throw new ArgumentException("Название должности не
может быть пустым", nameof(position));
```

```
                }

                // Формуємо детальный промпт для AI
                string prompt = $@"Ти - професійний кар'єрний консультант в IT сфері.
Людина хоче стати: {position}

Напиши детальну та структуровану інструкцію українською мовою, яка включає:

# 🎯 Шлях до професії {position} 🚀

## 1. 📝 Короткий опис професії
Опиши, чим займається спеціаліст на цій позиції

## 2. 🛠️ Необхідні навички та знання

### Технічні навички:
- Мови програмування, технології, інструменти 💻

### Soft skills:
- Комунікація, робота в команді 👥 🗣️

## 3. 🚀 План навчання (Дорожня карта)

ВАЖЛИВО: Додай ТЕКСТОВУ СХЕМУ-ДІАГРАМУ шляху навчання у вигляді ASCII-арту, наприклад:

```
    СТАРТ 🎓
      |
```
```




```

Детальний план:

- Що вивчати в першу чергу

- Що вивчати далі

- Рекомендовані курси та ресурси

## 4. 💼 Практичний досвід

- Рекомендації щодо практичних проєктів

- Де шукати стажування або першу роботу
```

```
## 5. ⏱️ Терміни навчання

Орієнтовні терміни для досягнення Junior рівня


## 6. 🚀 Кар'єрні перспективи

Можливості розвитку та зростання зарплати


Пиши зрозумілою мовою, структуровано, з конкретними рекомендаціями.
Використовуй емодзі для покращення сприйняття.";


                string answer = await AskCohereAsync(prompt);

                return answer;

            }

            catch (Exception ex)

            {

                Console.WriteLine($"❌ Помилка в GetCareerAdviceAsync:
{ex.Message}");

                return GetFallbackAdvice(position);

            }

        }


        /// <summary>

        /// Відправляє запит до Cohere API

        /// </summary>

        private async Task<string> AskCohereAsync(string question)

        {

            try

            {

                var options = new
RestClientOptions("https://api.cohere.com/v2/chat")

                {
```

```
    ThrowOnAnyError = false,
    Timeout = TimeSpan.FromSeconds(30)
};

var client = new RestClient(options);
var request = new RestRequest("", Method.Post);

request.AddHeader("Authorization", $"Bearer {_apiKey}");
request.AddHeader("Content-Type", "application/json");

var body = new
{
    stream = false,
    model = "command-a-03-2025",
    messages = new[]
    {
        new {
            role = "user",
            content = question
        }
    }
};

request.AddJsonBody(body);

Console.WriteLine("🤖 Відправка запиту до Cohere AI...");
var response = await client.ExecuteAsync(request);

if (!response.IsSuccessful)
{
```

```
                    throw new Exception($"HTTP помилка:
{response.StatusCode} - {response.ErrorMessage}");
                }

                var json = JObject.Parse(response.Content ?? "{}");
                var text =
json["message"]?["content"]?[0]?["text"]?.ToString();

                if (string.IsNullOrWhiteSpace(text))
                {
                    throw new Exception("Пуста відповідь від Cohere API");
                }

                Console.WriteLine("☑ Відповідь отримана від Cohere AI");
                return text;
            }
            catch (Exception ex)
            {
                Console.WriteLine($"❌ Помилка при зверненні до Cohere
API: {ex.Message}");
                throw;
            }
        }

        private string GetFallbackAdvice(string position)
        {
            return $@"# 🎯 Шлях до професії: {position}

## 📋 Опис професії
```

{position} - це затребувана спеціальність на сучасному IT-ринку, яка вимагає як технічних навичок, так і розуміння бізнес-процесів.

🛠 Необхідні навички

Технічні навички:

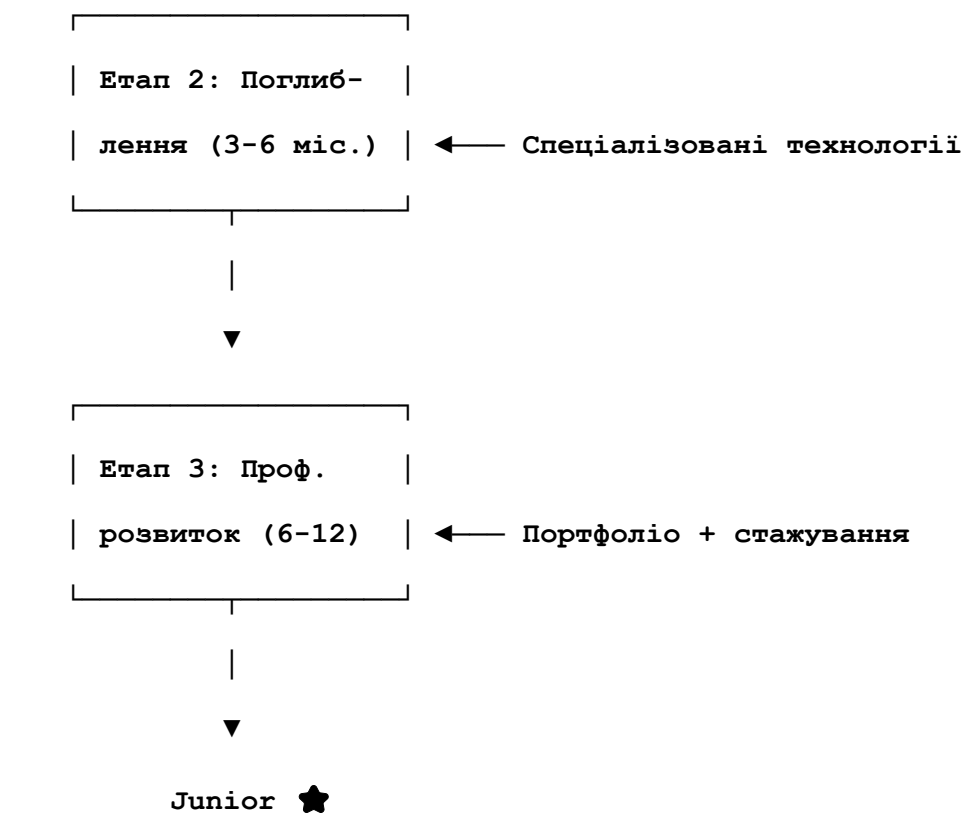
- Основи програмування та алгоритмів 💻
- Знання сучасних технологій та інструментів 🔧
- Розуміння принципів розробки ПЗ 📱
- Робота з системами контролю версій (Git) 🔄

Soft Skills:

- Комунікативні навички 🗣️
- Робота в команді 👥
- Самонавчання 📖
- Аналітичне мислення 🧠

🗺️ План навчання (Дорожня карта)

```
        СТАРТ 🎓
          |
          ▼
   ┌──────────────────┐
   |  Етап 1: Основи  |
   |   (1-2 місяці)   | ◄── Базові концепції
   └──────────┬───────┘
              |
              ▼
```




```

### Етап 1: Основи (1-2 місяці)

- Вивчіть базові концепції
- Освойте основні інструменти
- Практикуйтесь на простих завданнях

### Етап 2: Поглиблення (3-6 місяців)

- Вивчайте спеціалізовані технології
- Розробляйте навчальні проєкти
- Беріть участь у open-source проєктах

### Етап 3: Професійний розвиток (6-12 місяців)

- Створюйте портфоліо проєктів
- Готуйтесь до співбесід
- Шукайте стажування та junior позиції
```

💼 Практичний досвід

- Розробіть 3-5 власних проєктів для портфоліо 🗂️
- Беріть участь у хакатонах та конкурсах ⏳
- Шукайте стажування в IT-компаніях 🏢
- Практикуйтесь на платформах для програмістів 🥷

⏱️ Терміни навчання

Для досягнення Junior рівня зазвичай потрібно 6-12 місяців інтенсивного навчання за умови щоденної практики 4-6 годин.

🚀 Кар'єрні перспективи

- **Junior**: стартовий рівень, активне навчання 🍸
- **Middle**: 2-3 роки досвіду, самостійна робота 🦾
- **Senior**: 5+ років досвіду, експертні знання 🎓
- **Tech Lead / Architect**: управління командою або архітектурою 👑

📖 Рекомендовані ресурси

- **Онлайн-курси**: Coursera, Udemy, Pluralsight
- **Книги** за спеціальністю 📚
- **Офіційна документація** технологій 📄
- **YouTube канали** з навчальними матеріалами 📽️
- **Практичні платформи**: LeetCode, HackerRank, Codewars 💻

💰 Зарплатні очікування в Україні

- **Junior**: 30,000 - 60,000 грн
- **Middle**: 70,000 - 120,000 грн
- **Senior**: 130,000 - 200,000+ грн

```
---

💡 **Порада**: Головне - це постійність та практика! Кожного дня приділяйте час вивченню та практиці, і результат не змусить себе чекати.

⚠️ *Примітка: Це загальні рекомендації. Для отримання більш детальних порад рекомендується звернутися до кар'єрного консультанта.*";
        }

        public async Task<string> GetShortCareerTipAsync(string position)
        {
            try
            {
                string prompt = $@"Дай одну коротку пораду (максимум 2 речення) людині, яка хоче стати {position}.
Фокусуйся на найважливішій навичці або дії. Відповідай українською мовою.";

                string answer = await AskCohereAsync(prompt);

                if (answer.Length > 200)
                {
                    answer = answer.Substring(0, 197) + "...";
                }

                return answer;
            }
            catch (Exception)
            {
                return $"Для позиції {position} важливо постійно вчитися та практикуватися.";
            }
```

```
        }

        public async Task<string> GetCourseRecommendationsAsync(string
position)
        {
            try
            {
                string prompt = $@"Порекомендуй конкретні онлайн-курси та
платформи для вивчення професії {position}.
Вкажи:
1. Назви конкретних курсів (якщо знаєш)
2. Платформи для навчання (Coursera, Udemy, YouTube канали тощо)
3. Безкоштовні ресурси

Відповідай структуровано та коротко, українською мовою.";

                string answer = await AskCohereAsync(prompt);
                return answer;
            }
            catch (Exception)
            {
                return $@"Рекомендовані платформи для вивчення {position}:
- Coursera - академічні курси 🎓
- Udemy - практичні курси 💻
- YouTube - безкоштовні відео-уроки 📺
- Офіційна документація 📄
- Спеціалізовані форуми та спільноти 👥";
            }
        }
    }
```

```
}
```

Б.5. Файл SynapGrid.csproj — залежності проєкту

```
<Project Sdk="Microsoft.NET.Sdk.Web">

  <PropertyGroup>
    <TargetFramework>net8.0</TargetFramework>
    <Nullable>enable</Nullable>
    <ImplicitUsings>enable</ImplicitUsings>
  </PropertyGroup>

  <ItemGroup>
    <PackageReference Include="Selenium.WebDriver" Version="4.25.0" />
    <PackageReference Include="Selenium.WebDriver.ChromeDriver"
Version="130.0.6723.9100" />
    <PackageReference Include="Selenium.Support" Version="4.25.0" />
    <PackageReference Include="RestSharp" Version="112.1.0" />
    <PackageReference Include="Newtonsoft.Json" Version="13.0.3" />
  </ItemGroup>

</Project>
```