Хмельницький національний університет
Факультет інформаційних технологій
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

Інтерактивна система для підбору фільмів
на основі емоційного стану користувача «MOVI.FLOW-ER»
Назва теми

Рівень вищої освіти Перший (бакалаврський)

Галузь знань 12 «Інформаційні технології»

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

Шифр КвРІПЗ. 2201107.01.13.ПЗ

Виконав студент IV курсу, група ІПЗ-22-1 _____ Максим ПРИЙМАК
Підпис Ім'я, ПРІЗВИЩЕ

Керівник канд. пед. наук, доцент _____ Наталія ПРАВОРСЬКА
Науковий ступінь, звання Підпис Ім'я, ПРІЗВИЩЕ

Нормоконтролер канд. техн. наук, доцент _____ Оксана ЯШИНА
Посада Підпис Ім'я, ПРІЗВИЩЕ

До захисту допускаю:
Завідувач кафедри інженерії
програмного забезпечення _____ Леонід БЕДРАТЮК
Підпис Ім'я, ПРІЗВИЩЕ

1 червня 2026 р.

Хмельницький 2026

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій
Кафедра Інженерії програмного забезпечення
Рівень вищої освіти Перший (бакалаврський)
Галузь знань 12 «Інформаційні технології»
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри ІПЗ
Л. П. Бедратюк
02.01.2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Приймаку Максиму Олександровичу
Прізвище, ім'я, по батькові студента

1. Тема роботи Інтерактивна система для підбору фільмів на основі емоційного стану користувача «MOVI.FLOW-ER»

Керівник роботи Праворська Наталія Іванівна, канд. пед. наук, доцент
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 20.01.2026 р. № 7-КП
2. Строк подання студентом роботи на кафедру 01.06.2026 р.
3. Вихідні дані до роботи Матеріали переддипломної практики
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити) Дослідження предметної області та постановка задач, проєктування програмного забезпечення, програмна реалізація та тестування застосунку.
5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)

Три креслення
1. Діаграма варіантів використання
2. Діаграма зв'язків модулів
3. Діаграма розгортання

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Яшина О. М., доцент	05.05.2026	25.05.26
Антиплагіат	Форкун Ю. В., доцент		

7. Дата видачі завдання « 2 » січня 2026 р.

КАЛЕНДАРНИЙ ПЛАН

Назва етапів (розділів) кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1 Ознайомлення з тематикою дипломного проєктування, визначення та узгодження індивідуальної теми кваліфікаційної роботи (КвР)	01.12– 31.12.2026	
2 Збір матеріалу за темою КвР; дослідження предметної області, в якій планується використання програмного забезпечення (ПЗ), визначення задач та вимог.	01.01 – 20.02.2026	
3 Проєктування програмного забезпечення	21.02 – 20.03 2026	
4 Програмна реалізація з використанням відповідних засобів розроблення. Тестування ПЗ	21.03 – 30.04.2026	
5 Написання вступу, загальних висновків, оформлення переліку джерел посилання та додатків. Оформлення пояснювальної записки КвР згідно вимог	01.05 – 25.05.2026	
6 Попередній захист КвР	Травень 2026	Згідно графіка
7 Перевірка КвР на плагіат, нормоконтроль, отримання відгуків, рецензій та інших супровідних документів. Брошурування (зшиття) пояснювальної записки.	26.05 – 30.05.2026	
8 Здача КвР на кафедру; підготовка КвР для розміщення у репозитарії ХНУ; підготовка до захисту та захист КвР	з 01.06.2026	

Студент _____ Максим ПРИЙМАК
Підпис Ім'я, ПРІЗВИЩЕ

Керівник роботи _____ Наталія ПРАВОРСЬКА
Підпис Ім'я, ПРІЗВИЩЕ

АНОТАЦІЯ

Тема кваліфікаційної роботи: Інтерактивна система для підбору фільмів на основі емоційного стану користувача «MOVI.FLOW-ER».

Автор роботи: Приймак Максим Олександрович

Керівник роботи: Праворська Наталія Іванівна

Пояснювальна записка: 59 с., 14 рис., 7 табл., 6 дод., 42 джерел.

Графічна частина: 3 креслення.

ІНТЕРАКТИВНА СИСТЕМА, ВЕБЗАСТОСУНОК, ASP.NET CORE MVC, ПЕРСОНАЛІЗАЦІЯ КОНТЕНТУ, ЕМОЦІЙНИЙ СТАН КОРИСТУВАЧА, ПІДБІР ФІЛЬМІВ, КЛІЄНТ-СЕРВЕРНА АРХІТЕКТУРА, ENTITY FRAMEWORK CORE, БАЗА ДАНИХ SQL SERVER, TMDB API.

Метою роботи є розроблення інтерактивної системи для підбору фільмів, яка відповідатиме сучасним вимогам розробки та тенденціям створення рекомендаційних систем. Запропонована система орієнтована на врахування емоційного стану користувача з метою підвищення рівня персоналізації та покращення користувацького досвіду взаємодії з мультимедійним контентом.

У кваліфікаційній роботі проведено аналіз предметної області персоналізованих рекомендаційних систем та їх інформаційного забезпечення, визначено функціональні та нефункціональні вимоги до даного типу проєкту. Розроблено архітектуру вебсистеми, спроєктовано структуру основних модулів, базу даних, а також інтерфейс користувача з урахуванням принципів адаптивності.

Для реалізації системи використано мову програмування C#, технології HTML та CSS, а також фреймворк ASP.NET Core MVC із застосуванням шаблонів Razor. Інтеграція з зовнішнім джерелом даних TMDb API забезпечує отримання актуальної інформації про кінотвори. У результаті проєктування здійснено програмну реалізацію інтерактивної системи «MOVI.FLOW-ER», яка надає персоналізовані рекомендації кінотворів та може слугувати базовою платформою для подальшого розвитку інтелектуальних емоційно-орієнтованих рекомендаційних сервісів.

27.05.2026
Дата

Підпис

ВІДОМІСТЬ ДОКУМЕНТІВ

№ рядка	Формат	Позначення документа	Найменування документа	К-сть аркушів	№ екз.	Примітка
			Текстові документи			
1	А4	КвРІПЗ. 2201107.01.13.ПЗ	Пояснювальна записка	59		
2	А4		Завдання на кваліфіка-	1		
			ційну роботу			
3	А4		Анотація	1		
			Графічні документи			
4	А4		Презентаційні матеріали	17		
			(слайди)			
5	А3	КвРІПЗ. 2201107.01.13.Е8	Діаграма варіантів	1		
			використання			
6	А3	КвРІПЗ. 2201107.01.13.Е8	Діаграма зв'язків	1		
			модулів			
7	А3	КвРІПЗ. 2201107.01.13.Е8	Діаграма розгортання	1		

Змн.	Арк.	№ докум.	Підпис	Дата	КвРІПЗ. 2201107.01.13.ВД		
Виконав		Приймак М. О.		27.05.26	Інтерактивна система для підбору фільмів на основі емоційного стану користувача «MOVI.FLCW-ER» Відомість документів	Літ.	Арк. 1 / Акрушів 1
Керівник.		Праворська Н. І.		27.05.26			
Рецензент						ХНУ, ІПЗ-22-1	
Н. Контр.		Яшина О. М.		27.05.26			
Зав. каф.		Бедратюк Л.П.		27.05.26			

ЗМІСТ

ВСТУП

Розвиток інформаційних технологій характеризується активним впровадженням систем у різні сфери діяльності людини. Вебзастосунки стали невід'ємною частиною повсякденного життя, забезпечуючи доступ до інформації, сервісів та мультимедійного контенту незалежно від місця перебування користувача. Особливу роль у цьому процесі відіграють інтерактивні платформи, орієнтовані на персоналізацію взаємодії та адаптацію до індивідуальних потреб різних користувачів.

Однією з найбільш динамічних і популярних галузей цифрових сервісів є платформи для споживання мультимедійного контенту, зокрема кінотворів. Кількість доступних фільмів, серіалів, мультфільмів та аніме постійно зростає, що ускладнює процес вибору контенту для перегляду. Користувачі все частіше стикаються з проблемою перевантаження інформацією, витрачаючи значний час на пошук відповідного кінотвору. Саме такі умови особливої актуальності набувають рекомендаційні вебсистеми, здатні автоматизувати процес підбору контенту та підвищити зручність користування мультимедійними платформами.

Традиційні системи є ефективними, але вони не завжди враховують суб'єктивні чинники, зокрема емоційний стан користувача, який відіграє важливу роль у виборі контенту для перегляду. Один і той самий користувач може надавати перевагу різним типам кінотворів залежно від настрою, рівня втоми чи психологічного стану. Тому виникає потреба у створенні вебсистем нового покоління, які здатні поєднувати технічні параметри вибору з емоційно-орієнтованими характеристиками.

Актуальність розроблення інтерактивної вебсистеми підбору фільмів на основі стану користувача зумовлена зростанням інтересу до персоналізованих цифрових сервісів. Користувачі очікують індивідуального підходу, інтуїтивно зрозумілого інтерфейсу та швидкого отримання релевантних результатів. Інтеграція таких системи із зовнішніми базами даних мультимедійного контенту, а саме TMDb API, відкриває широкі можливості для створення актуальних та

інформативних рекомендаційних платформ [1].

Розроблення системи «MOVI.FLOW-ER» спрямоване на вирішення зазначених проблем шляхом створення інтерактивного середовища, яке забезпечує персоналізований підбір кінотворів відповідно до емоційного стану користувача. Система орієнтована на формування рекомендацій з урахуванням психологічних аспектів вибору контенту. Такий підхід дозволяє скоротити час на пошук фільмів і підвищити якість користувацького досвіду.

Використання сучасних технологій веброзробки, зокрема фреймворку ASP.NET Core MVC, мови програмування C# та шаблонів Razor, забезпечує надійність, масштабованість і зручність підтримки розроблюваної системи. Крім того, реалізація інтерактивного інтерфейсу сприяє підвищенню залученості користувачів та створенню приємного візуального середовища для взаємодії з контентом всієї системи.

Метою кваліфікаційної роботи є розроблення інтерактивної вебсистеми для персоналізованого підбору фільмів на основі емоційного стану користувача, яка відповідатиме сучасним вимогам веброзробки та рекомендаційних систем.

Для досягнення поставленої мети у розробці вимагається здійснити такі завдання проєктування:

- провести аналіз предметної області персоналізованих вебсистем;
- дослідити існуючі платформи підбору мультимедійного контенту;
- сформувати функціональні та нефункціональні вимоги до системи;
- розробити архітектуру системи та взаємодію її основних компонентів;
- спланувати структуру бази даних та модулів системи;
- обрати технології та інструменти реалізації;
- реалізувати програмну частину проєкту;
- виконати візуальну частину даної системи;
- провести тестування функціональності та стабільності роботи проєкту;
- оцінити результат та перспективи подальшого розвитку вебсистеми.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Змістовий аналіз предметної області

Стрімкий розвиток інформаційних технологій та цифрових комунікацій зумовив суттєві зміни у способах споживання мультимедійного контенту. Протягом останніх років спостерігається стійка тенденція переходу користувачів від традиційних форматів перегляду кінотворів, таких як телебачення або фізичні носії, до вебсервісів. Фільми, серіали, анімаційні стрічки та аніме стали однією з ключових форм дозвілля для різних вікових категорій користувачів. За даними міжнародних аналітичних організацій, кількість користувачів відеостримінгових сервісів щороку зростає, а ринок онлайн-відео демонструє стабільне збільшення обсягів споживання контенту [2]. Це свідчить про високий попит на цифрові сервіси, що забезпечують доступ до великих бібліотек кінотворів та інформації.

Онлайн-перегляд фільмів та серіалів поступово витісняє традиційні канали демонстрації різного контенту. Користувачі очікують не лише широкого вибору кінотворів, а й швидкого доступу до них, зручної навігації та можливості приймати рішення без значних часових витрат. Дані умови сприяють для значного зростання ролі інформаційних систем, які виконують довідкову функцію та допомагають користувачеві орієнтуватися у великому обсязі доступного контенту. Саме тому системи підбору та рекомендацій фільмів набувають особливої актуальності, вони спрямовані на скорочення часу пошуку та підвищення якості позитивного користувацького досвіду.

Однією з головних проблем сучасного цифрового середовища є явище інформаційного перевантаження. Платформи, що надають доступ до мультимедійного контенту, містять тисячі одиниць кінотворів різних жанрів, форматів і років випуску. Частина користувачів витрачає значну кількість часу саме на вибір фільму або серіалу, а не на безпосередній перегляд. Саме це явище породжує потребу у спеціальних рекомендаційних системах, які допомагають користувачам швидше знаходити контент відповідно до їхніх інтересів.

Популярні рекомендаційні системи здебільшого базуються на жанрових уподобаннях, рейтингах, популярності контенту або історії попередніх переглядів [3]. Подібний підхід використовується такими відомими сервісами, як IMDb, Letterboxd та іншими інформаційними платформами. Багато випадків, де вибір кінотвору залежить не лише від жанру чи рейтингу, а й від емоційного стану, рівня втоми, стресу або бажаного емоційного ефекту від перегляду. Така ситуація сильно впливає на загальне враження від користування сервісами та рівень задоволення. Відповіддю на це активно розвиваються рекомендаційні системи, які аналізують різні параметри користувача та пропонують релевантний контент.

Емоційний аспект споживання кінотворів відіграє важливу роль у формуванні користувацьких рішень. Дослідження, опубліковані American Psychological Association, підтверджують, що користувачі часто обирають фільми та серіали відповідно до власного настрою або психологічних потреб. Це свідчить про доцільність використання емоційно-орієнтованих підходів у рекомендаційних системах, які здатні адаптувати добір контенту до поточного стану користувача.

У зв’язку з цим зростає значення інтерактивних вебсистем, що забезпечують активну взаємодію між користувачем та інтерфейсом. Сучасні вебплатформи орієнтовані на створення персоналізованого середовища, яке дозволяє користувачеві формувати власні списки перегляду, зберігати обраний контент, оцінювати кінотвори та отримувати рекомендації в режимі реального часу. Інтерактивність таких систем сприяє підвищенню залученості користувачів та формує позитивний досвід взаємодії з цифровим сервісом. Системи здатні не лише скоротити час вибору кінотворів, а й підвищити якість користувацького досвіду.

Важливою складовою предметної області є використання офіційних відкритих джерел даних мультимедійного контенту. Одним із найбільш поширених та авторитетних сервісів є The Movie Database, який надає структуровану інформацію про фільми, серіали, мультфільми та аніме, включаючи описи, жанри, рейтинги, тривалість і акторський склад. Інтеграція з такими API дозволяє системам забезпечувати актуальність системи та автоматизувати процес наповнення бази даних.

Аналіз предметної області демонструє, що більшість існуючих сервісів зосереджені або на наданні довідкової інформації про кінотворів, або на використанні узагальнених рекомендаційних алгоритмів, що базуються переважно на історії переглядів чи популярності контенту. Водночас такі підходи здебільшого не враховують ситуативні фактори, зокрема емоційний стан користувача в конкретний момент часу, що суттєво впливає на сприйняття мультимедійного контенту та задоволеність від перегляду. Відсутність емоційно-орієнтованого механізму підбору знижує рівень персоналізації та не забезпечує повної адаптації системи до індивідуальних потреб користувача [4].

Дана ситуація створює передумови для розроблення нових інтерактивних вебсистем, які поєднують інструменти персоналізації, аналіз користувацьких параметрів та врахування емоційного контексту під час формування рекомендацій. Інтеграція механізмів вибору емоційного стану з алгоритмами підбору контенту дозволяє підвищити відповідність результатів, зробити процес вибору більш інтуїтивним та зменшити час пошуку відповідного кінотвору.

Таким чином, предметна область розроблюваного проєкту охоплює онлайн-взаємодію з мультимедійним контентом, побудову персоналізованих рекомендаційних систем, обробку та аналіз емоційного стану користувача, реалізацію інтерактивних користувацьких інтерфейсів, а також інтеграцію з офіційними зовнішніми базами даних кінотворів для отримання актуальної та структурованої інформації [5]. Окрему увагу приділено забезпеченню логічної узгодженості між модулями системи, збереженню історії взаємодії користувача та можливості подальшого розширення функціоналу.

Проведений змістовний аналіз підтверджує актуальність створення вебсистеми «MOVI.FLOW-ER», яка спрямована на підвищення якості користувацького досвіду, оптимізацію процесу вибору кінотворів відповідно до поточного емоційного стану користувача та впровадження більш гнучкого підходу до персоналізації рекомендаційного механізму.

1.2 Аналіз наявного програмно-технічного забезпечення предметної області

Стрімкий розвиток цифрових медіа та сервісів забезпечує активне зростання кількості вебплатформ, що надають доступ до перегляду та підбору фільмів. Популярність фільмів, серіалів, анімаційних стрічок та аніме як основних форм дозвілля зумовила появу великої кількості спеціалізованих систем, орієнтованих на споживання мультимедійного контенту в режимі онлайн. Користувачі все частіше надають перевагу цифровим платформам, які дозволяють швидко обрати та переглянути кінотвор без прив'язки до часу трансляції або фізичних носіїв.

Зростання потреби вебтехнологій та стрімінгових сервісів призвів до формування конкурентного середовища, у якому різні платформи пропонують власні підходи до організації контенту, навігації, рекомендацій і взаємодії з користувачем [6]. Сучасні вебсистеми для перегляду кінотворів відрізняються за функціональними можливостями, моделями доступу до контенту, рівнем персоналізації, підтримкою локалізації та використанням рекомендаційних алгоритмів. Для ефективного проєктування інтерактивної вебсистеми підбору фільмів необхідно дослідити наявні рішення, які вже функціонують на ринку та користуються популярністю серед широкої аудиторії.

Аналіз таких платформ дає змогу виявити основні тенденції розвитку предметної області, оцінити рівень зручності інтерфейсу, гнучкість систем фільтрації та рекомендацій, а також підходи до роботи з мультимедійними базами даних. Окрему увагу доцільно приділити тому, наскільки існуючі вебсистеми враховують індивідуальні потреби користувачів та їхній емоційний контекст під час вибору контенту. У межах даного підрозділу було охоплено аналіз п'яти популярних вебсистем для вибору та перегляду кінотворів. Таке порівняння дозволяє комплексно оцінити сучасний стан програмно-технічного забезпечення предметної області та сформувати обґрунтовані вимоги до розроблюваної вебсистеми «MOVI.FLOW-ER».

Першою платформою для аналізу доцільно розглянути стримінговий сервіс Netflix, який є одним із найбільших та найвпливовіших гравців на світовому ринку

онлайн-відео. Дана платформа надає користувачам доступ до широкої бібліотеки фільмів, серіалів, анімаційних стрічок та оригінального контенту власного виробництва. Netflix функціонує за моделлю платної підписки та орієнтований на персоналізований перегляд контенту [7]. Головна сторінка сервісу відзначається мінімалістичним дизайном, який зосереджує увагу користувача безпосередньо на відеоконтенті та рекомендаціях. На наступному Рисунку 1.1 зображено головну сторінку даного сайту.

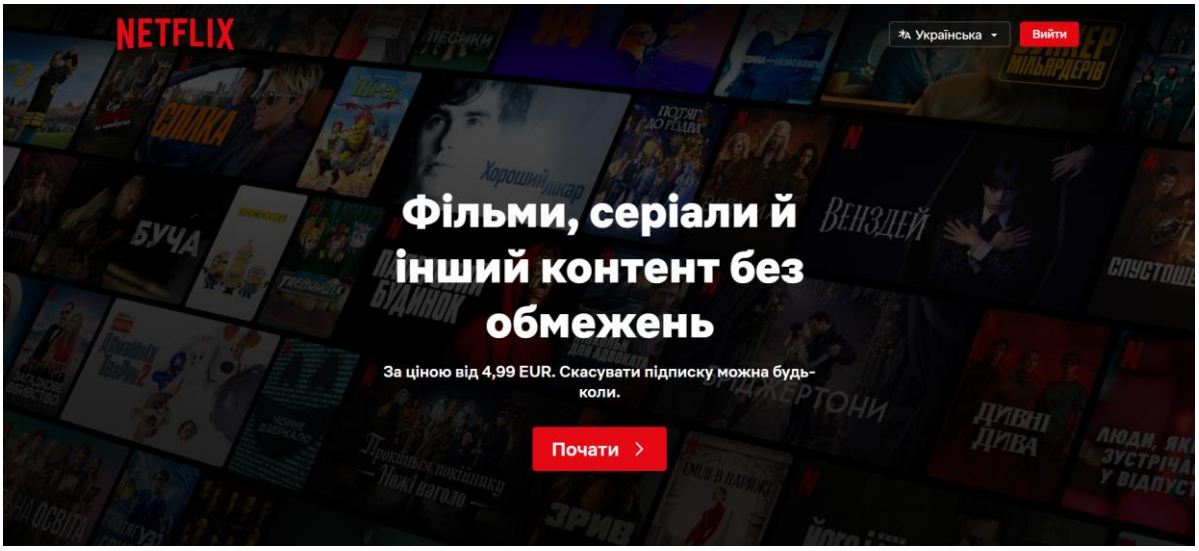


Рисунок 1.1 – Головна сторінка сайту Netflix

Навігація даної платформи побудована таким чином, щоб користувач міг швидко обрати цікавий кінотвор за жанром, популярністю, тематичними добірками або персональними рекомендаціями. Інтерфейс системи структуровано за принципом горизонтальних категорій та динамічних підбірок, що оновлюються відповідно до активності користувача, завдяки чому забезпечується зручність та логічна послідовність перегляду різного контенту. Додатково реалізовано механізми адаптивного відображення, які враховують попередні вподобання та історію взаємодії, що сприяє кращій взаємодії користувачів. Такий підхід дозволяє скоротити час пошуку фільмів, а також підвищити рівень залученості користувачів, створюючи індивідуальний досвід споживання мультимедійних кінотворів.

Перевагами Netflix є наступні складові:

— великий обсяг ліцензованого та оригінального контенту;

— високий рівень персоналізації рекомендацій;

— стабільна робота сервісу на різних платформах;

— сучасний та зручний інтерфейс користувача.

Саме до недоліків можна віднести такі елементи системи:

— відсутність безкоштовного доступу;

— обмеження контенту залежно від регіону;

— відсутність соціальних елементів взаємодії між користувачами.

Наступним об’єктом аналізу є стримінгова платфома Disney+, яка спеціалізується на контенті від компаній Disney, Pixar, Marvel, Star Wars та National Geographic [8]. Даний сервіс орієнтований на сімейну аудиторію та на шанувальників великих кінематографічних франшиз. Інтерфейс платформи побудований навколо брендованих розділів, що спрощує навігацію та пошук контенту. Головну сторінку даної системи можна побачити на Рисунку 1.2.

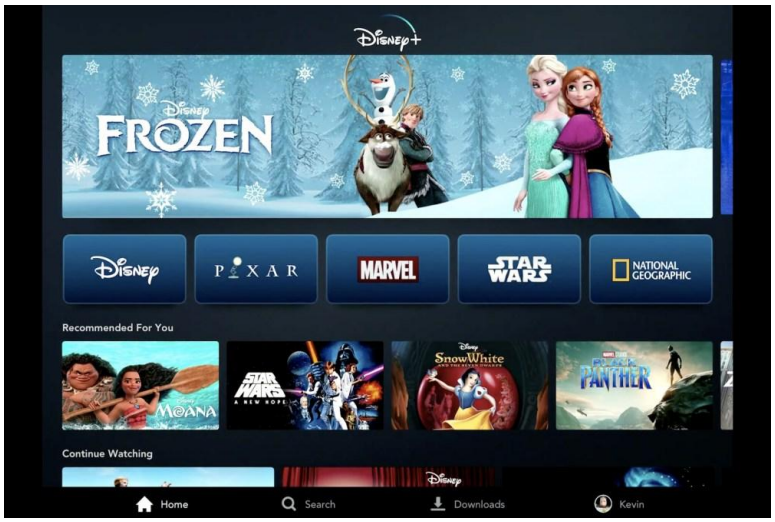


Рисунок 1.2 – Головна сторінка Disney+

Disney+ активно використовує виразні візуальні елементи та великі обкладинки, що робить процес вибору фільмів інтуїтивно зрозумілим і привабливим для користувача. Інтерфейс платформи побудований з акцентом на брендові колекції, що дозволяє швидко орієнтуватися у тематичних розділах та

формує чітку структуру навігації. Платформа підтримує створення кількох профілів для різних членів сім'ї та передбачає окремі налаштування для дитячого контенту, зокрема вікові обмеження та спеціальний безпечний режим перегляду. Це робить сервіс орієнтованим на сімейну аудиторію та забезпечує контроль доступу до контенту.

Серед переваг Disney+ можна виділити наступні елементи**:**

— чітка тематична структуризація контенту, що спрощує навігацію;

— висока якість відео та звуку;

— зручний та мінімалістичний інтерфейс для сімейного використання;

— наявність ексклюзивного контенту.

Також з недоліків слід зазначити такі опції**:**

— обмежена жанрова різноманітність;

— відсутність розширених фільтрів пошуку за детальними параметрами;

— менш розвинена система персоналізованих рекомендацій.

Таким чином, Disney+ демонструє ефективну модель сімейно-орієнтованої платформи з акцентом на брендований та ексклюзивний контент, що забезпечує впізнаваність сервісу та формує лояльну аудиторію. Орієнтація на власні медіафраншизи та безпечне середовище перегляду робить платформу особливо привабливою для сімей з дітьми. Водночас сервіс поступається універсальним стримінговим платформам у гнучкості персоналізації, різноманітності жанрового наповнення та глибині рекомендаційних механізмів, що може обмежувати інтерес користувачів платформи із більш широкими або специфічними кінематографічними вподобаннями.

Третьою платформою для аналізу є український вебсервіс UAKINO, який надає користувачам можливість перегляду фільмів та серіалів онлайн без обов'язкової реєстрації [9]. Даний ресурс орієнтований переважно на україномовну аудиторію та містить великий обсяг локалізованого контенту. Головна сторінка платформи має класичну структуру з переліком новинок, популярних фільмів та жанрових категорій. Головну сторінку даного сайту зображено на Рисунку 1.3.

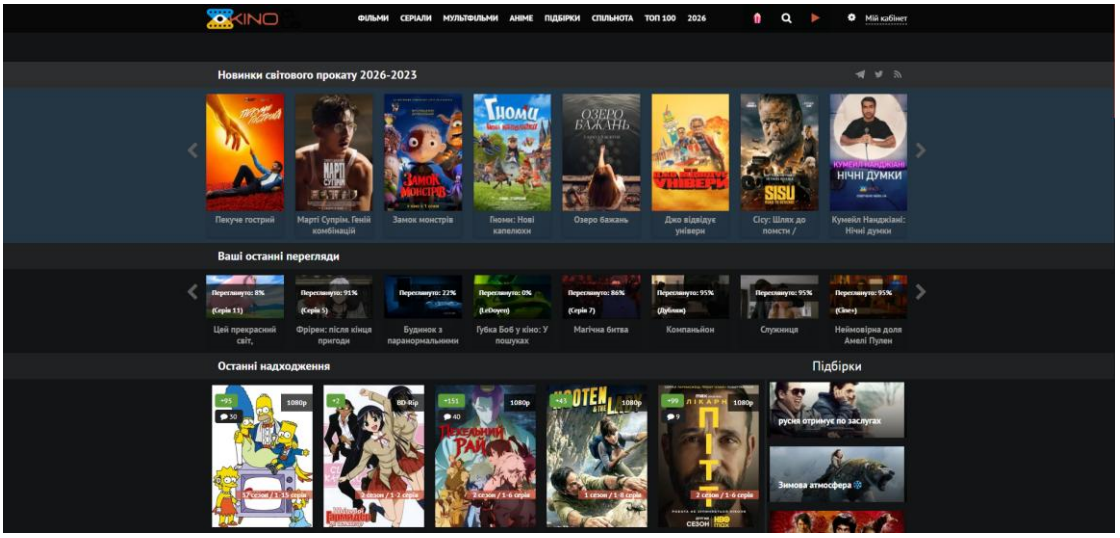


Рисунок 1.3 – Головна сторінка UAKINO

Навігація сайту реалізована у вигляді списків та фільтрів, що дозволяє швидко знайти потрібний контент за основними параметрами, такими як жанр, рік випуску або популярність. Така структура є зрозумілою та функціональною, однак переважно орієнтована на базовий пошук без глибокої персоналізації. Проте дизайн платформи є досить простим і не завжди відповідає сучасним вимогам до користувацького досвіду, зокрема щодо візуальної привабливості, адаптивності інтерфейсу та інтерактивності елементів.

Серед переваг платформи UAKINO можна виділити характеристики**:**

— безкоштовний доступ до контенту;

— наявність україномовного інтерфейсу;

— велика кількість фільмів різних років випуску.

Аналіз функціональних можливостей платформи виявляє низку недоліків**:**

— відсутність персоналізації;

— застарілий дизайн;

— надмірна кількість рекламних елементів.

Четвертим сервісом для аналізу є платформа MEGOGO, яка є одним із найпопулярніших легальних онлайн-кінотеатрів в Україні [10]. Сервіс поєднує можливість перегляду фільмів, серіалів, телевізійних каналів та спортивного контенту. MEGOGO працює за моделлю безкоштовного та платного доступу, що

робить його універсальним для різних категорій користувачів. Головну сторінку платформи продемонстровано на Рисунку 1.4, що дозволяє оцінити структуру інтерфейсу та особливості організації контенту.

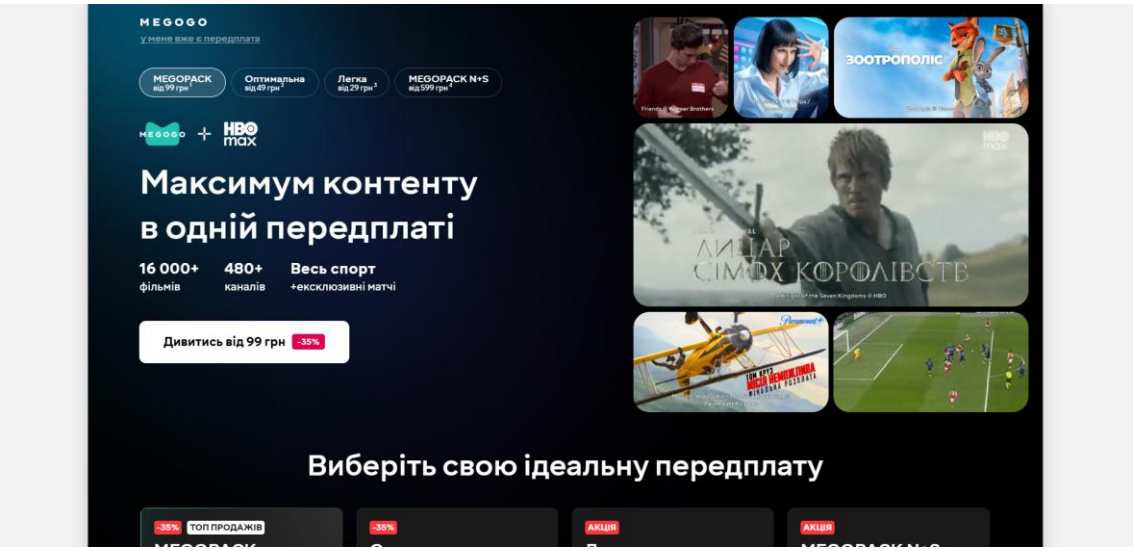


Рисунок 1.4 – Інтерфейс сторінки MEGOGO

Платформа має сучасний та адаптивний інтерфейс, підтримує створення персональних профілів користувачів і пропонує рекомендаційну систему, що формується на основі історії переглядів та індивідуальних вподобань. Завдяки цьому користувач отримує персоналізовані підбірки контенту та швидший доступ до актуальних новинок. Сервіс активно розвиває український дубляж, субтитрування та україномовний інтерфейс, що підвищує доступність платформи для національної аудиторії та відповідає сучасним тенденціям розвитку.

Перевагами MEGOGO є**:**

— легальний контент із дотриманням авторських прав;

— гнучка модель доступу;

— повноцінна підтримка української мови та локалізований контент;

— широкий функціонал.

Водночас сервіс має наступну недоліки**:**

— обмежений обсяг безкоштовного контенту;

— перевантаженість інтерфейсу для нових користувачів.

П’ятою платформою для аналізу є український сервіс SWEET.TV, який спеціалізується на онлайн-телебаченні та перегляді фільмів і серіалів за підпискою [11]. Даний вебзастосунок орієнтований на масового користувача та має просту структуру навігації. Інтерфейс сервісу є зрозумілим та адаптованим для різних пристроїв. На наступному Рисунку 1.5 зображено головну сторінку цього сайту, що дозволяє продемонструвати особливості його інтерфейсу та організацію контенту.

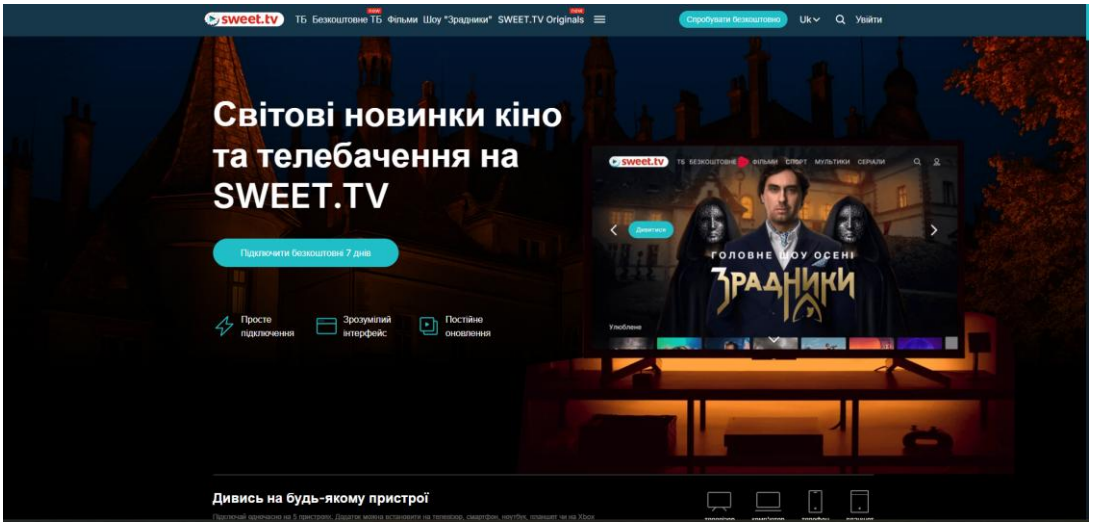


Рисунок 1.5 – Головна сторінка системи SWEET.TV

SWEET.TV пропонує обмежений, проте структурований каталог фільмів та серіалів, що опирається на зручності користування та стабільності роботи сервісу. Контент упорядковано за тематичними категоріями, жанрами та популярністю, що спрощує процес навігації та пошуку потрібного матеріалу. Платформа робить акцент на якості трансляції та технічній надійності, забезпечуючи стабільне відтворення відео без значних затримок або збоїв. Такий підхід дозволяє створити комфортне середовище для перегляду та виділити серед інших сервісів.

Серед основних переваг платформи SWEET.TV можна виокремити аспекти:

— простота використання, що забезпечується зрозумілим інтерфейсом;
— повноцінна українська локалізація, яка охоплює інтерфейс та дубляж;
— адаптивність до різних пристроїв та підтримка Smart TV;
— стабільна робота сервісу , що проявляється у якісному відтворенні відео.

До певних недоліків системи можна віднести наступне:

— обмежений каталог;

— слабка система рекомендацій, яка базується на загальних категоріях;

— відсутність розширених фільтрів пошуку за детальними параметрами;

— мінімальний рівень персоналізації.

Після аналізу п’ятьох існуючих вебсистем, що надають можливість онлайн-перегляду або вибору кінотворів, було встановлено, що кожна з розглянутих платформ має власний підхід до організації контенту, навігації та взаємодії з користувачем. Зарубіжні сервіси, такі як Netflix та Disney+, орієнтовані на закриту екосистему з платною підпискою, високим рівнем візуального оформлення та використанням алгоритмів персоналізованих рекомендацій. Водночас українські платформи, MEGOGO, SWEET.TV та UAKINO, здебільшого роблять акцент на локалізований контент, доступність сервісу та простоту використання, що відповідає потребам національної аудиторії.

Проведений аналіз показав, що більшість розглянутих вебсистем мають схожий базовий функціонал, а саме каталог кінотворів, жанрову класифікацію, пошук за назвою та фільтрацію контенту. Певна кількість обмежень та слабка персоналізація без урахування поточного емоційного стану користувача, а також відсутність можливостей для формування власних колекцій створює негативний досвід. Деякі випадки демонструють перевантаженість інтерфейсу або надмірну спрощеність, де більшість платформ орієнтуються на універсальні алгоритми підбору контенту, що не пропонують особливих підходів до інтерактивної взаємодії з різними користувачами.

Таким чином, результати аналізу наявних програмно-технічних рішень дозволяють сформувати основні вимоги до проєктованої вебсистеми підбору різних кінотворів. Поєднання інтуїтивного інтерфейсу, структурованої подачі інформації, персоналізованого підходу до вибору контенту надають можливості створення власних колекцій для перегляду. Врахування значних переваг та недоліків існуючих платформ стане основою для розроблення власної вебсистеми, яка буде орієнтована на підвищення якості користувацького досвіду та скорочення часу, необхідного для вибору фільму, серіалу, мультфільму або аніме.

1.3 Визначення функціональних та нефункціональних вимог до системи

Формування вимог до програмного забезпечення є одним із ключових етапів проєктування вебсистеми, оскільки саме на цьому етапі визначаються її основні можливості, обмеження та критерії якості. Для побудови інтерактивного проєкту підбору фільмів «MOVI.FLOW-ER» особливо важливим є врахування специфіки предметної області, а саме персоналізованого вибору мультимедійного контенту з урахуванням емоційного стану користувача. Чітко сформульовані вимоги дозволяють забезпечити узгодженість між функціональними можливостями системи, очікуваннями користувачів та технічними аспектами реалізації, а також значно спрощують подальші етапи розроблення, тестування й масштабування майбутнього вебзастосунку.

Проведений аналіз предметної області та наявних програмно-технічних рішень показав, що сучасні мультимедійні платформи повинні не лише надавати доступ до великого обсягу інформації про кінотвори, але й забезпечувати зручний, інтуїтивний та персоналізований процес вибору контенту. Саме тому вимоги до системи «MOVI.FLOW-ER» поділяються на функціональні та нефункціональні, що дає змогу комплексно охарактеризувати поведінку системи та її якісні властивості.

Функціональні вимоги визначають набір дій та можливостей, які повинна реалізовувати вебсистема для задоволення потреб користувачів [12]. У контексті даного проєкту вони охоплюють процеси взаємодії користувача з системою, логіку формування рекомендацій та роботу з мультимедійним контентом.

До основних функціональних вимог вебсистеми «MOVI.FLOW-ER» можна віднести наступні:

— реєстрація та авторизація користувачів у системі;
— збереження та обробка базових даних профілю користувача;
— вибір або визначення емоційного стану користувача;
— формування персоналізованих рекомендацій кінотворів на основі обраного стану, жанрових уподобань та історії взаємодії;
— перегляд каталогу кінотворів;

— пошук, сортування та фільтрація контенту за різними параметрами;

— перегляд детальної інформації про кінотвір;

— створення та керування власними списками перегляду;

— можливість оцінювання кінотворів;

— інтеграція з API TMDb для отримання актуальних даних;

— збереження історії переглядів і взаємодій користувача.

Реалізація зазначених вимог забезпечує повноцінний цикл взаємодії користувача з вебсистемою, а саме від первинного входу до отримання персоналізованих рекомендацій. Особливу роль у функціональності системи відіграє механізм підбору контенту на основі емоційного стану, що дозволяє відрізнити дану платформу від класичних рекомендаційних сервісів, орієнтованих виключно на жанри або рейтинги. Збереження історії взаємодії та оцінок дає змогу поступово підвищувати точність рекомендацій та адаптувати систему до індивідуальних вподобань користувача.

Нефункціональні вимоги описують якісні характеристики вебсистеми та визначають умови її стабільної, безпечної та зручної експлуатації [12]. Саме такі вимоги впливають безпосередньо на логіку роботи системи, а також є критично важливими для забезпечення високого рівня користувацького досвіду та надійності роботи сервісу.

До нефункціональних вимог вебсистеми «MOVI.FLOW-ER» належать:

— стабільність та безперебійність роботи системи;

— швидкодія та мінімальний час відгуку;

— масштабованість та можливість подальшого розширення функціоналу;

— зручний, інтуїтивно зрозумілий інтерфейс користувача;

— адаптивність інтерфейсу для різних пристроїв;

— захист персональних даних користувачів;

— цілісність та актуальність даних;

— повний захист особистої інформації;

— підтримка інтеграції з зовнішніми сервісами;

— простота супроводу та оновлення системи.

Вебсистема повинна працювати у режимі постійної доступності та забезпечувати коректне відновлення у випадку технічних збоїв. Важливим аспектом є швидке завантаження сторінок і оперативне формування рекомендацій, оскільки затримки негативно впливають на сприйняття сервісу користувачем. Інтерфейс системи має бути логічно структурованим, візуально зрозумілим та відповідати сучасним вимогам вебдизайну, що дозволить користувачам без труднощів взаємодіяти з функціоналом платформи.

Окрему увагу слід приділити безпеці та цілісності даних, захисту облікових записів та персональної інформації користувачів, оскільки кожній системі потрібний захист від несанкціонованого видалення, модифікації або фальсифікації даних [13]. Архітектура системи повинна передбачати можливість масштабування та впровадження нових функцій без суттєвих змін у вже реалізованих компонентах. Це дозволить у майбутньому розширювати вебсистему, інтегрувати додаткові алгоритми рекомендацій або підключати нові джерела даних.

Отже, визначені функціональні та нефункціональні вимоги формують основу для подальшого проєктування та реалізації інтерактивної вебсистеми підбору фільмів «MOVI.FLOW-ER». Дотримання даних вимог забезпечить створення надійного, адаптивного та орієнтованого на користувача програмного продукту, який повністю відповідає сучасним тенденціям розвитку персоналізованих мультимедійних сервісів.

1.4 Висновки. Постановка задачі

Процес створення першого розділу кваліфікаційної роботи дозволив провести комплексне дослідження предметної області, пов’язаної з онлайн-переглядом мультимедійного контенту та персоналізованим підбором кінотворів. Розглянуто сучасні тенденції розвитку цифрових платформ для перегляду фільмів, серіалів, мультфільмів та аніме. Було приділено увагу проблемі інформаційного перевантаження користувачів та скорочення часу на пошук кінотворів, що надають відповідний емоційний стан та особливі уподобання.

У підрозділі 1.1 здійснено змістовний аналіз предметної області, який описується на офіційних статистичних та аналітичних джерелах. Було встановлено, що онлайн-перегляд кінотворів є домінуючою формою споживання мультимедійного контенту, а кількість користувачів стримінгових сервісів постійно зростає. Дослідження показало, що традиційні підходи до рекомендацій, які базуються виключно на жанрах або рейтингах, не завжди задовольняють потреби користувачів. Це пояснює доцільність впровадження емоційно-орієнтованих підходів до підбору контенту, що дозволяють враховувати психологічний стан користувача та формувати більш релевантні рекомендації.

У підрозділі 1.2 виконано аналіз наявного програмно-технічного забезпечення предметної області на прикладі популярних міжнародних та українських вебплатформ, а саме Netflix, Disney+, MEGOGO, SWEET.TV та UA-Kino. Розгляд функціональних можливостей, інтерфейсних рішень та підходів до персоналізації контенту дозволив виявити як сильні сторони існуючих сервісів, так і їхні обмеження. Встановлено, що більшість платформ зосереджуються на рекомендаціях на основі історії переглядів і популярності контенту, приділяючи недостатню увагу емоційному аспекту вибору кінотворів. Отримані результати стали підґрунтям для формування власної концепції веб-системи, орієнтованої на більш глибоку персоналізацію.

У підрозділі 1.3 сформовано функціональні та нефункціональні вимоги до інтерактивної вебсистеми «MOVI.FLOW-ER». Визначено ключові функціональні можливості системи, серед яких персоналізований підбір кінотворів на основі емоційного стану користувача, інтеграція з відкритими базами даних мультимедійного контенту, керування списками перегляду та взаємодія з користувачем через інтуїтивний інтерфейс. Також були окреслені нефункціональні вимоги, що стосуються стабільності роботи, швидкодії, безпеки даних, масштабованості та зручності використання системи.

Проведений аналіз предметної області, існуючих рішень та вимог до системи дозволив сформувати технічну основу для подальшого проєктування та реалізації даної вебсистеми. На основі отриманих результатів можна сформулювати план

задачі кваліфікаційної роботи, яка полягає у створенні інтерактивної вебсистеми персоналізованого підбору кінотворів на основі емоційного стану користувача з використанням сучасних вебтехнологій та відкритих API.

Для досягнення даної мети у подальших розділах кваліфікаційної роботи необхідно описати наступні задачі:

— провести аналіз архітектурних підходів та стилів проєктування;
— визначити загальну архітектуру інтерактивної вебсистеми та взаємодію її компонентів;
— виконати декомпозицію системи на функціональні модулі та описати їх інтерфейси;
— спроєктувати структуру бази даних та взаємодії з зовнішніми API;
— обґрунтувати вибір технологій для реалізації серверної та клієнтської частин системи;
— реалізувати основний функціонал вебсистеми відповідно до сформованих вимог;
— виконати тестування програмного продукту та проаналізувати отримані результати;
— сформулювати загальні висновки ефективності реалізованої системи.

Отриманий результат у контексті першого розділу створює цілісне та структуроване уявлення про предметну область, дозволяє пояснити основні характеристики існуючих рішень, визначити їх сильні та слабкі сторони. Проведений аналіз окреслює функціональні та технологічні обмеження сучасних вебсервісів, а також формує обґрунтовані вимоги до майбутньої системи. Це визначає чіткі рамки подальшої роботи над кваліфікаційним проєктом, сприяє формуванню власних критеріїв проєктування та забезпечує логічний перехід до етапів моделювання, архітектурного планування та реалізації інтерактивної веб-системи «MOVI.FLOW-ER», орієнтованої на підвищення рівня персоналізації та якості користувацького досвіду.

2 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Аналіз та вибір архітектури вебзастосунку

У процесі проєктування інтерактивної вебсистеми персоналізованого підбору фільмів «MOVI.FLOW-ER» одним із ключових етапів є вибір оптимальної архітектури програмного забезпечення. Архітектура визначає загальні принципи побудови системи, організацію взаємодії її компонентів, а також безпосередньо впливає на продуктивність, масштабованість, надійність і зручність супроводу програмного продукту [14]. Правильно обрана архітектура дозволяє забезпечити ефективну обробку запитів користувачів, гнучкість розширення функціоналу та інтеграцію з зовнішніми сервісами.

Сучасні вебзастосунки можуть реалізовуватися з використанням різних архітектурних підходів, серед яких найбільш поширеними є монолітна архітектура, мікросервісна архітектура та клієнт-серверна архітектура. Кожен із цих підходів має свої переваги та недоліки, які необхідно враховувати при розробці програмного забезпечення. Для вибору типової архітектури для вебзастосунків обрано образ, що продемонстровано на Рисунку А.1.

Монолітна архітектура передбачає створення єдиного програмного продукту, у якому всі компоненти системи об’єднані в межах одного застосунку. Основною перевагою такого підходу є простота реалізації та розгортання, оскільки вся система функціонує як єдине ціле. Монолітні системи мають суттєві недоліки, зокрема складність масштабування, ускладнення внесення змін до окремих компонентів, а також зниження гнучкості при розширенні функціоналу [15].

Мікросервісна архітектура передбачає розподіл системи на набір незалежних сервісів, кожен з яких виконує окрему функцію. Такий підхід забезпечує високу гнучкість, можливість незалежного масштабування компонентів і зручність модифікації окремих частин системи. Однак впровадження мікросервісної архітектури пов’язане з підвищеною складністю розробки, необхідністю організації взаємодії між сервісами, а також потребою у використанні додаткових інструментів для моніторингу [16].

Згідно даного розроблюваного вебзастосунку, найбільш доцільним є використання клієнт-серверної архітектури. Даний підхід передбачає логічний поділ системи на дві основні складові, а саме клієнтську частину та серверну частину, які взаємодіють між собою через мережу. Така архітектура є стандартом для більшості сучасних вебзастосунків та дозволяє ефективно реалізувати інтерактивну взаємодію з користувачем [17].

Клієнтська частина вебзастосунку відповідає за відображення інтерфейсу користувача, обробку подій та взаємодію з користувачем. До основних функцій клієнтської частини належить візуалізація інформації про кінотворі, забезпечення навігації між розділами, обробка введених користувачем даних, а також формування запитів до серверної частини. Важливим аспектом є забезпечення інтуїтивно зрозумілого інтерфейсу, що сприяє швидкій роботі системи та покращенню загального користувацького досвіду.

Серверна частина виконує обробку запитів, реалізацію бізнес-логіки та взаємодію з базою даних. Вона відповідає за збереження інформації про користувачів, обробку їхніх запитів, формування персоналізованих рекомендацій, а також інтеграцію із зовнішніми сервісами. Саме на сервері реалізується основна логіка системи, включаючи алгоритми підбору фільмів відповідно до емоційного стану користувача.

Взаємодія між клієнтською та серверною частинами здійснюється за допомогою протоколу HTTP/HTTPS. Для передачі даних використовується формат JSON, який є легким, зручним для обробки та широко підтримується сучасними технологіями. Дане застосування дозволяє забезпечити швидкий обмін інформацією та ефективну взаємодію компонентів системи.

У межах клієнт-серверної архітектури доцільно використовувати REST-підхід до побудови програмного інтерфейсу. REST є архітектурним стилем, який базується на використанні стандартних HTTP-методів, таких як GET, POST, PUT та DELETE [18]. Використання REST API забезпечує стандартизовану взаємодію між клієнтом і сервером, спрощує розробку та дозволяє легко інтегрувати систему з іншими сервісами.

Особливістю вебсистеми «MOVI.FLOW-ER» є використання емоційно-орієнтованого підходу до формування рекомендацій. Це означає, що система враховує не лише історію переглядів користувача або популярність контенту, але й його поточний емоційний стан. Для реалізації такого підходу необхідно забезпечити обробку додаткових параметрів користувача, що потребує складної бізнес-логіки на серверній стороні.

Важливим аспектом проєктування є забезпечення безпеки системи. Використання протоколу HTTPS дозволяє захистити передані дані від перехоплення. Крім того, необхідно реалізувати механізми автентифікації та авторизації користувачів, що забезпечують контроль доступу до функціональних можливостей системи. Зокрема, можуть використовуватися токени доступу або сесійні механізми.

Ще одним важливим фактором є масштабованість системи. Обрана архітектура повинна забезпечувати можливість обробки зростаючої кількості користувачів без значного зниження продуктивності. Клієнт-серверна модель дозволяє масштабувати серверну частину шляхом додавання нових обчислювальних ресурсів або використання хмарних технологій.

Дана архітектура забезпечує зручність супроводу та оновлення системи. Оновлення серверної частини може здійснюватися без необхідності змін у клієнтській частині, що значно спрощує процес розгортання нових версій програмного продукту. Таке застосування є важливою перевагою при розробці сучасних вебзастосунків.

Отже, проведений аналіз архітектурних підходів дозволяє зробити висновок, що клієнт-серверна архітектура є досить відповідним рішенням для реалізації інтерактивної системи «MOVI.FLOW-ER». Вона забезпечує ефективну взаємодію між компонентами системи, гнучкість у розширенні функціоналу, можливість інтеграції із зовнішніми сервісами, а також відповідає сучасним вимогам до розробки вебзастосунків. Обрана архітектура створює надійну основу для подальшого проєктування та реалізації програмного продукту.

2.2 Опис структури даних та моделі бази даних

Проєктування структури даних є одним із ключових етапів розробки інтерактивної вебсистеми «MOVI.FLOW-ER», оскільки саме база даних забезпечує зберігання, обробку та швидкий доступ до інформації, необхідної для функціонування системи. В таких системах по запитах клієнтів файли передаються на персональні комп'ютери-клієнти та проходять обробку, а від правильності побудови моделі бази даних залежить ефективність роботи застосунку [19].

Основним призначенням бази даних у даному проєкті є збереження інформації про користувачів, кінотворі, оцінки, списки перегляду та результати персоналізованих рекомендацій. Власна база даних використовується для збереження даних користувачів та їхньої взаємодії з системою. У процесі проєктування системи було створено діаграму потоків даних, яка зображена на Рисунку 2.1 та демонструє загальну логіку роботи та взаємодії компонентів.

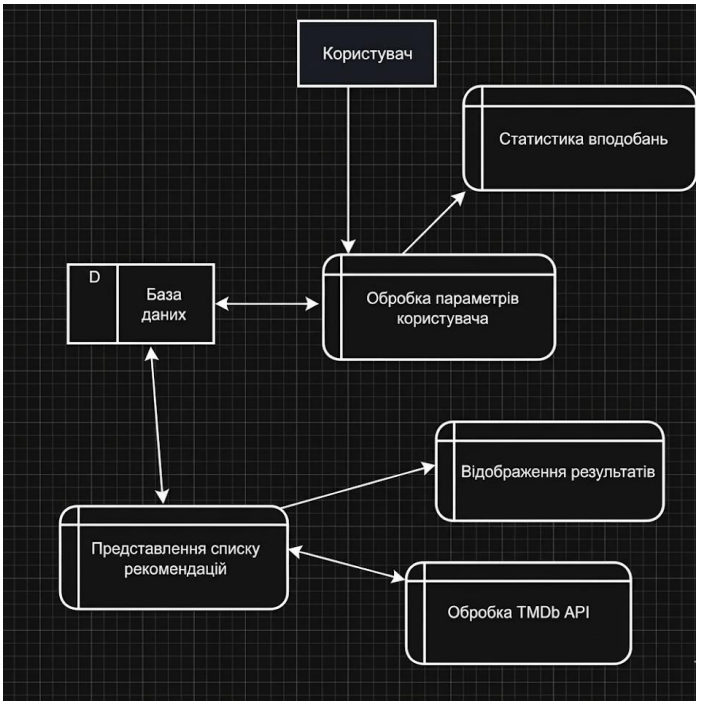


Рисунок 2.1 – DFD діаграма

При проєктуванні бази даних було застосовано принципи нормалізації, що дозволяють уникнути надлишковості даних, забезпечити їх цілісність та зменшити

ризик виникнення аномалій при оновленні. Для самої організації роботи необхідно забезпечити незалежність прикладних програм від даних, що надасть ефективне обслуговування користувачів системи [20].

У межах розроблюваної системи було визначено основні сутності, які відображають предметну область. До них належать:

— користувач (User);
— кінотвір (Movie);
— оцінка (Rating);
— список перегляду (Watchlist);
— емоційний стан (Mood);
— рекомендація (Recommendation).

Дані етапи надають можливість розподілити кожну сутність з її унікальними атрибутами та зобразити цю інформацію у вигляді таблиць для зберігання даних. Наступні таблиці сутностей продемонстровані з конкретними даними відповідно до їх атрибутів у наведених нижче таблицях (Таблиця 1-6).

Таблиця 2.1. Сутність «Кінотвори»

Id	tmdb_id	title	description	genre	Release_ye	rating	Poster_url
1	1	Персонаж	Головний герой	Фантастика	2020	8/10	https://photo.com/c/66
2	2	Гобіт	Пригода друзів	Фентезі	2005	9/10	https://photo.com/c/67
3	3	Тоторо	Дух	Аніме	2010	7/10	https://photo.com/c/65

Таблиця 2.2. Сутність «Користувач»

Id	Username	Email	Pass-word_Hash	Created_at	Role
1	Pavlo	us@gmail.com	A2G4H695	10.03.2026	User
2	Stas	s@gmail.com	C69G654G	20.05.2026	User
3	Rodrigo	m@gmail.com	8E82D736	01.11.2026	User

Таблиця 2.3. Сутність «Оцінки»

Id	User_id	Movie_id	Rating_value	Created_at
1	1	1	9	10.03.2026
2	2	2	8	20.05.2026
3	3	3	10	01.11.2026

Таблиця 2.4. Сутність «Список перегляду»

Id	User_id	Movie_id	Added_at
1	1	1	10.03.2026
2	2	2	20.05.2026
3	3	3	01.11.2026

Таблиця 2.5. Сутність «Емоційний стан»

Id	name	description
1	Радісний	user@gmail.com
2	Сумний	stas@gmail.com
3	Напружений	mexico26@gmail.com

Таблиця 2.6. Сутність «Рекомендації»

Id	User_id	Movie_id	Mood_id	Created_at
1	1	1	1	10.03.2026
2	2	2	2	20.05.2026
3	3	3	3	01.11.2026

Сформована структура бази даних інтерактивної вебсистеми «MOVI.FLOW-ER» охоплює всі ключові аспекти предметної області та забезпечує збереження як довідкової інформації про кінотворі, так і персональних даних користувачів. Запропонована модель включає сутності, що відповідають за користувачів, мультимедійний контент, оцінювання, формування списків перегляду, емоційні
стани та результати рекомендацій.

Аналогічний зв’язок реалізовано між таблицею кінотворів та таблицею оцінок, де один кінотвор може мати багато оцінок від різних користувачів. Таким чином, таблиця Ratings виступає як проміжна сутність, що реалізує зв’язок «багато-до-багатьох» між користувачами та кінотворями. Даний приклад упорядкування даних продемонстровано на Рисунку 2.2.

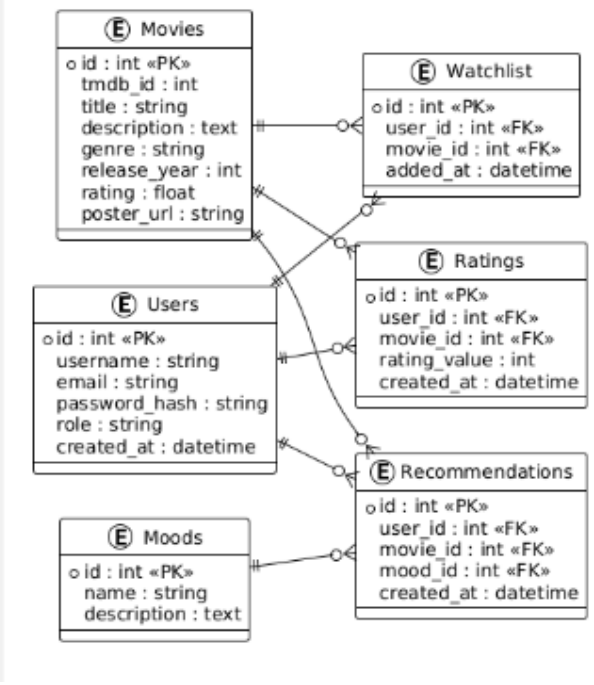


Рисунок 2.2 – ER-діаграма

Таблиця емоційних станів, у свою чергу, має зв’язок типу «один-до-багатьох» із таблицею рекомендацій, а саме кожен емоційний стан може використовуватись для формування багатьох рекомендацій. Усі зв’язки між таблицями реалізовані відповідно до логіки системи та забезпечують ефективну роботу. Запропонована структура дозволяє уникнути дублювання даних, забезпечує цілісність інформації та створює основу для механізмів персоналізації. Для кращого розуміння взаємодії програмних модулів системи було створено діаграму міжмодульних зв’язків, що зображена на Рисунку А.2.

2.3 Проєктування серверної частини вебзастосунку

Проєктування серверної частини вебзастосунку «MOVI.FLOW-ER» є одним із ключових етапів розробки системи, оскільки саме сервер забезпечує обробку запитів користувачів, взаємодію з базою даних та інтеграцію із сервісами. Серверна частина виступає центральним компонентом системи, який координує роботу всіх функціональних елементів і забезпечує коректне функціонування вебзастосунку. Основною задачею серверної частини є приймання даних від клієнта, їх обробка, формування відповідей, які повертаються користувачу у зручному форматі.

Серверна частина даного проєкту реалізується відповідно до клієнт-серверної архітектури з використанням REST-підходу до організації взаємодії. Це означає, що всі функціональні можливості системи представлені у вигляді окремих HTTP-запитів, які виконують певні операції над ресурсами. Такий підхід дозволяє забезпечити стандартизовану, зрозумілу взаємодію між клієнтом і сервером, що цілком задовільняє поставлені вимоги до поточного етапу проєктування.

Для реалізації серверної частини передбачається використання сучасного веб-фреймворку, який підтримує архітектурний підхід MVC. У цьому підході логіка застосунку розділяється на три основні компоненти, а саме модель для роботи з даними, контролер для обробки запитів користувача, представлення для формування відповіді [21]. Для кращого розуміння процесів, їх змін та всієї послідовності на Рисунку 2.3 зображено діаграму архітектури процесів системи.

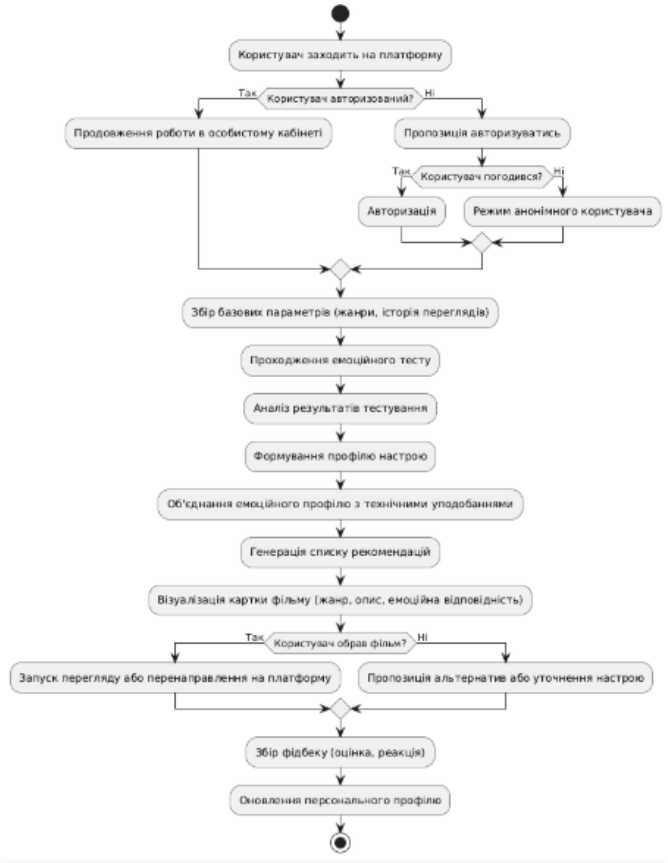


Рисунок 2.3 – Flowchart діаграма

Обробка отриманих даних на сервері включає декілька послідовних етапів. На першому етапі здійснюється перевірка коректності вхідних даних, яка дозволяє запобігти помилкам та некоректному введенню інформації. На другому етапі виконується обробка даних відповідно до бізнес-логіки системи. При формуванні рекомендацій система аналізує обраний користувачем емоційний стан, історію переглядів та оцінок, після чого здійснюється запит до бази даних.

Реалізація механізмів автентифікації та авторизації користувачів, а саме забезпечення безпеки доступу до системи використовуються сучасні підходи, такі як токенна автентифікація. Дане використання стало ефективним рішенням, яке вимагає здатності працювати з різноманітними клієнтами [22]. Після успішного входу користувач отримує спеціальний токен, який використовується для запитів до сервера. Саме це дозволяє ідентифікувати користувача та контролювати доступ до функціональних можливостей системи.

Важливим аспектом є забезпечення обробки помилок та виняткових ситуацій. Сервер повинен відповідає за обробку великого обсягу запитів,

збереження конфіденційних даних та підтримку інтеграції з іншими системами [23]. У таких випадках формується відповідь із відповідним кодом помилки та описом проблеми, що дозволяє клієнтській частині правильно обробити ситуацію.

Серверна частина вебзастосунку «MOVI.FLOW-ER» забезпечує повний цикл обробки даних, а саме від отримання запиту користувача до формування відповіді з урахуванням бізнес-логіки системи. Застосування сучасних підходів до організації серверної архітектури, інтеграції з API та забезпечення безпеки дозволяє створити масштабовану основу для функціонування вебзастосунку.

2.4 Проєктування клієнтської частини вебзастосунку

Клієнтська частина вебзастосунку «MOVI.FLOW-ER» є ключовим елементом взаємодії користувача із системою, оскільки саме вона забезпечує відображення інформації, приймання дій користувача та передачу даних на сервер для подальшої обробки. Основним завданням клієнтської частини є створення зручного, інтуїтивно зрозумілого та адаптивного інтерфейсу, який дозволяє користувачеві швидко знаходити необхідний контент, взаємодіяти з системою та отримувати персоналізовані рекомендації.

Проєктування клієнтської частини базується на сучасних підходах побудови вебінтерфейсів, зокрема використанні компонентної структури, що дозволяє розділити інтерфейс на логічні частини. Даний етап тісно взаємодіє із серверною через REST API, надаючи кінцевим користувачам доступ до всіх функцій платформи [24]. Інтерфейс застосунку складається з окремих компонентів, серед яких можна виділити навігаційну панель, блок рекомендацій, сторінку перегляду кінотвору, профіль користувача та систему керування списками перегляду.

Структура системи управління змістом у межах клієнтської частини організована таким чином, щоб забезпечити логічне групування інформації. Основні розділи вебзастосунку включають головну сторінку, сторінку рекомендацій, сторінку деталей кінотвору, сторінку профілю користувача та

сторінку списку перегляду. Головна сторінка виконує роль центрального елемента системи, де користувач може обрати свій емоційний стан та отримати персоналізовану добірку кінотворів. Сторінка рекомендацій відображає результати роботи алгоритмів підбору контенту.

Важливою складовою клієнтської частини є логічна організація інформації на сторінках. Інтерфейс побудований за принципом ієрархічної структури, де основні елементи розташовані у верхній частині сторінки, а додаткова інформація знаходиться у нижніх блоках. Сторінка кінотвору демонструє основну увагу на назві, постеру, рейтингу та короткому опису, тоді як додаткові елементи. Такий підхід дозволяє користувачеві швидко отримати основну інформацію та за потреби ознайомитися з деталями.

Клієнтська частина також забезпечує обробку дій користувача, таких як реєстрація, авторизація, оцінювання кінотворів, додавання до списку перегляду та вибір емоційного стану. Для цього використовуються спеціальні форми та елементи управління, які передають дані на сервер для подальшої обробки. Важливим аспектом є забезпечення валідації даних на стороні клієнта, що дозволяє зменшити кількість помилок та підвищити якість взаємодії з системою.

Дизайн інтерфейсу розроблюваного вебзастосунку орієнтований на сучасні тенденції у сфері вебдизайну, зокрема використання темної теми, контрастних акцентів та візуально привабливих карток кінотворів. Важливо розробити дизайн інтерфейсу, що передає складну інформацію в більш зручному, простому, інтуїтивному та стилізованому вигляді для користувача кінцевого продукту [25]. Адаптивність інтерфейсу гарантує коректне відображення застосунку на різних пристроях, тому користувачі отримують однакову якість взаємодії незалежно від формату екрану.

Схема роботи інтерфейсу передбачає послідовність взаємодії користувача з основними функціями системи. Спочатку користувач потрапляє на головну сторінку, де може авторизуватись або продовжити роботу як гість. Далі він обирає емоційний стан або використовує пошук для знаходження кінотвору. Після цього система формує відповідні рекомендації та відображає їх у вигляді карток. Клієнт

може переглянути детальну інформацію про кінотвор, оцінити його або додати до списку перегляду.

Отже, клієнтська частина вебзастосунку «MOVI.FLOW-ER» реалізує повноцінну систему взаємодії користувача з платформою, забезпечуючи зручність, швидкодію та високий рівень персоналізації. Продумана структура інтерфейсу, логічна організація інформації дозволяють створити ефективний інструмент.

2.5 Створення макета вебзастосунку та дизайн

Проєктування макета вебзастосунку «MOVI.FLOW-ER» є важливим етапом розробки, оскільки саме на цьому рівні формується структура представлення інформації та визначається зовнішній вигляд системи. Макет стосується того, як інформація структурована та класифікована, тому якісне опрацювання макета забезпечує зручність взаємодії користувача з вебзастосунком, швидкість орієнтації в інтерфейсі та загальне сприйняття продукту [26]. Основною метою даного етапу є створення логічно структурованого, естетично привабливого та функціонально зручного інтерфейсу, який відповідає сучасним вимогам до вебдизайну.

Процес створення структури вебзастосунку було умовно поділено на два основні етапи, а саме структуризацію інформації та її візуальне представлення. На першому етапі здійснюється аналіз усіх даних, які повинні бути доступні користувачу, а також їх логічне групування. У межах даного проєкту інформація класифікується за основними категоріями, які виконують окрему функцію у вигляді відповідних розділів вебзастосунку.

Структуризація інформації передбачає створення чіткої ієрархії матеріалів, яка дозволяє користувачеві легко орієнтуватися у системі. На верхньому рівні знаходяться основні розділи, доступ до яких здійснюється через головне навігаційне меню. До них належать головна сторінка, сторінка рекомендацій, профіль користувача та список перегляду. Сторінка рекомендацій включає картки кінотворів, які можна переглянути більш детально, переходячи на окрему сторінку

конкретного фільму або серіалу.

Особливу увагу приділено логічному групуванню контенту. Кінотворі об’єднуються за жанрами, популярністю, рейтингом або емоційним станом користувача. Такий підхід дозволяє створювати зрозумілі для користувача дії, що спрощує процес пошуку та вибору контенту. Крім того, використання тематичних підбірок підвищує рівень персоналізації та робить взаємодію з системою більш природною. Макет головної сторінки можна переглянути на Рисунку 2.4.

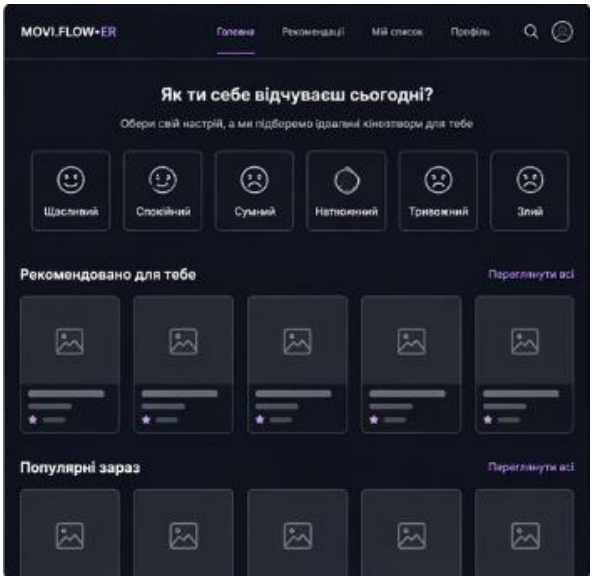


Рисунок 2.4 – Макет головної сторінки

Головна сторінка містить блок вибору емоційного стану користувача, що є основою функціональною особливістю системи. Після вибору стану з’являються персоналізовані рекомендації у вигляді карток. Сторінка кінотвору містить детальну інформацію, включаючи опис, рейтинг, жанр та можливість додавання до списку перегляду. Сторінка профілю дозволяє керувати персональними даними та переглядати історію взаємодії з системою.

Другим етапом є візуальне представлення структури вебзастосунку, яке передбачає розробку макетів сторінок та визначення стилістичних рішень. Основним принципом дизайну є мінімалізм та зручність сприйняття інформації. Мінімалістичний дизайн дозволяє фокусувати увагу на головному, зменшуючи навантаження [27]. Інтерфейс будується таким чином, щоб уникати надмірності

елементів та забезпечити фокус користувача на основному контенті.

Колірне рішення вебзастосунку орієнтоване на використання темної теми, яка є актуальною для мультимедійних платформ. Використання яскравих та насичених кольорів може викликати помилкове сприйняття глибини різних об'єктів [28]. Основний фон виконано у темних відтінках, що дозволяє зменшити навантаження на очі користувача та підкреслити візуальний контент. Для акцентів використовуються яскраві кольори, які виділяють активні кнопки, важливі елементи інтерфейсу та інтерактивні компоненти. Саме на макеті рекомендаційної сторінки можна переглянути вигляд пропозицій, Рисунок 2.5.

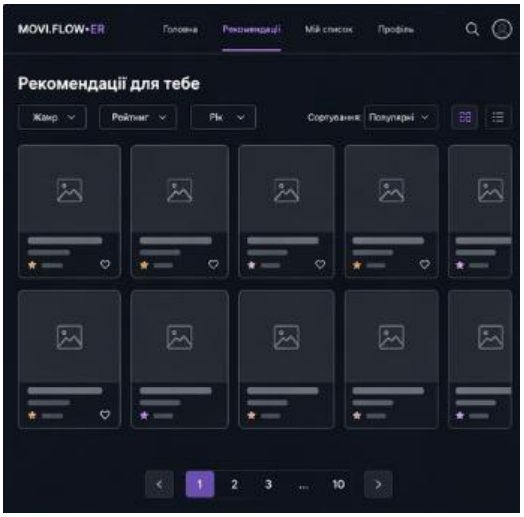


Рисунок 2.5 – Макет рекомендацій

Графічне оформлення включає використання якісних постерів кінотворів, іконок та візуальних елементів, що підкреслюють тематику застосунку. Використання однакових шрифтів, розмірів тексту та відступів дозволяє забезпечити гармонійність дизайну та виступає важливою складовою для викладу текстової інформації [29].

Макет вебзастосунку передбачає адаптивність інтерфейсу, що забезпечує коректне відображення на різних пристроях. На мобільних пристроях кількість елементів у рядку зменшується, а навігаційне меню може трансформуватися у компактний формат. Дану адаптацію можна переглянути на макетах авторизації та реєстрації, Рисунок 2.6.

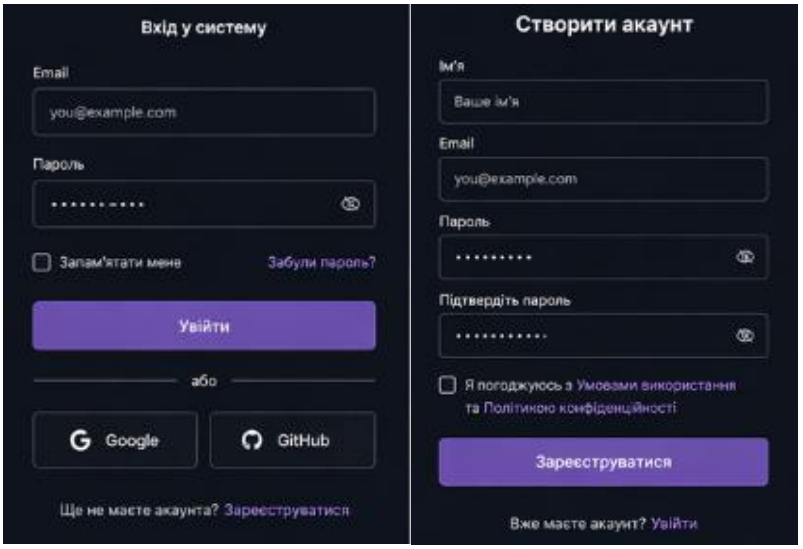


Рисунок 2.6 – Макет авторизації та реєстрації

Таким чином, створення макета вебзастосунку «MOVI.FLOW-ER» включає комплексний підхід до організації інформації та розробки дизайну інтерфейсу. Використання принципів структуризації, сучасних дизайнерських рішень та адаптивності дозволяє створити зручний, функціональний та естетично привабливий вебзастосунок, що відповідає потребам користувачів та сучасним стандартам веброзробки.

2.6 Аналіз та вибір технологій і методів реалізації вебзастосунку

У процесі розробки інтерактивної вебсистеми «MOVI.FLOW-ER» було здійснено комплексний аналіз сучасних вебтехнологій з метою вибору оптимального технологічного стеку, який забезпечить стабільність, інтерактивність, безпечність та можливість подальшого масштабування системи. Основна увага зосереджується на ключових елементах вибору стека технологій, а саме функціональних вимогах, масштабованості та безпеці [30]. Оскільки розроблювана система є клієнт-серверним вебзастосунком із динамічною обробкою даних та інтеграцією із зовнішніми API, вибір технологій здійснювався з урахуванням ефективної взаємодії між клієнтською та серверною частинами.

Основою клієнтської частини вебзастосунку є мова гіпертекстової розмітки

HTML5, яка використовується для формування логічної структури вебсторінок. Характеристики кожного елемента розмітки HTML визначаються у браузері, але можуть бути змінені або покращені додатковим використанням CSS дизайнера сторінок [31]. Для стилізації інтерфейсу застосовуються каскадні таблиці стилів CSS3, що дозволяють плавно перейти від одного стану елемента до іншого, а саме від початкового до кінцевого стану [32]. Використання гнучких механізмів позиціонування елементів підвищує зручність користування системою.

Дані стилі дозволяють реалізувати єдину дизайн-систему, що включає кольорову палітру, типографіку, систему відступів та розміщення елементів. Особлива увага приділяється адаптивності інтерфейсу, яка досягається за допомогою медіазапитів та гнучких механізмів побудови макета. Це забезпечує коректне відображення застосунку на різних пристроях, а саме від персональних комп'ютерів до смартфонів і планшетів. Такий підхід є необхідним, враховуючи сучасні тенденції мобільного споживання контенту.

Інтерактивна складова клієнтської частини реалізується за допомогою мови програмування JavaScript. Використання JavaScript дозволяє здійснювати динамічне керування елементами сторінки через об'єктну модель документа, змінювати структуру, реагувати на дії користувача та реалізовувати складні сценарії взаємодії. Зазвичай певні об'єкти мають бути змінними, щоб значення їх властивостей могло бути змінено за допомогою відповідних методів [33]. Обробка подій здійснюється шляхом використання механізму подій, що забезпечує гнучке керування поведінкою інтерфейсу.

Основою серверної частини вебзастосунку обрано мову програмування C# та платформу ASP.NET Core. Дана технологія є сучасним кросплатформним фреймворком для створення вебзастосунків та вебсервісів, який пропонує підвищену продуктивність та модульність, що є особливо важливим для розробки сучасних продуктів [34]. Використання C# дозволяє реалізувати об'єктно-орієнтований підхід до проєктування програмного забезпечення, що надає безпечні принципи програмування, такі як перевірка типів та меж масиву, виявлення використання неініціалізованих змінних, автоматичне прибирання сміття [35].

Платформа ASP.NET Core підтримує вбудований механізм маршрутизації запитів, систему контролерів та представлень, що дозволяє реалізувати архітектурний шаблон MVC. Важливою перевагою є підтримка механізму Dependency Injection, що застосування цієї концепції полягає у відділенні бізнес-логіки від її візуалізації. Завдяки цьому бізнес-логіка, рівень доступу до даних та представлення можуть бути ізольовані один від одного, що спрощує тестування та масштабування застосунку. Дане застосування концепції є корисним, коли користувач повинен бачити дані одночасно в різних контекстах, а робота з великою кількістю одночасних користувачів проходить без блокування потоків [36].

Для реалізації взаємодії з базою даних було обрано технологію Entity Framework Core, яка є кросплатформною версією популярної технології доступу до даних, яка дозволяє працювати з базою даних за допомогою об'єктів .NET та усувати потребу у більшій частині коду для доступу до даних [37]. Використання EF Core дозволяє працювати з базою даних через моделі C#, що мінімізує необхідність написання низькорівневих SQL-запитів. Такий підхід підвищує безпечність роботи з даними та забезпечує узгодженість між програмною логікою та структурою сховища. Для керування змінами структури бази даних застосовується механізм міграцій, що дозволяє відслідковувати еволюцію схеми БД протягом усього життєвого циклу проєкту.

У якості системи керування базами даних може використовуватися Microsoft SQL Server, сумісна з EF Core. Реляційна модель даних забезпечує оптимальне поєднання продуктивності, адаптивності та гнучкості. Це рішення дозволяє створити високоефективний програмний продукт, тому підхід є доцільним з огляду на необхідність зберігання користувацьких профілів, оцінок, списків перегляду, історії взаємодії та результатів рекомендацій [38].

Інтеграція із зовнішніми сервісами реалізується через стандартний HTTP-клієнт, що входить до складу .NET. Отримані від API дані у форматі JSON десеріалізуються у відповідні моделі C#, після чого можуть бути оброблені, збережені або використані для формування рекомендацій. Саме даниї підхід забезпечує гнучкість взаємодії із зовнішніми джерелами інформації та дозволяє

підтримувати актуальність контенту без необхідності його повного зберігання у локальній базі даних.

Особлива увага приділяється забезпеченню безпеки вебзастосунку. Платформа ASP.NET Core надає вбудовані механізми автентифікації та авторизації користувачів, захисту від атак та безпечного зберігання паролів із використанням алгоритмів хешування. Передача даних здійснюється через протокол HTTPS, що призначений для забезпечення надійного захисту тільки гіпертекстових документів вебсервера. Даний протокол гарантує авторизацію та захист документів [39].

Вибір зазначеного технологічного стеку зумовлений необхідністю забезпечити високу продуктивність, структурованість коду, гнучкість розширення функціоналу та відповідність сучасним стандартам веброзробки. Платформа .NET дозволяє реалізувати масштабовану систему, яка може бути доповнена новими модулями без суттєвого порушення існуючої архітектури. Крім того, використання C# та ASP.NET Core забезпечує можливість подальшого розвитку проєкту, оптимізації алгоритмів рекомендацій та інтеграції додаткових сервісів.

Таким чином, проведений аналіз технологій підтверджує доцільність використання мови програмування C#, фреймворку ASP.NET Core, Entity Framework Core та сучасних вебтехнологій HTML, CSS і JavaScript для реалізації інтерактивної вебсистеми «MOVI.FLOW-ER». Обрані інструменти забезпечують необхідний рівень функціональності, безпеки, продуктивності та підтримуваності програмного забезпечення, що відповідає вимогам кваліфікаційної роботи.

2.7 Висновки. Проєктування програмного забезпечення

Проведена робота охоплювала визначення архітектурного підходу, моделювання структури бази даних, проєктування серверної та клієнтської частин, розроблення концепції інтерфейсу користувача, а також обґрунтування вибору технологій реалізації. Отримані результати сформували цілісну технічну основу для подальшого етапу програмної реалізації системи.

У підрозділі 2.1 здійснено аналіз можливих архітектурних рішень та обґрунтовано вибір клієнт-серверної архітектури з використанням сучасного фреймворку ASP.NET Core. Даний підхід дозволяє забезпечити розмежування відповідальностей між компонентами системи, підвищити її масштабованість та стабільність роботи, а також створити гнучку основу для подальшого розвитку.

У підрозділі 2.2 виконувалося проєктування структури бази даних та визначено ключові сутності системи, а саме користувачів, кінотворів, оцінок, рекомендацій, списків перегляду та емоційних категорій. Модель забезпечує логічну цілісність даних, підтримує реалізацію персоналізованих сценаріїв взаємодії та створює основу для ефективного зберігання й опрацювання інформації. Визначені зв'язки між таблицями дозволяють реалізувати механізм формування рекомендацій.

У підрозділах 2.3 та 2.4 виконувалося проєктування серверної та клієнтської частин вебзастосунку. Серверна частина орієнтована на обробку запитів, реалізацію бізнес-логіки, взаємодію з базою даних та зовнішніми сервісами, а також забезпечення безпеки та авторизації користувачів. Клієнтська частина спроєктована з урахуванням принципів зручності та логічної структури інтерфейсу. Було визначено логічну організацію інформації на сторінках ресурсу, схему навігації та основні сценарії взаємодії користувача з системою.

У підрозділі 2.5 розроблено концепцію макета вебзастосунку та визначено загальні дизайнерські рішення. Було здійснено структуризацію інформації, визначено ієрархію матеріалів та організацію контенту на сторінках. Запропоновані візуальні та стилістичні рішення спрямовані на створення сучасного сервісу. У підрозділі 2.6 проведено аналіз та обґрунтовано вибір технологій реалізації, а саме мови програмування, фреймворку, а також сучасних вебтехнологій. Обраний технологічний стек забезпечує продуктивність, безпечність та масштабованість.

Отже, результати другого розділу дозволили сформувати детальну модель майбутньої вебсистеми, визначити її структурну, функціональну та технологічну основу. Проведене проєктування забезпечує логічний перехід до наступного етапу, а саме програмної реалізації та практичного впровадження системи.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ СИСТЕМИ

3.1 Розроблення бази даних

Розроблення бази даних інтерактивної вебсистеми «MOVI.FLOW-ER» є одним із ключових етапів програмної реалізації, оскільки саме база даних забезпечує збереження, оброблення та структуроване представлення інформації, необхідної для функціонування системи персоналізованого підбору кінотворів. На цьому етапі було здійснено практичну реалізацію логічної моделі даних, розробленої в попередньому розділі, з урахуванням вимог до стабільності, масштабованості та забезпечення цілісності даних.

Фізична реалізація бази даних виконана на основі реляційної моделі з використанням системи керування базами даних Microsoft SQL Server. Такий вибір обумовлений її надійністю, підтримкою транзакційності, високим рівнем безпеки та інтеграцією з технологічним стеком ASP.NET Core. База даних побудована відповідно до принципів нормалізації, що дозволяє уникнути надмірного дублювання інформації та забезпечити логічну узгодженість між сутностями.

У межах системи реалізовано таблиці, які відповідають основним сутностям предметної області, а саме користувачі, кінотворі, оцінки, списки перегляду, емоційні стани та рекомендації. Таблиця користувачів зберігає реєстраційні дані, службову інформацію та забезпечує ідентифікацію суб’єктів взаємодії із системою. Таблиця кінотворів містить структуровані дані про фільми, серіали та інші мультимедійні матеріали, отримані через інтеграцію з зовнішнім API. Таблиці оцінок та списків перегляду реалізують механізми персоналізації, дозволяючи зберігати історію взаємодії користувача з контентом.

Зв’язки між таблицями реалізовані через механізм первинних та зовнішніх ключів, це дозволяє забезпечити цілісність даних, підтримувати коректність відношень типу «один-до-багатьох». Користувач може мати значну частину оцінок, елементів у списку перегляду та багато записів рекомендацій. Така структура дозволяє ефективно реалізувати персоналізовану логіку системи без надмірного ускладнення моделі.

Під час реалізації бази даних особливу увагу приділено питанням забезпечення цілісності та безпеки інформації. Для кожної таблиці визначено первинні ключі, обмеження обов'язковості полів, типи даних відповідно до їх призначення, а також зовнішні ключі для підтримки міжтабличних зв'язків. Паролі користувачів не зберігаються у відкритому вигляді, а проходять процедуру хешування, що підвищує рівень захисту персональних даних.

Оновлення структури бази даних здійснюється за допомогою механізму міграцій, що дозволяє керувати версіями схеми та поетапно вносити зміни без втрати вже збережених даних. Такий підхід є особливо важливим у процесі розроблення, коли структура може розширюватися відповідно до нових функціональних вимог. Детальний зв'язок між таблицями можна побачити на діаграмі класів, Рисунку 3.1.

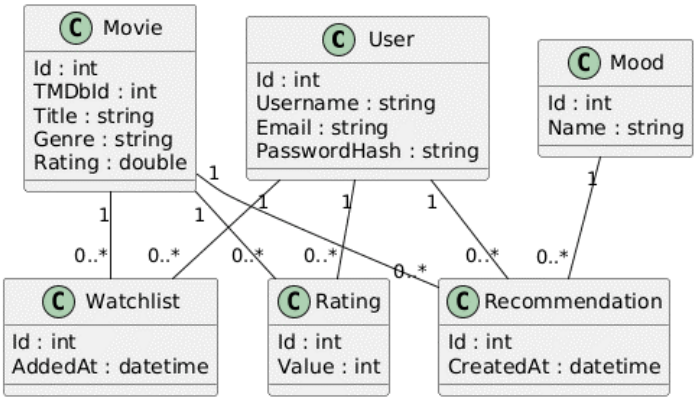


Рисунок 3.1 – Діаграма класів

Інтеграція бази даних із серверною частиною вебзастосунку реалізована через контекст доступу до даних, який забезпечує взаємодію між логікою застосунку та фізичною структурою бази даних. Усі операції збереження, отримання, оновлення та видалення даних виконуються через відповідні програмні механізми доступу до даних, що гарантує централізоване керування системою.

Представлена база даних відповідає вимогам функціональності, масштабованості та надійності. Вона забезпечує стабільну роботу системи, підтримує емоційно-орієнтовану модель рекомендацій та створює фундамент для подальшого розширення можливостей вебзастосунку «MOVI.FLOW-ER».

3.2 Розроблення програмних модулів

Розроблення програмних модулів вебзастосунку «MOVI.FLOW-ER» здійснювалося відповідно до визначеної архітектури та сформованих функціональних вимог. Основною метою цього етапу було створення логічно завершених компонентів системи, кожен із яких відповідає за окремий напрям функціональності та взаємодіє з іншими модулями через визначені інтерфейси. Діаграма послідовності може відображати паралельне виконання дій. Елементи діаграми допомагають візуалізувати послідовність дій та рішень. Саме це робить модель більш зрозумілою для аналізу та оптимізації процесу [40]. На Рисунку 3.2 продемонстровано діаграму, яка зображає відповідний потік дій системи.

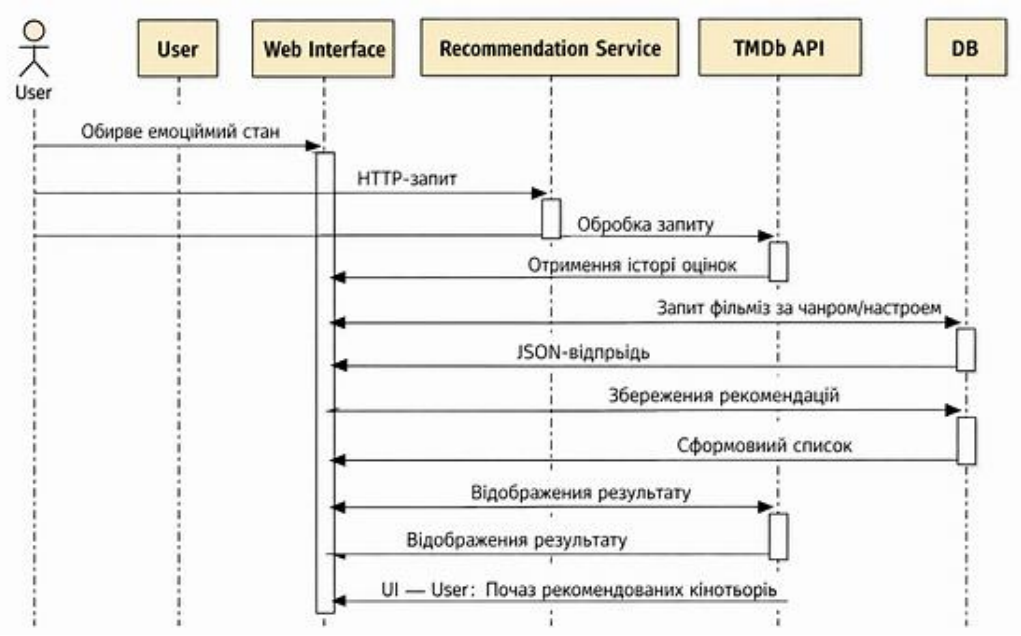


Рисунок 3.2 – Діаграма послідовності

Одним із базових модулів є модуль реєстрації користувачів. Його реалізація передбачає оброблення введених користувачем даних, перевірку їх коректності, валідацію електронної адреси, хешування пароля та збереження інформації до бази даних. Під час створення облікового запису система перевіряє унікальність пошти, що дозволяє уникнути дублювання записів. Усі персональні дані проходять серверну перевірку, що підвищує рівень безпеки та захисту від некоректного

введення або навмисних спроб порушення роботи системи.

Модуль авторизації забезпечує ідентифікацію користувача під час входу до системи. Після введення логіна та пароля здійснюється порівняння хешованого значення пароля з тим, що збережений у базі даних. Реалізований механізм дозволяє обмежувати доступ до персоналізованих функцій системи, таких як формування списків перегляду, оцінювання кінотворів та отримання рекомендацій відповідно до емоційного стану.

Центральним елементом застосунку є модуль персоналізованих рекомендацій. Його реалізація базується на поєднанні декількох джерел даних, а саме історії оцінок користувача, обраного емоційного стану та метаданих кінотворів, отриманих через інтеграцію з TMDb API. Таке застосування дозволяє отримувати реальні рецензії та автоматично обробляти їх у системі, що надає чітке бачення інтерфейсу для кінцевого користувача платформи [41]. Після вибору користувачем певного настрою система формує запит до бази даних та зовнішнього API, аналізує відповідні жанрові характеристики та генерує перелік релевантних кінотворів. Таким чином забезпечується адаптивність рекомендацій та створюється індивідуальний користувацький досвід.

Модуль керування списком перегляду реалізує можливість додавання та видалення кінотворів із персональної колекції. Під час додавання запису система перевіряє, чи не існує вже відповідний зв’язок між користувачем і обраним фільмом. Це дозволяє уникнути дублювання даних і підтримувати впорядкованість інформації. Аналогічним чином функціонує модуль оцінювання, який забезпечує збереження рейтингу, виставленого користувачем, та його подальше використання для формування рекомендацій.

Окремо реалізовано модуль інтеграції з зовнішнім API, що формує HTTP-запити, обробляє відповіді у форматі JSON, отримує дані та забезпечує їх тимчасове або постійне збереження у базі даних. Застосування JSON формату являє обмін даними між сервером та браузером, а також надає зберігання даних та створення файлів конфігурації [42]. Даний модуль дозволяє гарантувати актуальність інформації без необхідності ручного наповнення даних.

Усі програмні модулі взаємодіють між собою через серверну логіку, яка реалізована на платформі ASP.NET Core. Контролери обробляють HTTP-запити, передають їх до відповідних сервісів, які виконують бізнес-логіку, а після цього повертають результат у вигляді вебсторінки. Така структура дозволяє чітко розмежувати відповідальність між рівнями системи та забезпечити зручність підтримки програмного коду. На Рисунку 3.3 зображено діаграму компонентів програмних модулів.

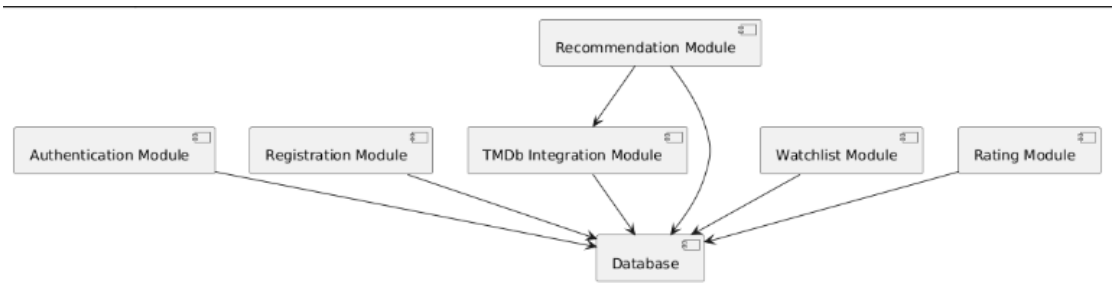


Рисунок 3.3 – Діаграма компонентів програмних модулів

Наведена діаграма компонентів відображає логічну структуру програмних модулів вебзастосунку «MOVI.FLOW-ER» та демонструє взаємозв'язки між ними. Результат створення модулів зображено у змісті файлів проєкту, а саме на Рисунку А.3. Центральне місце займає модуль рекомендацій, який інтегрується як із внутрішньою базою даних, так і з модулем взаємодії з TMDb API, що забезпечує отримання актуальної інформації про кінотворів. Модулі авторизації та реєстрації відповідають за автентифікацію користувачів та контроль доступу, а модулі оцінювання та списку перегляду забезпечують персоналізовану взаємодію з контентом системи.

Загалом реалізація програмних модулів забезпечує повний цикл взаємодії користувача із системою, а саме від створення облікового запису до отримання персоналізованих рекомендацій. Модульний підхід дозволяє гнучко розширювати функціональність вебзастосунку та інтегрувати додаткові сервіси. Детальний опис покрокового користування системою для різних категорія користувачів наведено у Додатку В, яке представляє керівництво користувача.

3.3 Технічні характеристики вебзастосунку

Технічні характеристики розробленої інтерактивної вебсистеми «MOVI.FLOW-ER» охоплюють особливості серверної та клієнтської частин, вимоги до програмного й апаратного середовища, а також умови, необхідні для стабільного та безперебійного функціонування застосунку. Визначення технічних параметрів є важливим етапом завершення розробки, оскільки дозволяє оцінити готовність системи до реального використання та подальшого масштабування.

Розроблений вебзастосунок функціонує на основі клієнт-серверної архітектури, де серверна частина відповідає за обробку запитів, виконання бізнес-логіки, формування рекомендацій та взаємодію з базою даних і зовнішніми сервісами, а клієнтська частина забезпечує відображення інформації та інтерактивну взаємодію з користувачем. Серверна логіка реалізована мовою програмування C# із використанням платформи .NET та фреймворку ASP.NET Core. Обрана технологія дозволяє ефективно реалізувати REST-орієнтовану взаємодію між клієнтом і сервером та забезпечити обробку великої кількості одночасних запитів.

Для зберігання даних використовується реляційна система керування базами даних Microsoft SQL Server. Взаємодія між серверною частиною та базою даних здійснюється через Entity Framework Core, який забезпечує об’єктно-реляційне відображення та спрощує роботу з моделями даних. Такий підхід дозволяє підтримувати цілісність інформації, контролювати зв’язки між таблицями та виконувати міграції структури бази даних без порушення її стабільності. База даних містить інформацію про користувачів, їхні оцінки, списки перегляду, історію взаємодії із системою та сформовані рекомендації.

Серверна частина застосунку підтримує захищене з’єднання через протокол HTTPS, що гарантує шифрування переданих даних та зменшує ризики перехоплення конфіденційної інформації. Реалізовано механізми автентифікації та авторизації, що надають розмежування прав доступу до функціональних можливостей системи. Обробка виняткових ситуацій та логування подій надають

контроль над станом застосунку та вчасну реакцію на можливі помилки в системі.

Для коректного функціонування серверної частини необхідне середовище, що підтримує виконання застосунків платформи .NET. Це може бути операційна система Windows Server або Linux із встановленим .NET Runtime та сервером баз даних Microsoft SQL Server. Для стабільної роботи середнього навантаження достатньо серверного обладнання з не менше ніж 4 ГБ оперативної пам'яті та стабільним мережевим з'єднанням, особливо з огляду на інтеграцію із зовнішнім API TMDb, яке забезпечує отримання актуальної інформації про кінотвори.

Клієнтська частина вебзастосунку реалізована із використанням HTML5, CSS3 та JavaScript. Інтерфейс побудований відповідно до принципів адаптивного дизайну, що дозволяє коректно відображати сторінки на різних типах пристроїв, а саме від персональних комп'ютерів до мобільних телефонів. Взаємодія з сервером здійснюється шляхом надсилання HTTP-запитів і отримання відповідей у форматі JSON, що забезпечує швидкий обмін даними. Для кращого розуміння технічних характеристик було створено Таблиця 3.1.

Таблиця 3.1 – Технічні характеристики вебзастосунку

Компонент	Характеристика
Архітектура системи	Розподіл логіки між серверною та клієнтською частинами.
Серверна платформа	ASP.NET Core (.NET), мова програмування C#.
База даних	Microsoft SQL Server із використанням Entity Framework Core.
Зовнішня інтеграція	Інтеграція з TMDb API.
Безпека	Підтримка HTTPS, механізми автентифікації та авторизації користувачів.
Клієнтські технології	HTML5, CSS3, JavaScript.
Сумісність із браузерами	Google Chrome, Microsoft Edge, Mozilla Firefox.
Мінімальні вимоги до сервера	4 ГБ оперативної пам'яті, встановлений .NET Runtime та SQL Server.
Мінімальні вимоги до клієнта	Сучасний веббраузер із підтримкою JavaScript та стабільне підключення до Інтернету.
Масштабованість	Можливість розгортання у хмарному середовищі.
Продуктивність	Оптимізовані SQL-запити, кешування часто використовуваних даних.

Інтерфейс системи оптимізований для сучасних браузерів, зокрема Google Chrome, Microsoft Edge та Mozilla Firefox, один із прикладів відкриття головної сторінки подано на Рисунку А.4. Для повноцінної роботи застосунку необхідна підтримка JavaScript та стабільне підключення до мережі Інтернет. Мінімальна рекомендована роздільна здатність екрана становить 1280×720 пікселів, що дозволяє коректно відображати структурні елементи інтерфейсу та інформаційні блоки з фільмами й рекомендаціями.

Окрему увагу під час реалізації технічних характеристик було приділено питанням продуктивності та масштабованості. Архітектура системи передбачає можливість розгортання у хмарному середовищі, що дозволяє збільшувати обчислювальні ресурси у разі зростання кількості користувачів. Оптимізація запитів до бази даних, використання асинхронних методів обробки запитів і кешування часто використовуваних даних сприяють зменшенню часу відповіді сервера та покращенню загального користувацького досвіду.

Таким чином, технічні характеристики вебсистеми «MOVI.FLOW-ER» відповідають сучасним вимогам до інтерактивних вебзастосунків, забезпечують стабільну роботу серверної та клієнтської частин, належний рівень безпеки, можливість масштабування та інтеграцію із зовнішніми ресурсами. Реалізовані технічні рішення створюють надійну основу для подальшого розвитку системи та її вдосконалення відповідно до потреб різних типів користувачів.

3.4 Завантаження вебзастосунку на хостинг

Остаточне розроблення та локальне тестування вебзастосунку «MOVI.FLOW-ER» стало рушієм для наступного етапу, а саме його публікації в мережі Інтернет для забезпечення віддаленого доступу користувачів до функціоналу системи. Для розміщення проєкту було обрано безкоштовний хостинг-провайдер Free ASP Hosting, який підтримує ASP.NET та виконання серверних застосунків на платформі .NET, що повністю відповідає даній системі.

Саме такий сервіс надає можливість розгортання вебзастосунків на базі ASP.NET із підтримкою C#, хостингом баз даних та доступом через FTP-протокол. Використання даного середовища дозволило забезпечити сумісність із серверною частиною застосунку, реалізованою на ASP.NET Core, а також коректну роботу відповідних механізмів обробки HTTP-запитів, взаємодії з базою даних та авторизації користувачів.

Процес розміщення вебзастосунку складався з декількох послідовних етапів. Спочатку було здійснено реєстрацію облікового запису на платформі freeasphosting.net із підтвердженням електронної пошти. Після створення акаунта система автоматично надала домен третього рівня та параметри доступу до серверного середовища, включаючи FTP-дані для передачі файлів і налаштування бази даних. Одним з наступних кроків стало підготування вебзастосунку до публікації. У середовищі розробки було виконано збірку проєкту в режимі Release із формуванням готових до розгортання файлів. Під час публікації було перевірено коректність конфігураційного файлу appsettings.json, зокрема параметрів підключення до бази даних, які було адаптовано відповідно до серверних налаштувань хостингу.

Передавання файлів на сервер здійснювалося за допомогою FTP-клієнта. Після підключення до хостингу через надані облікові дані було завантажено всі згенеровані файли вебзастосунку до кореневої директорії сайту. Особливу увагу було приділено правильності структури каталогів, збереженню статичних ресурсів та конфігураційних файлів. Окремим етапом стало створення та налаштування бази даних на сервері. Через панель керування хостингом було створено нову базу даних, після чого виконано імпорт структури таблиць, розробленої на попередніх етапах проєктування. Параметри підключення було оновлено у конфігурації застосунку для забезпечення коректної взаємодії із серверною СКБД.

Після завершення завантаження файлів і налаштування бази даних було проведено тестування працездатності вебзастосунку безпосередньо в браузері. Було перевірено коректність маршрутизації сторінок, роботу авторизації, механізм формування рекомендацій, відображення даних із бази, а також стабільність роботи

серверної логіки. У процесі тестування усунуто незначні конфігураційні помилки, пов’язані з правами доступу до окремих каталогів.

Отже, розміщення вебзастосунку «MOVI.FLOW-ER» на безкоштовному хостингу дозволило забезпечити його публічну доступність та підтвердити працездатність у реальному серверному середовищі. Виконаний процес розгортання продемонстрував відповідність програмної реалізації вимогам до вебзастосунків клієнт-серверної архітектури та підтвердив можливість подальшого масштабування і модернізації системи.

3.5 Тестування вебзастосунку

3.5.1 Вибір та обґрунтування методів тестування вебзастосунку

Процес тестування є невід’ємною складовою розроблення програмного забезпечення та спрямований на забезпечення його відповідності визначеним функціональним і нефункціональним вимогам. Тестування має особливе значення, оскільки застосунок реалізує клієнт-серверну взаємодію, інтегрується із зовнішнім API та працює з базою даних користувачів. Наявність багаторівневої архітектури зумовлює необхідність використання комплексного підходу до перевірки як окремих компонентів, так і системи в цілому.

Під час вибору методів тестування було враховано специфіку застосунку, середовище його реалізації, а також характер основних функцій системи: реєстрація та авторизація користувачів, формування рекомендацій на основі емоційного стану, взаємодія з базою даних, обробка HTTP-запитів та динамічне відображення інформації на клієнтській стороні. З огляду на це було обрано поєднання декількох методів тестування, що забезпечують перевірку логіки, інтерфейсу та інтеграційних механізмів.

Першим етапом перевірки стала функціональна перевірка застосунку. Функціональне тестування дозволяє встановити відповідність реалізованому продукту, а саме поставленим вимогам та сценаріям використання. Даний підхід

дозволив перевірити певні аспекти, тобто створення облікового запису, авторизація користувача, вибір емоційного стану, формування рекомендацій, додавання фільмів до списку перегляду, виставлення оцінок та збереження даних у базі. Особлива увага приділялася перевірці правильності обробки введених користувачем даних та реакції системи на некоректні запити.

Другим важливим методом стало інтеграційне тестування. Оскільки система використовує інтеграцію з відкритою базою даних мультимедійного контенту через TMDb API, виникає потреба у перевірці стабільності та коректності взаємодії із зовнішніми сервісами. Інтеграційне тестування дозволило оцінити правильність формування HTTP-запитів, обробку JSON-відповідей, десеріалізацію отриманих даних та їх збереження у внутрішній структурі застосунку. Було перевірено поведінку системи у випадках затримки відповіді сервера, відсутності з'єднання або повернення помилкових кодів відповіді.

Окрему увагу було приділено модульному тестуванню, яке дозволяє перевірити окремі логічні компоненти застосунку без залучення зовнішніх залежностей. Для цього використовувалися засоби тестування, сумісні з платформою .NET, що дає змогу перевіряти коректність роботи сервісів формування рекомендацій, методів роботи з базою даних та алгоритмів фільтрації контенту. Модульне тестування дозволяє виявити логічні помилки на ранніх етапах і мінімізувати ризик поширення дефектів на інші частини системи.

Ще одним важливим аспектом перевірки стало тестування користувацького інтерфейсу. Оскільки застосунок орієнтований на інтерактивну взаємодію та персоналізацію, значення має не лише правильність обчислень, а й зручність користування. Було здійснено перевірку коректності відображення елементів інтерфейсу на різних пристроях і при різних роздільних здатностях екрана. Тестування включало перевірку адаптивності дизайну, правильність переходів між сторінками, роботу навігаційного меню та відповідність візуальних елементів заданій стилістиці.

Крім функціональних перевірок, було враховано нефункціональні аспекти тестування. До них належать тестування продуктивності, перевірка часу обробки

запитів та стабільність роботи системи під навантаженням. Оскільки вебзастосунок розміщено на безкоштовному хостингу, особливо важливо було перевірити час відповіді сервера при одночасному виконанні декількох запитів. Аналіз показав, що при стандартному навантаженні система працює стабільно та забезпечує прийнятний рівень швидкодії.

Вибір комбінованого підходу до тестування обумовлений необхідністю комплексної перевірки багаторівневої архітектури застосунку. Використання лише одного методу тестування не дозволило б забезпечити повну оцінку якості системи. Поєднання функціонального, інтеграційного, модульного та інтерфейсного тестування створює умови для всебічної перевірки застосунку, що відповідає сучасним підходам до інженерії програмного забезпечення.

Таким чином, обрані методи тестування забезпечують верифікацію коректності реалізації програмної логіки, перевірку взаємодії компонентів системи та відповідність застосунку встановленим вимогам. Комплексний підхід дозволяє мінімізувати кількість помилок у робочій версії вебзастосунку «MOVI.FLOW-ER» та підвищити рівень його надійності, стабільності й зручності використання. Застосування зазначених методів створює основу для подальшого аналізу результатів тестування та вдосконалення програмного продукту.

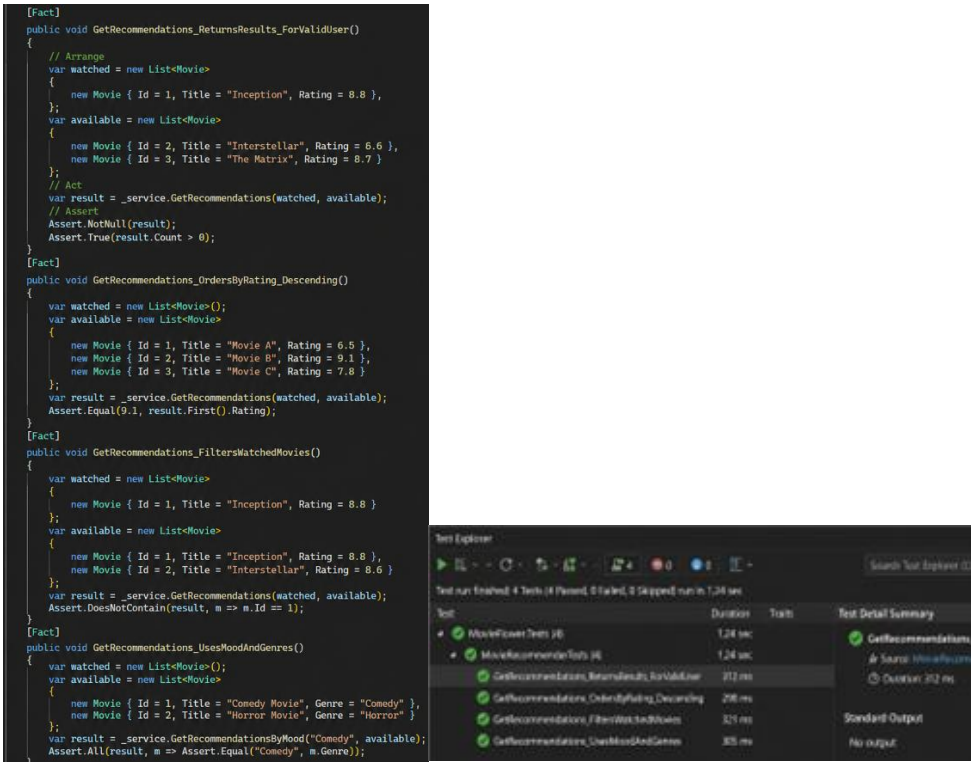
3.5.2 Перевірка на помилки за допомогою unit-тестів

Серед остаточних та ключових етапів варто виділити забезпечення якості вебзастосунку, а саме проведення модульного тестування окремих компонентів системи. Unit-тестування дозволяє перевірити коректність роботи окремих програмних модулів ізольовано від інших частин системи. Такий підхід дає можливість своєчасно виявляти логічні помилки, перевіряти відповідність очікуваних і фактичних результатів, а також гарантувати стабільність функціонування ключових сервісів після внесення змін у програмний код.

Серверна частина даної системи реалізована мовою програмування C# та

платформою ASP.NET Core, тому для проведення поточних тестувань було використано фреймворк xUnit. Для ізоляції залежностей застосовувалась бібліотека Moq, що дозволяє створювати mock-об’єкти та перевіряти поведінку сервісів без необхідності підключення до реальної бази даних або зовнішніх API.

У межах тестування було перевірено основні бізнес-логічні модулі системи, зокрема модуль рекомендацій, модуль роботи зі списком перегляду, а також перевірку обробки користувацьких даних. Особливу увагу було приділено правильності формування персоналізованих рекомендацій залежно від обраного емоційного стану користувача, оскільки цей функціонал є ключовим у концепції вебзастосунку. Даний приклад тестування зображено детально на Рисунку 3.4.



```
[Fact]
public void GetRecommendations_ReturnsResults_ForValidUser()
{
    // Arrange
    var watched = new List<Movie>
    {
        new Movie { Id = 1, Title = "Inception", Rating = 8.8 },
    };
    var available = new List<Movie>
    {
        new Movie { Id = 2, Title = "Interstellar", Rating = 8.6 },
        new Movie { Id = 3, Title = "The Matrix", Rating = 8.7 }
    };
    // Act
    var result = _service.GetRecommendations(watched, available);
    // Assert
    Assert.NotNull(result);
    Assert.True(result.Count > 0);
}
[Fact]
public void GetRecommendations_OrdersByRating_Descending()
{
    var watched = new List<Movie>();
    var available = new List<Movie>
    {
        new Movie { Id = 1, Title = "Movie A", Rating = 6.5 },
        new Movie { Id = 2, Title = "Movie B", Rating = 9.1 },
        new Movie { Id = 3, Title = "Movie C", Rating = 7.8 }
    };
    var result = _service.GetRecommendations(watched, available);
    Assert.Equal(9.1, result.First().Rating);
}
[Fact]
public void GetRecommendations_FiltersWatchedMovies()
{
    var watched = new List<Movie>
    {
        new Movie { Id = 1, Title = "Inception", Rating = 8.8 }
    };
    var available = new List<Movie>
    {
        new Movie { Id = 1, Title = "Inception", Rating = 8.8 },
        new Movie { Id = 2, Title = "Interstellar", Rating = 8.6 }
    };
    var result = _service.GetRecommendations(watched, available);
    Assert.DoesNotContain(result, m => m.Id == 1);
}
[Fact]
public void GetRecommendations_UsesMoodAndGenres()
{
    var watched = new List<Movie>();
    var available = new List<Movie>
    {
        new Movie { Id = 1, Title = "Comedy Movie", Genre = "Comedy" },
        new Movie { Id = 2, Title = "Horror Movie", Genre = "Horror" }
    };
    var result = _service.GetRecommendationsByMood("Comedy", available);
    Assert.All(result, m => Assert.Equal("Comedy", m.Genre));
```

Рисунок 3.4 – Виконання тестових випадків

Тестування сервісу рекомендацій перевіряє метод отримання рекомендацій непорожній список фільмів для конкретного емоційного стану. У тесті задавався очікуваний набір даних, після чого здійснювалась перевірка кількості отриманих елементів та відсутності null-значень у результаті. Якщо фактичний результат співпадав із очікуваним, тест вважався успішно пройденим.

Аналогічним чином було протестовано модуль роботи зі списком перегляду. Зокрема, перевірялось, чи коректно додається фільм до списку користувача та чи повертається позитивний результат виконання операції. У межах тесту здійснювалась імітація виклику методу додавання фільму до Watchlist із передачею ідентифікатора користувача та ідентифікатора фільму. Після виконання методу перевірялось, чи повернуто значення true та чи збережено відповідні параметри.

Запуск unit-тестів здійснювався за допомогою вбудованих засобів Visual Studio. У результаті виконання тестового набору всі створені тести були успішно пройдені, що підтверджується відповідним звітом про виконання (див. згенероване зображення з результатами тестування). У звіті відображено кількість виконаних тестів, час їх виконання та статус, що свідчить про відповідність очікуваних і фактичних результатів.

Отже, дане модульне тестування дозволило підтвердити дійсність реалізації ключових функціональних компонентів вебзастосунку «MOVI.FLOW-ER». Використання тестів забезпечує підвищення надійності програмного продукту, спрощує подальшу підтримку та масштабування системи, а також зменшує ризик виникнення помилок при розширенні функціоналу.

3.5.3 Валідація та верифікація вебзастосунку

Валідація та верифікація вебзастосунку «MOVI.FLOW-ER» здійснювалась для контролю якості програмного продукту після реалізації основного функціоналу та проведення модульного тестування. Метою даного етапу було підтвердження відповідності розробленої системи визначеним функціональним і нефункціональним вимогам, а також перевірка коректності технічної реалізації клієнтської та серверної частин.

Валідація застосунку проводилася у двох напрямах, а саме була технічна валідність вебінтерфейсу та логічна валідність обробки даних. Технічна валідність клієнтської частини перевірялася за допомогою автоматизованих механізмів, які

демонструють стандарти HTML та CSS. Застосування сервісу W3C Validator дозволило проаналізувати структуру HTML-документів на відповідність специфікації HTML5. У процесі перевірки здійснювався аналіз правильності вкладеності тегів, наявності обов'язкових атрибутів, коректності використання семантичних елементів та відсутності застарілих конструкцій. Виявлені синтаксичні неточності були усунені, що забезпечило відповідність вебсторінок стандартам та покращило їх кросбраузерну сумісність.

Одним з важливих пунктів був аналіз на коректність стилізації інтерфейсу. CSS-файли було проаналізовано на предмет дублювання властивостей, конфліктів селекторів та надлишкових правил. Оптимізація стилів дозволила зменшити обсяг статичних ресурсів та підвищити швидкість завантаження сторінок. Таким чином, технічна валідація сприяла підвищенню стабільності відображення інтерфейсу на різних пристроях та у різних браузерах.

Логічна валідність форм введення даних перевірялася шляхом тестування механізмів клієнтської та серверної валідації. На стороні клієнта використовувалися вбудовані механізми перевірки HTML-форм та JavaScript-обробники подій, що забезпечували контроль обов'язкових полів, перевірку формату електронної пошти, обмеження довжини текстових полів та недопущення введення некоректних значень. На серверній стороні реалізовано додаткову перевірку моделей даних із використанням механізмів атрибутів валідації у C#, що унеможливлює збереження некоректних або небезпечних даних у базі.

Верифікація вебзастосунку проводилася шляхом зіставлення реалізованих функцій із вимогами, сформульованими у технічному завданні. Для цього було створено матрицю відповідності «вимога – реалізація – результат перевірки», що дозволило систематизувати процес контролю. Перевірено реалізацію таких функціональних можливостей:

— реєстрація користувача, авторизація та автентифікація;
— збереження профілю, вибір емоційного стану;
— інтеграція із зовнішнім API для отримання інформації про фільм;
— додавання до списку перегляду, виставлення оцінок та формування історії.

Під час перевірки механізму формування рекомендацій аналізувалася коректність передачі параметрів запиту до зовнішнього API, правильність обробки отриманого JSON-відповіді та відображення результатів у користувацькому інтерфейсі. Було підтверджено, що система формує рекомендації відповідно до обраного емоційного стану та забезпечує фільтрацію результатів без помилок обробки даних. Також перевірено коректність роботи механізму, що відповідає за збереження вибраних кінотворів у списку перегляду та уникнення дублювання записів у базі даних.

Верифікація нефункціональних вимог включала перевірку швидкодії, стабільності та безпеки. Швидкодія оцінювалася шляхом вимірювання часу відповіді сервера на типові запити користувача. Аналіз показав, що середній час формування сторінки з рекомендаціями та даними про фільм відповідає допустимим показникам і не перевищує встановлених порогових значень. Стабільність перевірялася шляхом багаторазового виконання однотипних сценаріїв використання без виникнення збоїв або втрати даних.

Безпекова верифікація передбачала перевірку механізмів автентифікації, правильності зберігання паролів у зашифрованому вигляді та захисту від типових вебзагроз, таких як SQL-ін’єкції та некоректна передача даних. Використання ORM-засобів доступу до бази даних та параметризованих запитів мінімізувало ризики несанкціонованого втручання.

Окремо було перевірено відповідність інтерфейсу принципам зручності використання. Проведено тестування основних сценаріїв взаємодії користувача із системою, зокрема реєстрації, вибору емоційного стану та перегляду рекомендацій. За результатами перевірки встановлено, що навігація є логічною, а структура сторінок відповідає визначеній моделі організації інформації.

Отже, дані дії, а саме перевірки валідації дозволили підтвердили технічну коректність реалізації клієнтської частини та механізмів обробки даних, а верифікація засвідчила повну відповідність реалізованого функціоналу вимогам технічного завдання. Отримані результати дозволяють зробити висновок про цілісність архітектури системи, стабільність її роботи та готовність вебзастосунку.

3.5.4 Аналіз результатів тестування вебзастосунку

Після проведення модульного тестування, інтеграційної перевірки, валідації та верифікації вебзастосунку «MOVI.FLOW-ER» було виконано узагальнений аналіз отриманих результатів з метою оцінки рівня якості реалізованої системи, стабільності її роботи та відповідності встановленим вимогам. Даний етап дозволив систематизувати результати тестування та визначити ступінь готовності програмного продукту до практичного використання.

Даний огляд функціонального тестування перевірено реалізацію всіх ключових сценаріїв взаємодії користувача із системою. Зокрема, було протестовано процеси реєстрації та авторизації, збереження профілю користувача, формування персоналізованих рекомендацій на основі емоційного стану, додавання кінотворів до списку перегляду, виставлення оцінок та перегляд детальної інформації про фільми. За результатами перевірки встановлено, що всі визначені функції працюють відповідно до логіки, описаної у технічному завданні. Поточні результати тестування можна зобразити у Таблиці 3.2.

Таблиця 3.2 – Результати тестування

Критерій перевірки	Результат	Висновок
Функціональна коректність	Сценарії виконано	Відповідність вимогам
Модульне тестування	100% тестів	Логіка реалізована
Інтеграція з API	Обробка даних	Стабільність
Валідація форм	Введення запитів	Захист даних
Продуктивність	Час відповіді	Відповідність вимогам
Безпека	Паролі хешуються	Базовий рівень захисту

Модульне тестування серверної логіки продемонструвало високий рівень коректності обробки даних. Усі тести завершилися успішно, що підтвердило правильність реалізації алгоритмів формування рекомендацій, роботи з базою даних та обробки відповідей від зовнішнього API. Очікувані результати співпали з фактичними значеннями, що свідчить про відсутність логічних помилок у ключових компонентах системи.

Інтеграційне тестування підтвердило коректність взаємодії між клієнтською та серверною частинами. Було перевірено передачу даних через HTTP-запити, правильність маршрутизації, обробку JSON-відповідей та відображення отриманої інформації в інтерфейсі користувача. Усі сценарії взаємодії виконувалися без збоїв, що свідчить про узгодженість роботи компонентів архітектури.

Аналіз нефункціональних показників засвідчив, що система відповідає вимогам швидкодії та стабільності. Середній час відповіді сервера на стандартний запит користувача перебуває в межах допустимих значень, що забезпечує комфортну взаємодію із застосунком. Навіть за умов багаторазового виконання однотипних операцій не спостерігалося деградації продуктивності або накопичення помилок.

Отримані результати демонструють повну відповідність реалізованого вебзастосунку до поставлених задач та визначених технічних вимог. У ході тестування не виявлено критичних недоліків, що перешкоджали б експлуатації системи або впливали на цілісність даних. Незначні зауваження, пов’язані з оптимізацією інтерфейсу, були усунені на завершальному етапі проєктування. Це дозволило підвищити загальну якість користувацького досвіду та забезпечити більш логічну й послідовну взаємодію з функціоналом системи.

Загальний аналіз результатів тестування підтверджує, що система є стабільною, функціонально завершеною та придатною для хостингу. Проведена перевірка забезпечила впевненість у надійності роботи вебзастосунку та його готовності до використання кінцевими користувачами в умовах реального середовища. Дані архітектурні рішення та обрані технології повністю підтримують подальше масштабування й модернізацію системи без суттєвих змін у структурі.

3.6 Висновки. Програмна реалізація

Проєктування кваліфікаційної роботи успішно пройшло повний цикл програмної реалізації та тестування інтерактивної вебсистеми персоналізованого підбору фільмів «MOVI.FLOW-ER». Реалізація програмного продукту виконувалась відповідно до сформованих у попередніх розділах функціональних та нефункціональних вимог, а також на основі розробленої архітектурної моделі та структури бази даних. Отримані результати підтверджують можливість створення повноцінної вебплатформи, орієнтованої на врахування емоційного стану користувача під час формування рекомендацій мультимедійного контенту.

Розроблення бази даних дозволило забезпечити повне зберігання інформації про користувачів, кінотворів, оцінки, списки перегляду, рекомендації та емоційні стани. Структура бази даних відповідає принципам нормалізації, що дозволило уникнути надлишковості даних та забезпечити цілісність інформації. Реалізовані зв’язки між сутностями забезпечують логічну узгодженість операцій збереження, оновлення та видалення даних. Такий підхід сприяв створенню стабільного інформаційного середовища, яке є основою для коректної роботи рекомендаційного механізму.

Програмна реалізація вебзастосунку виконувалася з використанням сучасних засобів розробки на платформі .NET із застосуванням мови програмування C#. Серверна частина забезпечує обробку запитів користувача, взаємодію з базою даних, інтеграцію із зовнішнім API для отримання актуальної інформації про кінотворів, а також формування персоналізованих рекомендацій. Клієнтська частина реалізує зручний та інтуїтивно зрозумілий інтерфейс, що дозволяє користувачеві швидко взаємодіяти із системою, обирати емоційний стан, переглядати добірки, додавати фільми до списків та здійснювати оцінювання.

Реалізація програмних модулів забезпечило функціонування системи реєстрації та авторизації користувачів, модулю керування профілем, механізму формування рекомендацій, модуля роботи зі списками перегляду, а також інтеграції з зовнішньою базою даних кінотворів. Усі модулі розроблялися згідно

принципів модульності та розділення відповідальностей, що спрощує підтримку коду та його подальше масштабування. Такий підхід забезпечує гнучкість системи та можливість розширення функціоналу без суттєвих змін у базовій архітектурі.

Окрему увагу було приділено розгортанню вебзастосунку на хостингу. Проєкт було успішно розміщено на безкоштовному серверному середовищі, що дозволило перевірити його працездатність в умовах реального мережевого середовища. У процесі публікації було виконано налаштування підключення до бази даних, конфігурацію параметрів безпеки та тестування доступності ресурсу через мережу. Результати розгортання підтвердили коректність конфігурації серверної частини та стабільність функціонування вебзастосунку.

Важливим етапом третього розділу стало тестування системи. Було обґрунтовано вибір методів тестування, що охоплюють функціональну перевірку, модульне тестування, верифікацію відповідності технічним вимогам та валідацію коректності відображення інтерфейсу. Реалізовані unit-тести дозволили перевірити логіку окремих компонентів, зокрема алгоритм формування рекомендацій, обробку емоційного стану користувача та механізм взаємодії з базою даних. Отримані результати тестування засвідчили відповідність очікуваних та фактичних результатів роботи програмних модулів.

Проведена валідація вебзастосунку підтвердила коректність структури HTML-документів, відсутність критичних помилок розмітки та відповідність основним стандартам веброзробки. Верифікація функціоналу показала повну відповідність реалізованих можливостей поставленій задачі кваліфікаційної роботи. Усі ключові функції, визначені у вимогах, були реалізовані та протестовані в реальних сценаріях використання.

Аналіз результатів тестування продемонстрував стабільність роботи системи при стандартному навантаженні та коректність обробки запитів користувача. Не було виявлено критичних помилок, що могли б призвести до втрати даних або некоректної роботи застосунку. Оптимізація окремих елементів інтерфейсу проведена на одному з останніх етапів. Саме даний підхід дозволив підвищити зручність використання системи та покращити загальне сприйняття інтерфейсу.

Результати програмної реалізації демонструють повністю розроблений вебзастосунок, який відповідає сучасним вимогам до інтерактивних мультимедійних платформ. Система поєднує персоналізований підхід, інтеграцію з відкритими джерелами даних та зручний інтерфейс користувача. Реалізований механізм врахування емоційного стану користувача дозволяє підвищити відповідність рекомендацій та скоротити час на пошук відповідного контенту.

Практична цінність отриманого результату полягає у можливості подальшого розвитку вебсистеми, зокрема впровадження більш складних алгоритмів аналізу поведінки користувачів, розширення функціоналу соціальної взаємодії та оптимізації продуктивності при зростанні кількості користувачів. Створена архітектура дозволяє масштабувати систему та адаптувати її до нових вимог без суттєвих змін базових компонентів.

Отже, в процесі виконання окреслених задач кваліфікаційної роботи, було успішно реалізовано програмну частину інтерактивної системи «MOVI.FLOW-ER», проведено її тестування та підтверджено відповідність поставленій меті. Отримані результати надають інформацію про завершеність програмного продукту, його працездатність та готовність до використання в реальному середовищі. Реалізована система демонструє доцільність застосування емоційно-орієнтованого підходу до підбору фільмів та підтверджує актуальність подальших досліджень персоналізованих рекомендаційних систем.

Висновки

Виконання кваліфікаційної роботи полягало у розробці системи персоналізованого підбору фільмів «MOVI.FLOW-ER», орієнтованого на формування рекомендацій відповідно до емоційного стану користувача. Основною ідеєю проєкту стало створення зручного інструменту для швидкого та обґрунтованого вибору фільмів, серіалів, мультфільмів та аніме в умовах значного інформаційного перевантаження. Розроблений інтерфейс сприяє швидкій навігації, формуванню персональних списків перегляду та підвищенню загального рівня користувацького досвіду.

Мета роботи полягала у розроблені інтерактивної системи для підбору фільмів, яка відповідатиме сучасним вимогам розробки та тенденціям створення рекомендаційних систем. Запропонована система орієнтована на врахування емоційного стану користувача з метою підвищення рівня персоналізації та покращення користувацького досвіду взаємодії з мультимедійним контентом.

Значною перевагою розробленої системи є логічна організація ролей та функціональних можливостей. Передбачено роль користувача з доступом до персоналізованих рекомендацій, списків перегляду та оцінювання контенту, а також адміністративну частину для керування інформацією про кінотворів, редагування даних та підтримання актуальності бази. Розмежування доступу дозволяє забезпечити цілісність даних, контроль за змінами в системі та стабільність її функціонування.

Для реалізації проєкту було використано мову програмування C# та платформу .NET із застосуванням фреймворку ASP.NET Core MVC. Обрані технології забезпечують чітке розділення компонентів системи на модель, представлення та контролер, що позитивно впливає на підтримку і масштабованість програмного продукту. Архітектурний підхід дозволив організувати структуровану взаємодію між клієнтською та серверною частинами застосунку. Зберігання даних реалізовано із використанням SQL Server, що гарантує надійність обробки інформації, списки перегляду та рекомендації.

Важливим етапом розробки стало проєктування бази даних та визначення взаємозв’язків між сутностями. Структура БД включає таблиці користувачів, кінотворів, жанрів, емоційних категорій, оцінок і списків перегляду. Було забезпечено коректну нормалізацію даних та реалізовано зв’язки між таблицями для автоматизованого отримання рекомендацій. Продумана структура дозволяє уникнути дублювання інформації, підвищує швидкодію запитів та спрощує подальше розширення функціоналу системи. Інтеграція всіх компонентів у межах єдиної клієнт-серверної архітектури забезпечує узгоджену роботу програмного забезпечення навіть при зростанні кількості користувачів.

Тестування програмного забезпечення включало перевірку основних функціональних модулів, а саме реєстрації та авторизації користувачів, формування рекомендацій, додавання кінотворів до списку перегляду та виставлення оцінок. Було створено модульні тести, які підтвердили коректність бізнес-логіки та обробки вхідних даних. Перевірено валідацію форм, правильність маршрутизації запитів і стабільність взаємодії з базою даних. Проведене тестування засвідчило відсутність критичних помилок та відповідність системи поставленим вимогам.

Розроблений вебзастосунок може стати основою створення повноцінної рекомендаційної платформи, де система демонструє можливість ефективного поєднання емоційного підходу до підбору контенту з сучасними вебтехнологіями. Подальший розвиток проєкту може передбачати впровадження складніших алгоритмів рекомендацій, розширення аналітики поведінки користувачів та інтеграцію із зовнішніми сервісами кіноіндустрії. Було досягнуто поставленої мети, реалізовано функціонально завершений вебзастосунок, підтверджено доцільність використання обраних технологій та архітектурних рішень. Отримані результати свідчать про можливість практичного застосування системи та її подальшого масштабування відповідно до вимог сучасних технологій.

Перелік джерел посилання

1. НАЗАРЕНКО, Єлисей Валерійович. *Розробка веб-сайту огляду фільмів з інтеграцією IMDb API та YouTube API*. Івано-Франківськ: ЗВО" Університет Короля Данила", факультет суспільних та прикладних наук, кафедра інформаційних технологій, 2024. 9 с.

2. ОБЕРЕМКО, П. Ю. Вебдодаток для спільного перегляду фільмів та спілкування в реальному часі. 2024. 7 с.

3. НИШТА, Ірина Іванівна. *Розробка мобільного застосунку для рекомендацій фільмів з використанням технологій React Native*. 2025. Bachelor's Thesis. ТНТУ імені Івана Пулюя. 11 с.

4. TENNAKOON, Nimasha; SENAWEERA, Oshada; DHARMARATHNE, H. A. S. G. Emotion-based movie recommendation system. *International Journal on Advances in ICT for Emerging Regions (ICTer)*, 2024, 17.1. 1 с.

5. YERKIN, Adilet, et al. Multi-channel emotion analysis for consensus reaching in group movie recommendation systems. *arXiv preprint arXiv:2404.13778*, 2024. 1 с.

6. РУДЕНКО, Євненія Павлівна. *Інформаційна технологія забезпечення використовуваності та веб-доступності стримінгової платформи*. 2022. Master's Thesis. Сумський державний університет. 10 с.

7. NETFLIX. URL: https://about.netflix.com/en (дата звернення: 16.04.26).

8. Disney+. URL: https://www.disney.co.uk/ (дата звернення: 16.04.26).

9. UAKINO. URL: https://uakino.best/ua/ (дата звернення: 16.04.26).

10. MEGOGO. URL: https://megogo.net/ua (дата звернення 16.04.26).

11. SWEET.TV. URL: https://sweet.tv/uk (дата звернення 16.04.26).

12. КІМАК, Володимир. Вибір нефункціональних вимог при розробці програмного забезпечення. In: *2023 2nd International Conference on Innovative Solutions in Software Engineering (ICISSE)*. 2023. p. 237. 234 с.

13. YESINA, M. V.; KRAVCHENKO, A. A.; KRAVCHENKO, S. O. Огляд загроз безпеці та цілісності даних у хмарних обчисленнях. *Radiotekhnika*, 2023, 212: 30-35. 2 с.

14. ШЕЛЕСТ, В. А. Дослідження питання обрання архітектури вебзастосунку на основі аналізу вимог прикладної задачі. 2025. 8 с.

15. БУДЬОННИЙ, М. А. Проектування архітектури інтернет-магазину. 2024. 1 с.

16. КРАВЧУК, О. А.; СИНЮК, О. М. Переваги та недоліки мікросервісної архітектури. 1 с.

17. ПОСВІСТАК, В. С.; ДЕМКІВСЬКА, Т. І. Клієнт-серверна архітектура та її використання при розробці програмного забезпечення. *Інформаційні технології в науці, виробництві та підприємництві*, 2020. 2 с.

18. СТАРІЧЕНКО, В. С. Порівняльний аналіз оптимізації архітектури обробки сповіщень на стороні Back-end: синхронного REST API та асинхронного підходу з Apache Kafka. 2025. 14 с.

19. ДЕМИДЕНКО, М. А. Введення в сучасні бази даних. 2020. 6 с.

20. ДОЦЕНКО, Сергій Ілліч. Організація та системи керування базами даних. 2023. 19 с.

21. ДИДО, Р. А., et al. ПРОЄКТУВАННЯ ВЕБСИСТЕМИ З БАЗОЮ ДАНИХ ДЛЯ ОНЛАЙН-БРОНЮВАННЯ НОМЕРІВ У ГОТЕЛІ НА ОСНОВІ АРХІТЕКТУРИ MVC. *Вісник ХНТУ*, 2025, 2.2 (93): 141-148. 3 с.

22. ГАВРИСЮК, О. Б. Вибір ефективних механізмів автентифікації для розроблення Web-застосунків з різною архітектурою та навантаженням. 2025. 30 с.

23. ПЛЯЦИК, Олександр Віталійович. Серверна частина системи управління взаємовідносинами з клієнтами. 2025. 8 с.

24. ЧЕРКАСОВА, Е. С. Програмна система для організації та управління груповими подорожами. Клієнтська частина. 2025. 17 с.

25. ОСНІЦЬКИЙ, Іван Євгенович. Розробка дизайну інтерфейсу комп'ютерної гри. In: *Наукова онлайн конференція «Креативна трансформація та модернізація сучасного суспільства»*. СГ НТМ «Новий курс», 2024. 6 с.

26. MOTSYK, ROSTISLAV; HRYNCHAK, OLEKSANDR. ВЕБ-ТЕХНОЛОГІЇ ТА ДИЗАЙН. *Хмельницького національного університету*, 2021, 22. 22 с.

27. МАТОЛІЧ, Ірина Яремівна; ЛЕВИЦЬКА, Роксолана Романівна. *Ключові тренди графічного дизайну 2025 року: мінімалізм, AI і турбота про довкілля*. 2025. PhD Thesis. Hamburg, Germany: International Science Group. 4 с.

28. БІЛЕЦЬ, Д. Ю. Дослідження впливу вподобань користувачів на колірне рішення дизайну мультимедійного видання. 2025. 10 с.

29. ЯРОВИЙ, Володимир; БЄЛЯВСЬКА, Оксана. Шрифт у створенні ефективного дизайну: відбір шрифтів та їх вплив на читаність. *Українська культура: минуле, сучасне, шляхи розвитку*, 2024, 49: 495-502. 3 с.

30. ЯДРОВА, Марина Василівна; НОВАК, Іван Сергійович. МОЖЛИВОСТІ ТА ПРОБЛЕМИ ВИКОРИСТАННЯ СУЧАСНИХ СТЕКІВ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ В РОЗРОБЦІ ВЕЛИКИХ WEB-СИСТЕМ ЕЛЕКТРОННОЇ КОМЕРЦІЇ B2C. In: *The 4 th International scientific and practical conference "Modern problems of science, education and society"(June 19-21, 2023) SPC "Sci-conf. com. ua", Kyiv, Ukraine. 2023. 1281 p.* 2023. p. 301. 301 с.

31. MOTSYK, ROSTISLAV; HRYNCHAK, OLEKSANDR. ВЕБ-ТЕХНОЛОГІЇ ТА ДИЗАЙН. *Хмельницького національного університету*, 2021, 22. 25 с.

32. ПАВЛЕНКО, Юлія Степанівна. Верстка веб-сторінок з допомогою HTML та CSS: методичні рекомендації. 2022. 52 с.

33. ЖЕРЕБЦОВ, О. А.; ІЖИКОВ, А. Ю. Використання функціонального програмування у JavaScript та фреймворках. *Інноваційні рішення в інженерії програмного забезпечення*, 2022, 9: 162-169. 169 с.

34. СТОЄВ, Є. Д.; РАЛЕНКО, В. С. ВИКОРИСТАННЯ ТЕХНОЛОГІЇ .NET ДЛЯ РОЗРОБКИ МАСШТАБОВАНИХ БІЗНЕС-ПЛАТФОРМ. *ТЕЗИ ДОПОВІДЕЙ ТЕХНІЧНІ НАУКИ*, 2024, 77. 79 с.

35. КОНОВАЛЕНКО, Ігор Володимирович; МАРУЩАК, Павло Орестович. Платформа. NET та мова програмування C# 8.0. 2020. 14 с.

36. ФІЛІМОНЧУК, Т. В.; НАСТЕНКО, О. С. Аналіз актуальності використання шаблону MVC при розробці сучасних веб-застосунків. 2021. 1 с.

37. ХОМИШИН, В. Г. Захист ASP. NET CORE Веб-додатків від впровадження SQL-коду. *Збірник тез доповідей XIII Міжнародної науково-практичної конференції молодих учених та студентів „Актуальні задачі сучасних технологій “*, 2024, 432-433. 2 с.

38. СУХАРСЬКИЙ, Віталій Олександрович. *Розробка Android-застосунку для управління складом з використанням мови програмування Java, та бази даних SQL server*. 2024. Master's Thesis. Тернопільський національний технічний університет імені Івана Пулюя. 14 с.

39. ЧИНЧИК, Д.; КОРОБЕЙНІКОВА, Т.; ЗАХАРЧЕНКО, С. Методи та засоби комплексного захисту корпоративної мережі. *EDITOR COORDINATOR*, 2021, 433. 441 с.

40. КОСТЮК, Юлія Володимирівна. Використання діаграми діяльності для моделювання та аналізу процесу прийняття рішень. *Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення (випуск 78): матеріали Міжнародної наукової інтернет-конференції,(м. Тернопіль, Україна–м. Переворськ, Польща, 8-9 червня 2023 р.)/[редкол.: О. Патряк та ін.]; ГО “Наукова спільнота”; WSSG w Przeworsku.–Тернопіль: ФО-П Шпак ВБ–204 с.–ISSN 2522-932X*, 2023, 60. 61 с.

41. САЛТАН, С. О. AI-орієнтована система «CineMind» для персоналізованих рекомендацій фільмів. 2025. 41 с.

42. ОЛІЙНИК, Дарина. Робота з даними в форматі JSON. 2020. 4 с.

ДОДАТОК А

(обов’язковий)

ГРАФІЧНІ МАТЕРІАЛИ

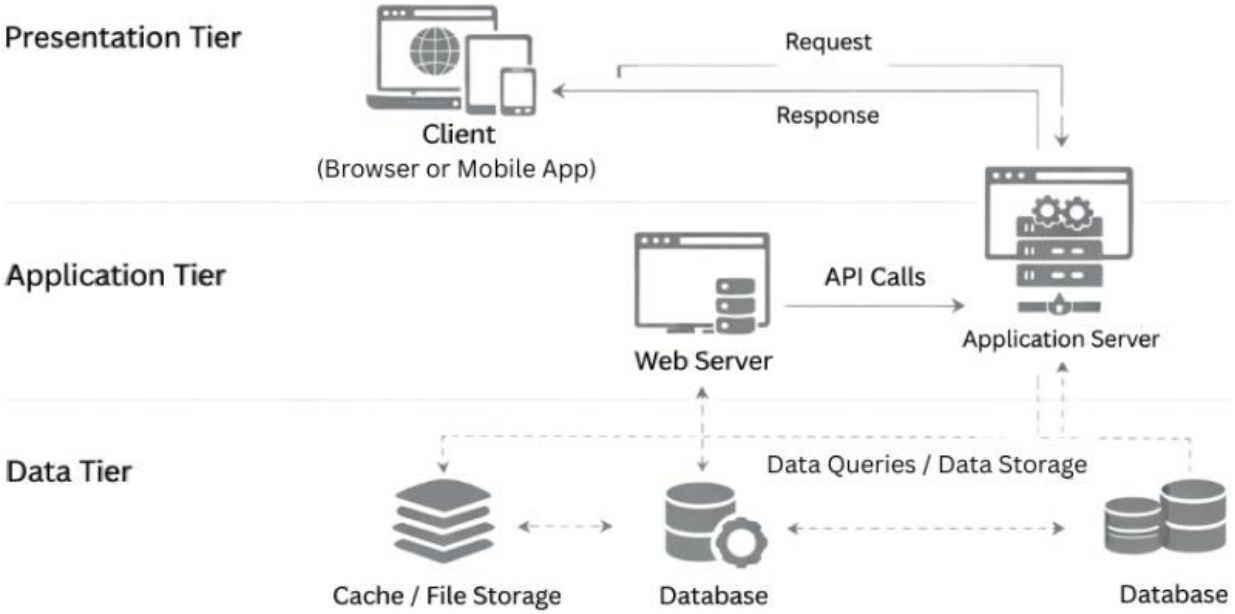


Рисунок А.1 – Web Application Architecture

Рисунок А.2 – Діаграма міжмодульних зв’язків

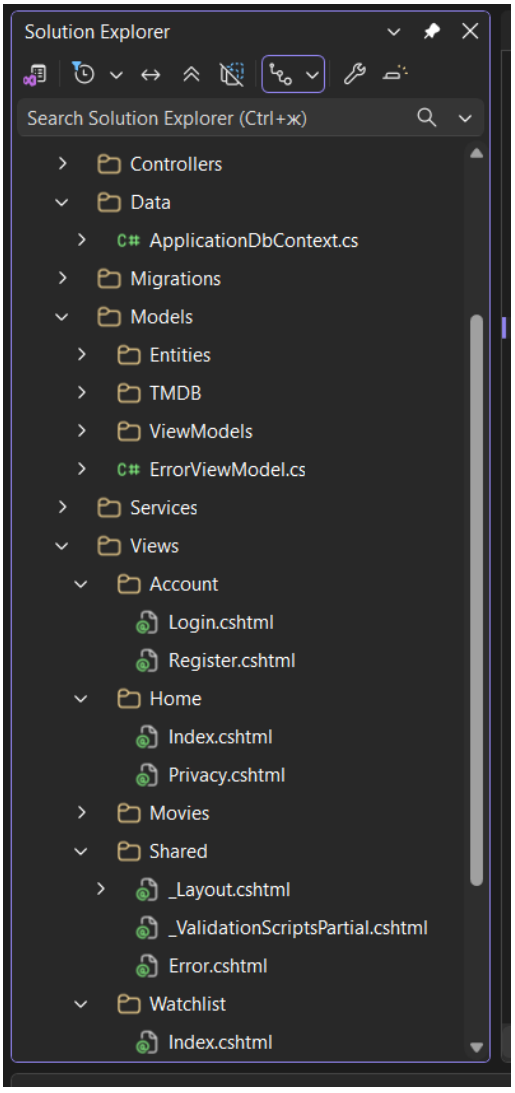


Рисунок А.3 – Зміст файлів проєкту

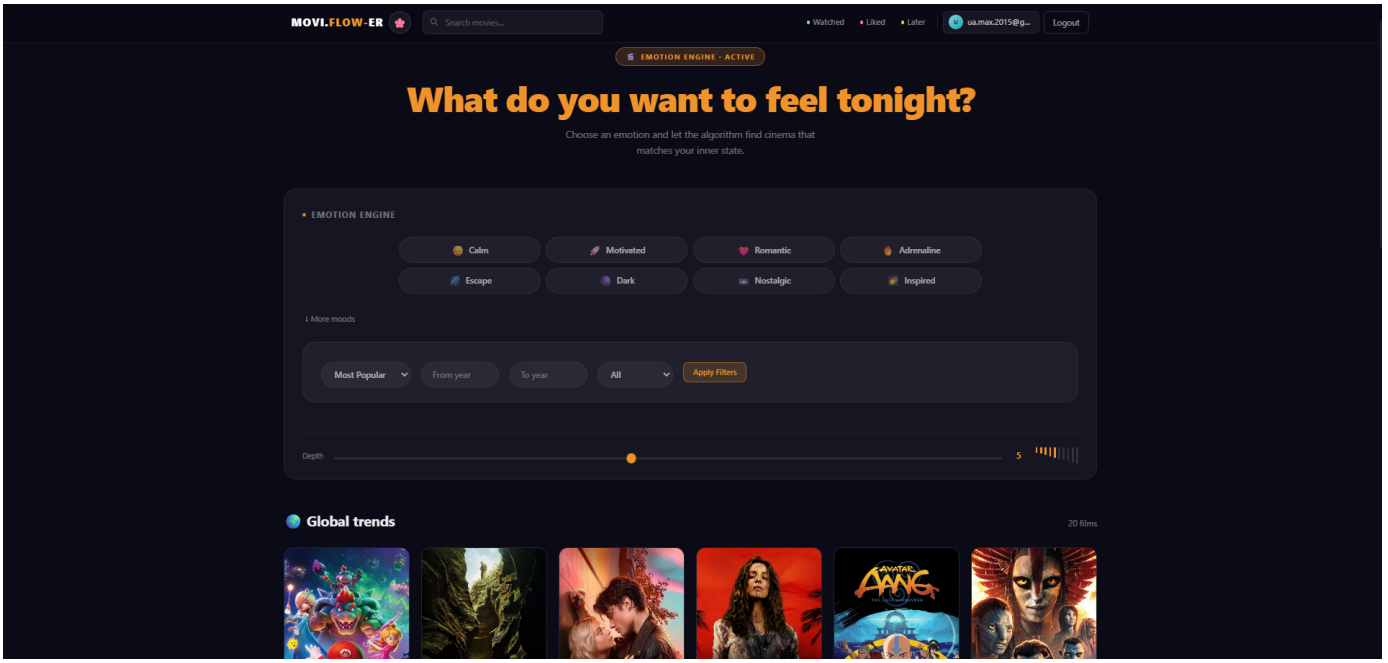


Рисунок А.4 – Приклад головного екрану у Google Chrome

ДОДАТОК Б

(обов’язковий)

ТЕХНІЧНЕ ЗАВДАННЯ

на розробку інтерактивної системи для підбору кінотворів на основі емоційного стану користувача «MOVI.FLOW-ER»

1. Перелік умовних скорочень

№	Термін	Скорочення	Коментар
1	Вебзастосунок	ВЗ	Програмний продукт, доступний через веббраузер
2	База даних	БД	Система для зберігання даних про основні сутності
3	Емоційний стан	ЕС	Параметр, що обирає користувач для отримання рекомендацій
4	Інтерфейс коритсувача	UI	Частина системи, з якою безпосередньо взаємодіє користувач
5	Система управління базами даних	СУБД/СКБД	Програмне забезпечення для створення та ведення БД
6	Model-View-Controller	MVC	Архітектурний шаблон для розподілу логіки програми
7	The Movie Database API	TMDb API	Зовнішній сервіс для отримання інформації про кінотвори
8	JSON	JSON	Формат обміну даними між клієнтом і сервером

2. Загальні відомості

Інтерактивна система «MOVI.FLOW-ER» має на меті підвищити якість вибору кінотворів (фільмів, серіалів, мультфільмів, аніме) шляхом урахування емоційного стану користувача як ключового критерію персоналізації. На відміну від традиційних рекомендаційних платформ, які здебільшого орієнтуються на жанрові вподобання, рейтинги або популярність контенту, пропонована система враховує суб'єктивний психологічний контекст. Завдяки цьому формування рекомендацій відбувається не лише на основі статистичних даних чи історії переглядів, але й з урахуванням актуального стану користувача в момент звернення до сервісу.

Система забезпечує персоналізовані пропозиції на основі настрою, жанрових уподобань та накопиченої історії взаємодії, що дозволяє суттєво скоротити час, необхідний для пошуку релевантного контенту, та підвищити загальний рівень користувацького досвіду. Такий підхід сприяє зменшенню інформаційного перевантаження, характерного для сучасних медіа-бібліотек, і робить процес вибору кінотвору більш інтуїтивним, емоційно комфортним та заощадливим у часовому вимірі.

2.1. Призначення системи

Призначенням вебзастосунку «MOVI.FLOW-ER» є надання користувачам зручного інструменту для отримання релевантних рекомендацій кінотворів відповідно до їхнього поточного емоційного стану. Система орієнтована на:

— персоналізований підбір кінотворів, рекомендації формуються з урахуванням обраного настрою, жанрових вподобань та історії переглядів;

— інтерактивну взаємодію, можливість оцінювати фільми, додавати до списку перегляду, переглядати детальну інформацію;

— інтеграцію з відкритим джерелом даних; використання TMDb API для отримання актуальної інформації про кінотвори;

— підвищення доступності контенту, забезпечення зручного пошуку, фільтрації за жанрами, роком випуску, популярністю.

2.1 Загальні положення

Розробка вебзастосунку «MOVI.FLOW-ER» виконується відповідно до

стандарту ISO/IEC 12207 та передбачає такі етапи:

— аналіз предметної області та формування вимог;

— проєктування архітектури;

— проєктування структури бази даних;

— розробка програмних модулів;

— тестування та валідація;

— розгортання на хостингу та супровід.

2.2 Повне найменування системи та умовне позначення

Повне найменування: Інтерактивна система для підбору фільмів на основі стану користувача «MOVI.FLOW-ER».

Умовне позначення: MOVI.FLOW-ER.

2.3.1. Технології розробки

До стеку технологій розробки належить такі поняття:

— мова програмування: C#;

— фреймворк: ASP.NET Core MVC;

— клієнтські технології: HTML5, CSS3, JavaScript;

— СКБД: Microsoft SQL Server;

— ORM: Entity Framework Cor;

— зовнішній API: TMDb API.

2.4. Найменування замовника та виконавця

Замовник: Приймак Максим Олександрович.

Виконавець: Приймак Максим Олександрович.

2.5. Цільова аудиторія

Цільова аудиторія вебзастосунку «MOVI.FLOW-ER» задіює широке коло користувачів, які цікавляться кінотворами та прагнуть швидко знаходити контент відповідно до настрою:

— кіномани та глядачі, які часто стикаються з проблемою вибору фільму;

— користувачі, що бажають отримувати персоналізовані рекомендації;

— особи, які цінують емоційну відповідність контенту поточному стану;

— молоді люди, активні користувачі стрімінгових сервісів;

— дослідники в галузі рекомендаційних систем.

3. Призначення та цілі створення програмного забезпечення

3.1. Мета створення вебзастосунку

Метою розробки є створення інтерактивної вебсистеми, яка забезпечує персоналізований підбір кінотворів на основі емоційного стану користувача, відповідає сучасним вимогам веброзробки та рекомендаційних систем, а також зменшує час пошуку контенту.

3.2. Характеристики об'єкта автоматизації

Об'єктом автоматизації є процес підбору кінотворів користувачем, який включає:

— визначення поточного емоційного стану;

— отримання рекомендацій з урахуванням історії оцінок та переглядів;

— управління персональними списками перегляду;

— перегляд детальної інформації про кінотвори.

Користувачами системи є зареєстровані особи та незареєстровані гості з обмеженим доступом.

3.3. Основні функції продукту

Функції для користувача:

— реєстрація та авторизація;

— вибір емоційного стану;

— отримання персоналізованих рекомендацій;

— перегляд каталогу кінотворів;

— пошук, сортування та фільтрація за жанром, роком, популярністю;

— перегляд детальної інформації;

— додавання кінотворів до списку перегляду;

— оцінювання кінотворів.

3.4. Конкурентні переваги

Урахування емоційного стану – на відміну від більшості сервісів, що базуються лише на жанрах чи популярності.

Інтеграція з TMDb API – забезпечує актуальний та структурований контент без ручного наповнення.

Модульна архітектура ASP.NET Core – висока продуктивність, масштабованість, безпека.

Сучасний адаптивний інтерфейс – темна тема, зручні картки фільмів, підтримка мобільних пристроїв.

Персоналізовані списки перегляду та оцінки – формування індивідуальної стрічки рекомендацій.

4. Вимоги до забезпечення

4.1. Управління користувачами

До вимог управління користувачами входить:

— реєстрація за електронною поштою та паролем;

— валідація даних;

— авторизація з використанням хешування паролів.

4.2. Управління емоційним станом

Надання користувачеві можливості вибрати стандартні емоційні стани серед всього набору. Збереження обраного стану в БД для формування рекомендацій.

4.3. Формування рекомендацій

Алгоритм підбору фільмів на основі:

— обраного емоційного стану;

— жанрових уподобань;

— популярності та рейтингу.

Відображення результатів у вигляді карток.

4.4. Робота з каталогом кінотворів

Робота з каталогом кінотворів у системі «MOVI.FLOW-ER» базується на інтеграції із зовнішнім сервісом TMDb API, який виступає основним джерелом актуальної та структурованої інформації про фільми, серіали, мультфільми та аніме. Отримання даних відбувається шляхом надсилання стандартизованих HTTP-запитів до відповідних ендпоінтів API; відповіді, отримані у форматі JSON, десеріалізуються у внутрішні моделі даних. З метою оптимізації продуктивності та зменшення навантаження на зовнішнє джерело, система передбачає зберігання у локальній базі даних мінімального, але достатнього набору полів: зовнішній ідентифікатор TMDb ID, назва кінотвору, опис, жанрова приналежність, рік випуску, рейтинг та URL-адреса постера.

Оновлення локальної копії може виконуватися за розкладом, автоматично при отриманні нових даних у відповідь на пошукові запити користувачів, або в ручному режимі за ініціативою адміністратора. Додатково система забезпечує відображення детальної сторінки кожного кінотвору, на якій, окрім базових атрибутів, може бути представлена розширена інформація: список акторів, режисер, тривалість, трейлер, а також персоналізовані елементи. Такий підхід дозволяє поєднати актуальність даних із зовнішнього джерела та швидкодію, досягнуту завдяки роботі з локальним сховищем.

4.5. Список перегляду та оцінки

Користувачеві надається можливість додавати кінотвори до власного персоналізованого списку «Дивитися пізніше», а також видаляти їх звідти. Оцінювання переглянутих фільмів здійснюється за шкалою, визначеною користувацькими вподобаннями. Уся історія виставлених оцінок зберігається в системі та використовується для подальшого вдосконалення рекомендаційних алгоритмів, що підвищує релевантність пропозицій.

4.6. Пошук та фільтрація

Система забезпечує можливість пошуку кінотворів за назвою з частковим

збігом. Для зручності відбору контенту реалізовано фільтрацію за жанром, роком випуску та рейтингом. Крім того, передбачено сортування результатів за популярністю, рейтингом або датою виходу, що дозволяє користувачеві налаштовувати відображення каталогу.

4.7. Збереження та визначення потоків даних у БД

Система використовує реляційну базу даних, яка містить такі основні сутності:

— «Користувач»;

— «Кінотвір»;

— «Оцінка»;

— «Список перегляду»;

— «Емоційний стан»;

— «Рекомендація.

Логічні зв’язки між сутностями побудовано за принципом «один-до-багатьох»:

— один користувач може мати багато оцінок, багато записів у списку перегляду та багато отриманих рекомендацій;

— один кінотвір може отримувати багато оцінок від різних користувачів;

— один емоційний стан може використовуватися для формування багатьох рекомендацій.

Така структура забезпечує цілісність даних та ефективну роботу системи.

4.8. Характеристика користувачів системи

Незареєстрований користувач може переглядати каталог, шукати фільми, але не отримує персоналізованих рекомендацій, не може оцінювати чи додавати до списку.

Зареєстрований користувач має доступ до всіх функцій, а саме вибір настрою, рекомендації, оцінки, список перегляду, історія.

5. Нефункціональні вимоги

5.1. Вимоги до надійності та відмовостійкості

До вимог відмовостійкості вебзастосунку «MOVI.FLOW-ER» належить забезпечення високого рівня безперебійної роботи. Система повинна підтримувати

автоматичне відновлення після тимчасових збоїв без втрати даних та без необхідності ручного втручання. Для захисту інформації від непередбачуваних апаратних або програмних відмов передбачається регулярне резервне копіювання бази даних за встановленим графіком. Крім того, серверна частина має реалізовувати механізми обробки виняткових ситуацій із детальним логуванням подій, а саме усі помилки, попередження та критичні збої фіксуються у спеціалізованих журналах для подальшого аналізу та усунення. Логування дозволяє адміністратору відстежувати джерело помилок, оцінювати частоту збоїв і вживати профілактичних заходів для підвищення загальної стабільності функціонування системи.

5.1.1. Системні вимоги до серверу

Програмне забезпечення	Версія / Характеристика
Операційна система	Windows Server 2019/2022 або Linux (Ubuntu 20.04+)
Вебсервер	IIS (Windows) або Kestrel + Nginx (Linux)
СКБД	Microsoft SQL Server 2019+
Платформа .NET	.NET 6/8 Runtime
Поштовий сервер	SMTP (для сповіщень, опціонально)

Мінімальні апаратні вимоги до сервера: 2 vCPU, 4 GB RAM, 50 GB SSD.

5.1.2. Вимоги до структури та функціонування

Архітектура вебзастосунку базується на клієнт-серверній моделі з використанням шаблону MVC (Model-View-Controller). Взаємодія між клієнтською та серверною частинами здійснюється за допомогою протоколів HTTP/HTTPS, причому для окремих запитів реалізовано REST API. Форматом обміну даними визначено JSON. Автентифікація користувачів забезпечується через механізм токенів або сесій, що гарантує безпечний доступ до функціоналу системи.

5.2. Вимоги до інтерфейсу користувача

Інтерфейс користувача має відповідати сучасним вимогам вебдизайну, зокрема використовувати темну тему та мінімалістичний стиль. Навігація має бути інтуїтивно зрозумілою та включати головну сторінку, сторінку рекомендацій, профіль користувача та список перегляду. Візуалізація емоційних станів реалізується за допомогою іконок або емодзі. Картки кінотворів обов'язково містять постер, назву, рік випуску та середній рейтинг.

5.2.1. Сторінки вебзастосунку

До складу вебзастосунку входять такі основні сторінки:

— головна сторінка, на якій користувач може обрати настрій та переглянути блок рекомендацій;

— сторінка каталогу з можливістю пошуку, фільтрації та відображення сітки фільмів;

— детальна сторінка кінотвору, що містить опис, постер, рейтинг, а також кнопки для оцінювання та додавання до списку;

— сторінка профілю користувача для редагування персональних даних та перегляду історії оцінок;

— сторінка списку перегляду;

— сторінки входу та реєстрації.

5.3. Вимоги до розвитку та модернізації системи

Модульна архітектура системи дозволяє додавати нові алгоритми рекомендацій, зокрема колаборативну фільтрацію або методи машинного навчання, без необхідності змінювати ядро застосунку. Передбачена можливість інтеграції з іншими зовнішніми API для розширення функціоналу. Також адміністратор системи має змогу розширювати перелік емоційних станів через адміністративну панель.

5.4. Вимоги до стандартизації та уніфікації

Система повинна відповідати стандартам W3C для HTML5 та CSS3. Забезпечується кросбраузерна сумісність з останніми версіями браузерів Google

Chrome, Mozilla Firefox, Microsoft Edge та Safari. Для побудови програмного інтерфейсу використовуються принципи REST, що гарантує уніфікований підхід до взаємодії компонентів.

5.5. Вимоги до інформаційного забезпечення

Захист персональних даних користувачів реалізується відповідно до вимог Закону України «Про захист персональних даних». Цілісність даних забезпечується механізмами каскадного видалення та використанням зовнішніх ключів у базі даних. Актуальність інформації про кінотвори підтримується шляхом періодичного опитування зовнішнього API TMDb.

6. Порядок контролю та приймання системи

6.1. Приймальні випробування

Приймальні випробування вебзастосунку проводяться робочою групою, до складу якої входять виконавець та представник замовника. У ході випробувань перевіряється відповідність системи функціональним вимогам (згідно з п. 4.1) та нефункціональним вимогам (згідно з п. 5). Тестування здійснюється за методиками, розробленими виконавцем, що забезпечує системний підхід до оцінювання якості програмного продукту.

6.2. План модернізації

У перспективі розвитку системи передбачається впровадження колаборативної фільтрації на основі оцінок користувачів для підвищення точності рекомендацій. Планується додавання соціального функціоналу, зокрема можливості коментування кінотворів та створення спільних списків перегляду. Окремим напрямком модернізації є розробка мобільного додатку на основі вебверсії, а також інтеграція з популярними стрімінговими сервісами для забезпечення прямого переходу до перегляду контенту.

6.3. Тестування та контроль якості

Контроль якості вебзастосунку охоплює декілька рівнів тестування. Модульне тестування виконується за допомогою фреймворків xUnit та Moq та спрямоване на

перевірку окремих сервісів, зокрема логіки формування рекомендацій та роботи з базою даних. Інтеграційне тестування забезпечує перевірку коректності взаємодії з зовнішнім API TMDb, а також роботи контролерів. Функціональне тестування передбачає виконання основних сценаріїв використання системи, таких як реєстрація користувача, вибір емоційного стану, отримання рекомендацій. Тестування інтерфейсу охоплює перевірку адаптивності дизайну, кросбраузерної сумісності та валідації HTML/CSS-коду.

7. Управління проєктом

Управління проєктом здійснюється з використанням гнучкої методології Agile у форматі Scrum. Етапи виконання робіт визначаються календарним планом кваліфікаційної роботи, що передбачає послідовну реалізацію завдань від аналізу предметної області до тестування та впровадження. Контроль виконання забезпечується шляхом подання щотижневих звітів керівнику роботи. Управління ризиками включає регулярний аналіз можливих збоїв та формування резервних копій даних для мінімізації наслідків критичних помилок.

7.1. Порядок впровадження

Впровадження вебзастосунку передбачає декілька послідовних етапів. На першому етапі виконується розгортання системи на тестовому хостингу freeasphosting.net для проведення попереднього тестування. Далі здійснюється пілотна експлуатація за участю обмеженої групи користувачів (5–10 осіб) з метою збору відгуків та виявлення недоліків. За результатами пілотування проводиться коригування функціоналу, після чого система розгортається на основному хостингу та стає доступною для публічного використання.

7.1.1. Підтримка та обслуговування

Після офіційної здачі проєкту виконавець забезпечує технічну підтримку системи протягом трьох місяців. Підтримка включає консультування адміністратора з питань експлуатації, допомогу у вирішенні проблем, пов'язаних із роботою серверної частини та бази даних, а також моніторинг стабільності функціонування основних модулів. Оновлення контенту здійснюється на регулярній основі шляхом

автоматичної синхронізації з зовнішнім API TMDb, яка відбувається за встановленим розкладом (наприклад, один раз на добу). За потреби адміністратор може ініціювати ручне оновлення даних для отримання актуальної інформації про нові кінотвори.

Виправлення критичних помилок, виявлених після релізу, виконується протягом перших двох тижнів; цей термін охоплює усунення недоліків, що впливають на працездатність системи, цілісність даних або безпеку користувачів. Менш суттєві помилки та пропозиції щодо покращення можуть бути враховані в наступних оновленнях після завершення гарантійного періоду.

7.1.2. Заключні рекомендації

Система «MOVI.FLOW-ER» є повноцінним рекомендаційним сервісом, який дозволяє користувачам економити час на пошуку кінотворів завдяки персоналізації за емоційним станом. Система продемонструвала коректну роботу основних функцій, стабільність у середовищі тестового хостингу та відповідність поставленим вимогам. Для підвищення конкурентоздатності та подальшого розвитку рекомендується вдосконалення алгоритмів рекомендацій, зокрема впровадження методів колаборативної фільтрації та машинного навчання. Реалізація зазначених напрямків дозволить перетворити систему на багатофункціональний медійний хаб із високим рівнем залученості користувачів.

ДОДАТОК В

(обов'язковий)

ПРОГРАМНИЙ КОД ОСНОВНИХ МОДУЛІВ

В.1 Код AccountController:

```
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using MOVIFLOWER.Models.Entities;
using MOVIFLOWER.Models.ViewModels;
using System.Security.Claims;

namespace MOVIFLOWER.Controllers
{
    public class AccountController : Controller
    {
        private readonly UserManager<ApplicationUser> _userManager;
        private readonly SignInManager<ApplicationUser> _signInManager;

        public AccountController(
            UserManager<ApplicationUser> userManager,
            SignInManager<ApplicationUser> signInManager)
        {
            _userManager = userManager;
            _signInManager = signInManager;
        }

        // ──────────────────────────────────────
        // REGISTER
        // ──────────────────────────────────────

        public IActionResult Register()
        {
            if (User.Identity?.IsAuthenticated == true)
                return RedirectToAction("Index", "Home");

            return View();
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Register(RegisterViewModel model)
        {
            if (!ModelState.IsValid)
                return View(model);

            var existingByEmail = await _userManager.FindByEmailAsync(model.Email);
            if (existingByEmail != null)
            {
                ModelState.AddModelError("Email", "This email is already
registered");
                return View(model);
            }

            var existingByName = await
_userManager.FindByNameAsync(model.DisplayName);
            if (existingByName != null)
            {
```

```
ModelState.AddModelError("DisplayName", "This username is already taken");
                return View(model);
            }

            var user = new ApplicationUser
            {
                UserName = model.DisplayName,
                Email = model.Email,
                DisplayName = model.DisplayName,
                CreatedAt = DateTime.UtcNow
            };

            var result = await _userManager.CreateAsync(user, model.Password);

            if (result.Succeeded)
            {
                await _signInManager.SignInAsync(user, isPersistent: false);
                return RedirectToAction("Index", "Home");
            }

            foreach (var error in result.Errors)
                ModelState.AddModelError("", error.Description);

            return View(model);
        }

        // ──────────────────────────────────────
        // LOGIN
        // ──────────────────────────────────────

        public IActionResult Login()
        {
            if (User.Identity?.IsAuthenticated == true)
                return RedirectToAction("Index", "Home");

            return View();
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Login(LoginViewModel model)
        {
            if (!ModelState.IsValid)
                return View(model);

            ApplicationUser? user = null;

            if (model.Login.Contains('@'))
                user = await _userManager.FindByEmailAsync(model.Login);

            user ??= await _userManager.FindByNameAsync(model.Login);

            if (user == null)
            {
                ModelState.AddModelError("", "Invalid login or password");
                return View(model);
            }

            var result = await _signInManager.PasswordSignInAsync(
                user,
                model.Password,
                model.RememberMe,
                lockoutOnFailure: true);

            if (result.Succeeded)
                return RedirectToAction("Index", "Home");
```

```
 if (result.IsLockedOut)
            {
ModelState.AddModelError("", "Account locked. Try again in a few minutes.");
                return View(model);
            }

            ModelState.AddModelError("", "Invalid login or password");
            return View(model);
        }

        // ─────────────────────────────────────
        // LOGOUT
        // ─────────────────────────────────────

        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Logout()
        {
            await _signInManager.SignOutAsync();
            return RedirectToAction("Index", "Home");
        }


        [HttpGet]
        public IActionResult ExternalLogin(string provider, string returnUrl = null)
        {
            var redirectUrl = Url.Action("ExternalLoginCallback", "Account",
                new { ReturnUrl = returnUrl });

            var properties =
_signInManager.ConfigureExternalAuthenticationProperties(provider, redirectUrl);

            return Challenge(properties, provider);
        }

        public async Task<IActionResult> ExternalLoginCallback(string returnUrl =
null)
        {
            var info = await _signInManager.GetExternalLoginInfoAsync();

            if (info == null)
                return RedirectToAction("Login");

            var signInResult = await _signInManager.ExternalLoginSignInAsync(
                info.LoginProvider,
                info.ProviderKey,
                isPersistent: false);

            if (signInResult.Succeeded)
                return RedirectToLocal(returnUrl);


            var email =
info.Principal.FindFirstValue(System.Security.Claims.ClaimTypes.Email);

            var user = new ApplicationUser
            {
                UserName = email,
                Email = email,
                DisplayName = email
            };

            var result = await _userManager.CreateAsync(user);

            if (result.Succeeded)
            {
                await _userManager.AddLoginAsync(user, info);
```

```
  await _signInManager.SignInAsync(user, false);
                return RedirectToLocal(returnUrl);
            }

return RedirectToAction("Login");
        }

        private IActionResult RedirectToLocal(string returnUrl)
        {
            if (Url.IsLocalUrl(returnUrl))
                return Redirect(returnUrl);

            return RedirectToAction("Index", "Home");
        }
    }
}
```

B.2 Код HomeController:

```
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using MOVIFLOWER.Data;
using MOVIFLOWER.Models.Entities;
using MOVIFLOWER.Models.ViewModels;
using MOVIFLOWER.Services;
using System.Diagnostics;

namespace MOVIFLOWER.Controllers
{
    public class HomeController : Controller
    {
        private readonly TmdbService _tmdbService;
        private readonly ApplicationDbContext _context;
        private readonly UserManager<ApplicationUser> _userManager;
        private readonly MoodPulseService _pulseService;

        public HomeController(
     TmdbService tmdbService,
     ApplicationDbContext context,
     UserManager<ApplicationUser> userManager,
     MoodPulseService pulseService)
        {
            _tmdbService = tmdbService;
            _context = context;
            _userManager = userManager;
            _pulseService = pulseService;
        }

        public async Task<IActionResult> Index(
            string? mood = null, int intensity = 5, bool expand = false,
            string? sort = "popular", int? yearFrom = null, int? yearTo = null,
            string? type = "all", int page = 1)
        {
            var result = await _tmdbService.GetDiscoverMoviesAsync(mood, sort ??
"popular", yearFrom, yearTo, type ?? "all", page);
            ViewBag.CurrentMood = mood;
            ViewBag.Intensity = intensity;
            ViewBag.Sort = sort;
            ViewBag.YearFrom = yearFrom;
            ViewBag.YearTo = yearTo;
            ViewBag.Type = type;
            ViewBag.Page = page;
```

```
  ViewBag.TotalPages = result.TotalPages;
            ViewBag.Expand = expand;
            ViewBag.Pulse = await _pulseService.GetPulseAsync();
            return View(result.Movies);
        }

[HttpGet]
        public IActionResult Quiz() => View();

        [HttpPost]
        public async Task<IActionResult> QuizResult(MoodQuizViewModel model)
        {
            var mood = ResolveMood(model);
            var type = model.Type ?? "all";

            var (movies, _) = await _tmdbService.GetDiscoverMoviesAsync(mood,
"popular", null, null, type, page: 1);


            var watchedIds = new HashSet<int>();
            if (User.Identity?.IsAuthenticated == true)
            {
                var userId = _userManager.GetUserId(User);
                var ids = await _context.UserMovieInteractions
                    .Where(x => x.UserId == userId && x.IsWatched)
                    .Select(x => x.TmdbId)
                    .ToListAsync();
                watchedIds = ids.ToHashSet();
            }

            var recommendations = movies
                .Where(m => !watchedIds.Contains(m.id))
                .Take(3)
                .ToList();

            ViewBag.Mood = mood;
            ViewBag.QuizType = type;
            ViewBag.Vibe = model.Vibe;
            ViewBag.Energy = model.EnergyLevel;

            return View(recommendations);
        }

        private static string ResolveMood(MoodQuizViewModel m) =>
            (m.Vibe, m.EnergyLevel) switch
            {
                ("romantic", _) => "romantic",
                ("dark", _) => "dark",
                ("inspired", "high") => "motivated",
                ("inspired", _) => "inspired",
                ("fun", "high") => "adrenaline",
                ("fun", _) => "light",
                ("nostalgic", _) => "nostalgic",
                ("mysterious", _) => "mysterious",
                (_, "high") => "adrenaline",
                (_, "calm") => "calm",
                _ => "escape"
            };

        public IActionResult Privacy() => View();

        [ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore =
true)]
        public IActionResult Error()
            => View(new ErrorViewModel { RequestId = Activity.Current?.Id ??
HttpContext.TraceIdentifier });
    }
```

```
    public class ErrorViewModel
    {
        public string? RequestId { get; set; }
        public bool ShowRequestId => !string.IsNullOrEmpty(RequestId);
    }
}
```

B.3 Код MovieController:

```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using MOVIFLOWER.Data;
using MOVIFLOWER.Models.DTOs;
using MOVIFLOWER.Models.Entities;
using MOVIFLOWER.Models.TMDB;
using MOVIFLOWER.Services;

using System.Security.Claims;

namespace MOVIFLOWER.Controllers
{
    public class MoviesController : Controller
    {
        private readonly ApplicationDbContext _context;
        private readonly TmdbService _tmdbService;

        public MoviesController(ApplicationDbContext context, TmdbService tmdb-
Service)
        {
            _context = context;
            _tmdbService = tmdbService;
        }

        // SEARCH
        public async Task<IActionResult> Search(string query)
        {
            if (string.IsNullOrEmpty(query))
                return RedirectToAction("Index", "Home");
            var movies = await _tmdbService.SearchMoviesAsync(query);
            ViewBag.CurrentMood = $"Search: {query}";
            return View("~/Views/Home/Index.cshtml", movies);
        }



        // DETAILS
        public async Task<IActionResult> Details(int tmdbId, string? mediaType)
        {

            var type = string.Equals(mediaType, "tv", StringComparison.OrdinalIgnore-
Case)
                ? "tv"
                : "movie";

            var movie = await _tmdbService.GetMovieDetailsAsync(tmdbId, type);
            if (movie == null) return NotFound();


            var statsTask = _tmdbService.GetMovieStatsAsync(_context, tmdbId);
            var castTask = _tmdbService.GetMovieCastAsync(tmdbId, type);
            var similarTask = _tmdbService.GetSimilarMoviesAsync(tmdbId, type);

            await Task.WhenAll(statsTask, castTask, similarTask);
```

```
            var (likes, dislikes) = await statsTask;

            UserMovieInteraction? interaction = null;
            bool hasReview = false;

            if (User.Identity?.IsAuthenticated == true)
            {
          var userId = User.FindFirstValue(ClaimTypes.NameIdentifier);

                interaction = await _context.UserMovieInteractions
                    .FirstOrDefaultAsync(x => x.UserId == userId && x.TmdbId == tmd-
bId);

                hasReview = await _context.MovieReviews
                    .AnyAsync(r => r.UserId == userId && r.TmdbId == tmdbId);
            }

            var reviews = await _context.MovieReviews
                .Where(r => r.TmdbId == tmdbId)
                .OrderByDescending(r => r.CreatedAt)
                .ToListAsync();

            var dto = new MovieWithStatsDto
            {
                Movie = movie,
                LikesCount = likes,
                DislikesCount = dislikes,
                IsLikedByUser = interaction?.IsLiked ?? false,
                IsDislikedByUser = interaction?.IsDisliked ?? false,
                IsWatchedByUser = interaction?.IsWatched ?? false,
                IsWatchLaterByUser = interaction?.IsWatchLater ?? false,
                HasReviewedByUser = hasReview,
                Cast = await castTask,
                Reviews = reviews,
                SimilarMovies = await similarTask
            };

            ViewBag.MediaType = type;
            return View(dto);
        }

        // ————————————————————————————————
        // ADD REVIEW
        // ————————————————————————————————

        [HttpPost]
        [Authorize]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> AddReview(int tmdbId, string mediaType,
string emoji, string? text)
        {
            var userId = User.FindFirstValue(ClaimTypes.NameIdentifier);
            var userName = User.Identity?.Name ?? "User";

            // Тільки один відгук
            var already = await _context.MovieReviews
                .AnyAsync(r => r.UserId == userId && r.TmdbId == tmdbId);

            if (!already)
            {
                _context.MovieReviews.Add(new MovieReview
                {
                    UserId = userId!,
                    UserName = userName,
                    TmdbId = tmdbId,
                    Emoji = emoji,
                    Text = text?.Trim() ?? "",
```

```
        CreatedAt = DateTime.UtcNow
                });

                await _context.SaveChangesAsync();
            }

            return RedirectToAction("Details", new { tmdbId, mediaType });
        }

[HttpPost, Authorize, ValidateAntiForgeryToken]
        public async Task<IActionResult> Like(int tmdbId, string? mediaType =
"movie", string? returnUrl = null)
        {
            var userId = User.FindFirstValue(ClaimTypes.NameIdentifier);
            var interaction = await _context.UserMovieInteractions
                .FirstOrDefaultAsync(x => x.UserId == userId && x.TmdbId == tmdbId);

            if (interaction == null)
            {
                _context.UserMovieInteractions.Add(new UserMovieInteraction
                {
                    UserId = userId!,
                    TmdbId = tmdbId,
                    MediaType = mediaType ?? "movie",
                    IsLiked = true,
                    IsDisliked = false,
                    CreatedAt = DateTime.UtcNow
                });
            }
            else
            {
                interaction.IsLiked = !interaction.IsLiked;
                interaction.MediaType = mediaType ?? "movie";
                if (interaction.IsLiked) interaction.IsDisliked = false;
            }

            await _context.SaveChangesAsync();
            return !string.IsNullOrEmpty(returnUrl) && Url.IsLocalUrl(returnUrl)
                ? Redirect(returnUrl)
                : RedirectToAction("Details", new { tmdbId, mediaType });
        }


        // DISLIKE (toggle, взаємовиключення з Like)
        [HttpPost, Authorize, ValidateAntiForgeryToken]
        public async Task<IActionResult> Dislike(int tmdbId, string? mediaType =
"movie", string? returnUrl = null)
        {
            var userId = User.FindFirstValue(ClaimTypes.NameIdentifier);
            var interaction = await _context.UserMovieInteractions
                .FirstOrDefaultAsync(x => x.UserId == userId && x.TmdbId == tmdbId);

            if (interaction == null)
            {
                _context.UserMovieInteractions.Add(new UserMovieInteraction
                {
                    UserId = userId!,
                    TmdbId = tmdbId,
                    MediaType = mediaType ?? "movie",
                    IsDisliked = true,
                    IsLiked = false,
                    CreatedAt = DateTime.UtcNow
                });
            }
            else
            {
                interaction.IsDisliked = !interaction.IsDisliked;
```

```
   interaction.MediaType = mediaType ?? "movie";
                if (interaction.IsDisliked) interaction.IsLiked = false;
            }

            await _context.SaveChangesAsync();
            return !string.IsNullOrEmpty(returnUrl) && Url.IsLocalUrl(returnUrl)
                ? Redirect(returnUrl)
                : RedirectToAction("Details", new { tmdbId, mediaType });
        }

        // MARK AS WATCHED (toggle, взаємовиключення з WatchLater)
[HttpPost, Authorize, ValidateAntiForgeryToken]
        public async Task<IActionResult> MarkAsWatched(int tmdbId, string? mediaType
= "movie")
        {
            var userId = User.FindFirstValue(ClaimTypes.NameIdentifier);
            var interaction = await _context.UserMovieInteractions
                .FirstOrDefaultAsync(x => x.UserId == userId && x.TmdbId == tmdbId);

            if (interaction == null)
            {
                _context.UserMovieInteractions.Add(new UserMovieInteraction
                {
                    UserId = userId!,
                    TmdbId = tmdbId,
                    MediaType = mediaType ?? "movie",
                    IsWatched = true,
                    IsWatchLater = false,
                    CreatedAt = DateTime.UtcNow
                });
            }
            else
            {
                interaction.IsWatched = !interaction.IsWatched;
                interaction.MediaType = mediaType ?? "movie";
                if (interaction.IsWatched) interaction.IsWatchLater = false;
            }

            await _context.SaveChangesAsync();
            return RedirectToAction("Details", new { tmdbId, mediaType });
        }

        // TOGGLE WATCH LATER (взаємовиключення з Watched)
        [HttpPost, Authorize, ValidateAntiForgeryToken]
        public async Task<IActionResult> ToggleWatchLater(int tmdbId, string? medi-
aType = "movie")
        {
            var userId = User.FindFirstValue(ClaimTypes.NameIdentifier);
            var interaction = await _context.UserMovieInteractions
                .FirstOrDefaultAsync(x => x.UserId == userId && x.TmdbId == tmdbId);

            if (interaction == null)
            {
                _context.UserMovieInteractions.Add(new UserMovieInteraction
                {
                    UserId = userId!,
                    TmdbId = tmdbId,
                    MediaType = mediaType ?? "movie",
                    IsWatchLater = true,
                    IsWatched = false,
                    CreatedAt = DateTime.UtcNow
                });
            }
            else
            {
                interaction.IsWatchLater = !interaction.IsWatchLater;
                interaction.MediaType = mediaType ?? "movie";
```

```
   if (interaction.IsWatchLater) interaction.IsWatched = false;
            }

            await _context.SaveChangesAsync();
            return RedirectToAction("Details", new { tmdbId, mediaType });
        }

        // LIST PAGES
        [Authorize]
        public async Task<IActionResult> Watched()
        {
            var userId = User.FindFirstValue(ClaimTypes.NameIdentifier);
            var ids = await _context.UserMovieInteractions
.Where(x => x.UserId == userId && x.IsWatched).Select(x => x.TmdbId).ToListAsync();
            var movies = (await Task.WhenAll(ids.Select(id => _tmdbService.GetMovie-
DetailsAsync(id, "movie"))))
                .Where(m => m != null).ToList();
            return View(movies);
        }

        [Authorize]
        public async Task<IActionResult> Liked()
        {
            var userId = User.FindFirstValue(ClaimTypes.NameIdentifier);
            var ids = await _context.UserMovieInteractions
                .Where(x => x.UserId == userId && x.IsLiked).Select(x => x.Tmd-
bId).ToListAsync();
            var movies = (await Task.WhenAll(ids.Select(id => _tmdbService.GetMovie-
DetailsAsync(id, "movie"))))
                .Where(m => m != null).ToList();
            return View(movies);
        }

        [Authorize]
        public async Task<IActionResult> Disliked()
        {
            var userId = User.FindFirstValue(ClaimTypes.NameIdentifier);
            var ids = await _context.UserMovieInteractions
                .Where(x => x.UserId == userId && x.IsDisliked).Select(x => x.Tmd-
bId).ToListAsync();
            var movies = (await Task.WhenAll(ids.Select(id => _tmdbService.GetMovie-
DetailsAsync(id, "movie" ))))
                .Where(m => m != null).ToList();
            return View(movies);
        }

        [Authorize]
        public async Task<IActionResult> WatchLater()
        {
            var userId = User.FindFirstValue(ClaimTypes.NameIdentifier);

            var items = await _context.UserMovieInteractions
                .Where(x => x.UserId == userId && x.IsWatchLater)
                .Select(x => new { x.TmdbId, x.MediaType })
                .ToListAsync();

            var movies = await Task.WhenAll(
                items.Select(i =>
                    _tmdbService.GetMovieDetailsAsync(
                        i.TmdbId,
                        i.MediaType ?? "movie"))
            );

            return View(movies.Where(m => m != null).ToList());
        }

        // ADMIN
```

```
[Authorize(Roles = "Admin")]
        public async Task<IActionResult> Index() => View(await _context.Mov-
ies.ToListAsync());

        [Authorize(Roles = "Admin")]
        public IActionResult Create() => View();

        [HttpPost, Authorize(Roles = "Admin"), ValidateAntiForgeryToken]
        public async Task<IActionResult> Create(Movie movie)
        {
            if (!ModelState.IsValid) return View(movie);
            movie.CreatedAt = DateTime.UtcNow;
            _context.Add(movie);
            await _context.SaveChangesAsync();
            return RedirectToAction(nameof(Index));
}

        [Authorize(Roles = "Admin")]
        public async Task<IActionResult> Edit(int? id)
        {
            if (id == null) return NotFound();
            var movie = await _context.Movies.FindAsync(id);
            return movie == null ? NotFound() : View(movie);
        }

        [HttpPost, Authorize(Roles = "Admin"), ValidateAntiForgeryToken]
        public async Task<IActionResult> Edit(int id, Movie movie)
        {
            if (id != movie.Id) return NotFound();
            if (!ModelState.IsValid) return View(movie);
            try { _context.Update(movie); await _context.SaveChangesAsync(); }
            catch (DbUpdateConcurrencyException)
            { if (!_context.Movies.Any(e => e.Id == id)) return NotFound(); else
throw; }
            return RedirectToAction(nameof(Index));
        }

        [Authorize(Roles = "Admin")]
        public async Task<IActionResult> Delete(int? id)
        {
            if (id == null) return NotFound();
            var movie = await _context.Movies.FirstOrDefaultAsync(m => m.Id == id);
            return movie == null ? NotFound() : View(movie);
        }

        [HttpPost, ActionName("Delete"), Authorize(Roles = "Admin"), ValidateAntiFor-
geryToken]
        public async Task<IActionResult> DeleteConfirmed(int id)
        {
            var movie = await _context.Movies.FindAsync(id);
            if (movie != null) { _context.Movies.Remove(movie); await _con-
text.SaveChangesAsync(); }
            return RedirectToAction(nameof(Index));
        }

        public async Task<IActionResult> Popular()
            => View(await _tmdbService.GetPopularMoviesAsync());

        public async Task<IActionResult> ByKeyword(string keyword)
        {
            if (string.IsNullOrEmpty(keyword)) return View(new List<TmdbMovieDto>());
            return View("Popular", await _tmdbService.GetMoviesByKeywordAsync(key-
word));
        }

        public async Task<IActionResult> SyncPopular()
        {
```

```
  await _tmdbService.SavePopularToDatabase(_context);
            return RedirectToAction("Index");
        }



        public async Task<IActionResult> Recommendations(string mood)
        {
            var movies = _context.Movies.AsQueryable();

            if (!string.IsNullOrEmpty(mood))
            {
                switch (mood.ToLower())
                {
                    case "happy":
                        movies = movies.Where(m =>
m.Genre.Contains("Comedy") ||
                            m.Genre.Contains("Family"));
                        break;

                    case "sad":
                        movies = movies.Where(m =>
                            m.Genre.Contains("Drama") ||
                            m.Genre.Contains("Romance"));
                        break;

                    case "excited":
                        movies = movies.Where(m =>
                            m.Genre.Contains("Action") ||
                            m.Genre.Contains("Adventure"));
                        break;

                    case "scared":
                        movies = movies.Where(m =>
                            m.Genre.Contains("Horror") ||
                            m.Genre.Contains("Thriller"));
                        break;
                }
            }

            return View(await movies.ToListAsync());
        }


    }
}
```

B.4 Код ProfileController:

```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using MOVIFLOWER.Models.Entities;
using MOVIFLOWER.Models.ViewModels;

namespace MOVIFLOWER.Controllers
{
    [Authorize]
    public class ProfileController : Controller
    {
        private readonly UserManager<ApplicationUser> _userManager;
```

```
  private readonly SignInManager<ApplicationUser> _signInManager;

        public ProfileController(
            UserManager<ApplicationUser> userManager,
            SignInManager<ApplicationUser> signInManager)
        {
            _userManager = userManager;
            _signInManager = signInManager;
        }

        // ──────────────────────────────────────
        // GET /Profile
        // ──────────────────────────────────────

        public async Task<IActionResult> Index()
        {
            var user = await _userManager.GetUserAsync(User);

            if (user == null)
return RedirectToAction("Login", "Account");

            var model = new ProfileViewModel
            {
                UserName = user.UserName ?? "",
                Email = user.Email ?? ""
            };

            return View(model);
        }

        // ──────────────────────────────────────
        // POST /Profile/Update
        // ──────────────────────────────────────

        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Update(ProfileViewModel model)
        {

            if (!string.IsNullOrEmpty(model.NewPassword) &&
string.IsNullOrEmpty(model.CurrentPassword))
                ModelState.AddModelError("CurrentPassword", "Enter your current
password to set a new one");

            if (!ModelState.IsValid)
                return View("Index", model);

            var user = await _userManager.GetUserAsync(User);

            if (user == null)
                return RedirectToAction("Login", "Account");

            var hasErrors = false;

            // ── Username ──────────────────────
            if (user.UserName != model.UserName)
            {

                var existing = await _userManager.FindByNameAsync(model.UserName);
                if (existing != null && existing.Id != user.Id)
                {
                    ModelState.AddModelError("UserName", "This username is already
taken");
                    hasErrors = true;
                }
                else
                {
```

```
 var result = await _userManager.SetUserNameAsync(user, model.UserName);
                    if (!result.Succeeded)
                    {
                        foreach (var e in result.Errors)
                            ModelState.AddModelError("UserName", e.Description);
                        hasErrors = true;
                    }
                }
            }

            // ── Email ──────────────────────────
            if (user.Email != model.Email)
            {

                var existing = await _userManager.FindByEmailAsync(model.Email);
                if (existing != null && existing.Id != user.Id)
                {
                    ModelState.AddModelError("Email", "This email is already in
use");
                    hasErrors = true;
}
                else
                {
                    var result = await _userManager.SetEmailAsync(user, model.Email);
                    if (!result.Succeeded)
                    {
                        foreach (var e in result.Errors)
                            ModelState.AddModelError("Email", e.Description);
                        hasErrors = true;
                    }
                }
            }

            // ── Password ───────────────────────
            if (!string.IsNullOrEmpty(model.NewPassword) &&
!string.IsNullOrEmpty(model.CurrentPassword))
            {

                var passwordCheck = await _userManager.CheckPasswordAsync(user,
model.CurrentPassword);

                if (!passwordCheck)
                {
                    ModelState.AddModelError("CurrentPassword", "Current password is
incorrect");
                    hasErrors = true;
                }
                else
                {
                    var result = await _userManager.ChangePasswordAsync(
                        user,
                        model.CurrentPassword,
                        model.NewPassword);

                    if (!result.Succeeded)
                    {
                        foreach (var e in result.Errors)
                            ModelState.AddModelError("NewPassword", e.Description);
                        hasErrors = true;
                    }
                }
            }

            if (hasErrors)
                return View("Index", model);
```

```
      await _signInManager.RefreshSignInAsync(user);

                TempData["Success"] = "Profile updated successfully";
                return RedirectToAction("Index");
            }
        }
    }
```

B.5 Код RecommendationController:

```
    using Microsoft.AspNetCore.Authorization;
    using Microsoft.AspNetCore.Mvc;
    using MOVIFLOWER.Services;
    using System.Security.Claims;

    namespace MOVIFLOWER.Controllers
{ [Authorize]
    public class RecommendationController : Controller
    {
        private readonly RecommendationService _service;

        public RecommendationController(RecommendationService service)
        {
            _service = service;
        }

        public async Task<IActionResult> Index()
        {
            if (!User.Identity.IsAuthenticated)
                return RedirectToAction("Login", "Account");

            var userId = User.FindFirstValue(ClaimTypes.NameIdentifier);

            var movies = await _service.GetRecommendations(userId);

            return View(movies);
        }
    }
}
```

B.6 Код WatchlistController:

```
    using Microsoft.AspNetCore.Authorization;
    using Microsoft.AspNetCore.Identity;
    using Microsoft.AspNetCore.Mvc;
    using Microsoft.EntityFrameworkCore;
    using MOVIFLOWER.Data;
    using MOVIFLOWER.Models.Entities;


    namespace MOVIFLOWER.Controllers
    {
        [Authorize]
        public class WatchlistController : Controller
        {
            private readonly ApplicationDbContext _context;
            private readonly UserManager<ApplicationUser> _userManager;

            public WatchlistController(
```

```
  ApplicationDbContext context,
            UserManager<ApplicationUser> userManager)
        {
            _context = context;
            _userManager = userManager;
        }

        public async Task<IActionResult> Index()
        {
            var user = await _userManager.GetUserAsync(User);

            var list = await _context.Watchlists
                .Include(w => w.Movie)
                .Where(w => w.UserId == user!.Id)
                .ToListAsync();

            return View(list);
        }
public async Task<IActionResult> Add(int movieId)
        {
            var user = await _userManager.GetUserAsync(User);

            var exists = await _context.Watchlists
                .AnyAsync(w => w.MovieId == movieId && w.UserId == user!.Id);

            if (!exists)
            {
                var item = new Watchlist
                {
                    MovieId = movieId,
                    UserId = user.Id
                };

                _context.Watchlists.Add(item);
                await _context.SaveChangesAsync();
            }

            return RedirectToAction("Index", "Movies");
        }

        public async Task<IActionResult> Remove(int id)
        {
            var item = await _context.Watchlists.FindAsync(id);

            if (item != null)
            {
                _context.Watchlists.Remove(item);
                await _context.SaveChangesAsync();
            }

            return RedirectToAction(nameof(Index));
        }
```

ДОДАТОК Г
(обов’язковий)

КЕРІВНИЦТВО КОРИСТУВАЧА

Загальний вступ

Система підбору фільмів призначена для автоматизованого формування персоналізованих рекомендацій на основі вподобань користувача, переглянутих фільмів, обраних жанрів та рейтингових показників. Вебзастосунок забезпечує зручну навігацію, інтуїтивний інтерфейс та можливість взаємодії як для звичайних користувачів, так і для адміністратора.

Наведене керівництво описує порядок роботи із системою, функціональні можливості кожної сторінки та логіку переходів між розділами.

Після переходу за вебадресою користувач потрапляє на головну сторінку системи підбору фільмів, яка є доступною як для неавторизованих, так і для зареєстрованих користувачів, Рисунок Г.1.

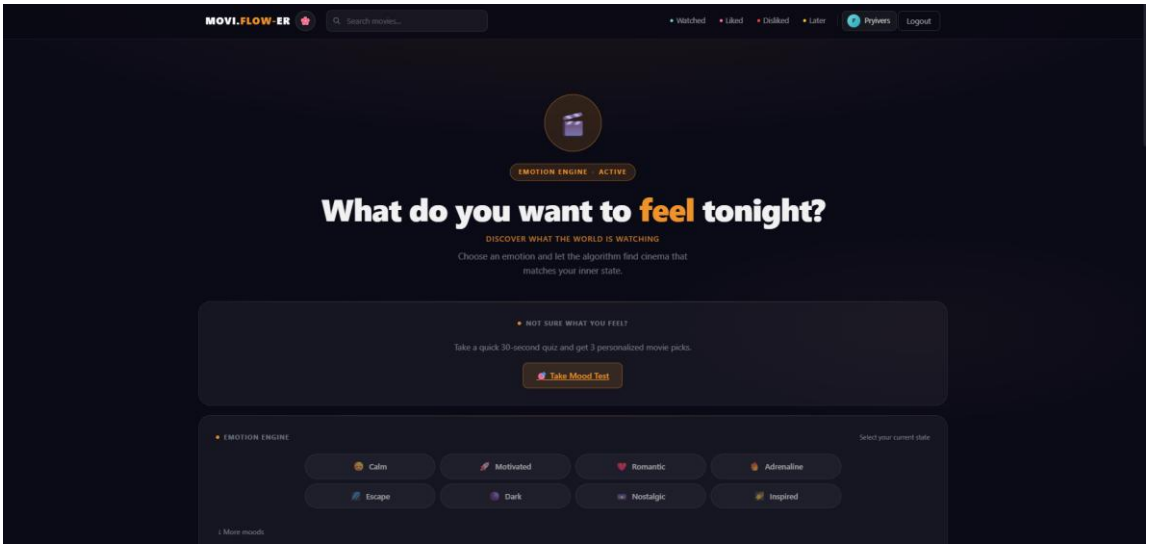


Рисунок Г.1 – Головна сторінка системи

На даній сторінці відображається:

— перелік популярних або нових фільмів;

— короткий опис кожного фільму;

— жанр;

— середній рейтинг;

— кнопка перегляду детальної інформації.

Користувач має можливість здійснювати:

— пошук фільмів за назвою;

— фільтрацію за жанром;

— сортування за рейтингом або датою додавання.

У верхній частині сторінки розміщено панель навігації з такими розділами:

— «Головна»;

— «Мій профіль»;

— «Символ квітки»;

— «Список вподобань»;

— «Список дизлайків»;

— «Список переглянутих фільмів»;

— «Список фільмів, які залишено на потім»;

— «Вхід/Реєстрація».

Головна сторінка виконує роль стартової точки взаємодії із системою та забезпечує швидкий доступ до ключового функціоналу.

Для отримання доступу до персоналізованих рекомендацій користувачу необхідно пройти процедуру авторизації, Рисунок Г.2.

На сторінці «Вхід/Реєстрація» користувач повинен:

— ввести електронну адресу;

— ввести пароль;

— натиснути кнопку «Увійти».

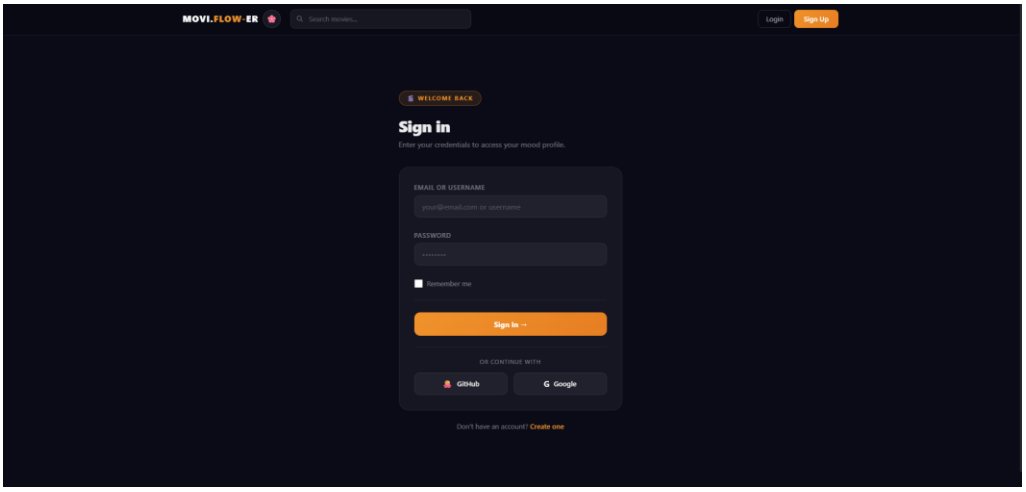


Рисунок Г.2 – Сторінка авторизації

У разі відсутності облікового запису користувач може пройти процедуру реєстрації, заповнивши відповідні поля.

Після успішної автентифікації система перенаправляє користувача до його особистого профілю або на сторінку рекомендацій.

Дана сторінка забезпечує безпечний доступ до персональних даних та реалізує механізм контролю ролей у системі.

Сторінка «Рекомендації» є центральним функціональним модулем системи, Г.3.

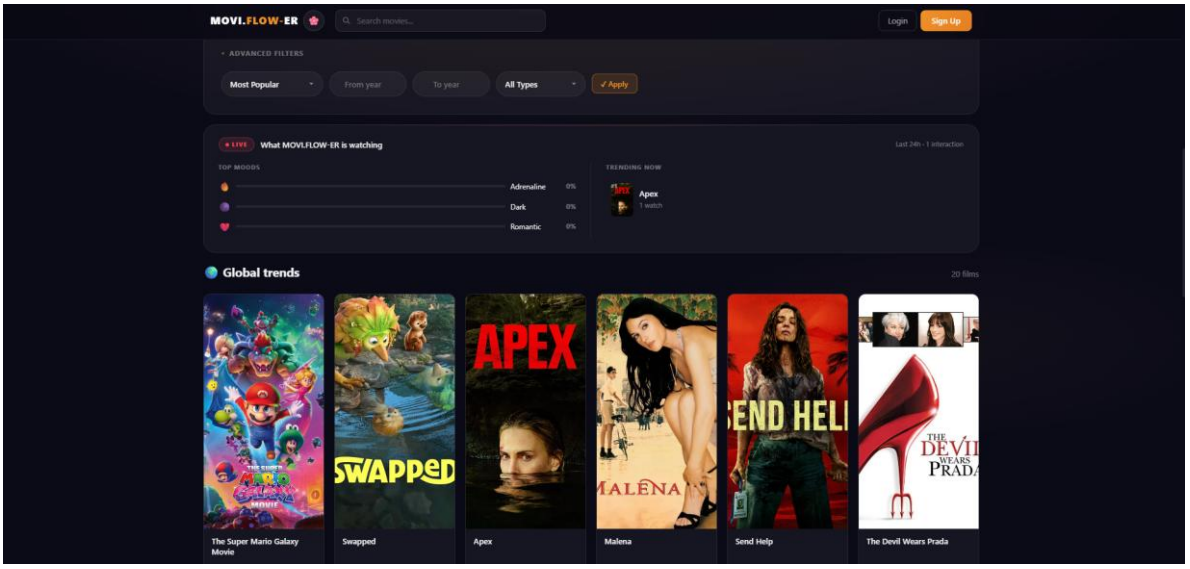


Рисунок Г.3 – Список рекомендацій

На цій сторінці користувач може:

— переглянути персонально підібрані фільми;

— побачити відсортований список за рейтингом;

— отримати рекомендації на основі переглянутих фільмів;

— переглянути жанрово-подібні варіанти.

Алгоритм рекомендацій враховує:

— раніше переглянуті фільми;

— оцінки користувача;

— жанрові вподобання;

— доступні у базі даних фільми;

— фільтрацію вже переглянутих позицій.

Список рекомендацій автоматично оновлюється після додавання нових оцінок або зміни уподобань.

Таким чином, сторінка рекомендацій реалізує основну мету системи — формування релевантного персоналізованого списку фільмів.

Після вибору конкретного фільму користувач переходить на сторінку детального перегляду, Рисунок Г.4.

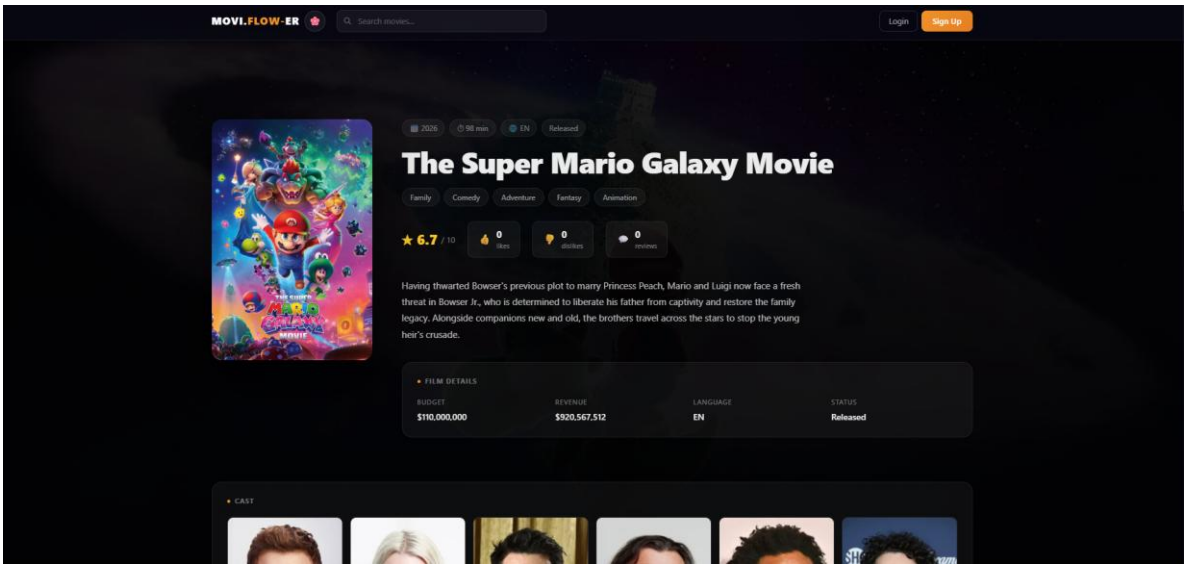


Рисунок Г.4 – Сторінка деталей

На цій сторінці відображається:

— повна назва фільму;

— опис сюжету;

— жанр;

— рейтинг;

— рік випуску;

— кнопка додавання до списку переглянутих;

— можливість виставлення оцінки.

Даний розділ дозволяє користувачу глибше ознайомитися з фільмом перед його переглядом та впливає на подальше формування рекомендацій.

Після авторизації користувач має доступ до особистої сторінки «Профіль», Рисунок Г.5.

У профілі відображається:

— ім’я користувача;

— електронна адреса;

— список переглянутих фільмів;

— виставлені оцінки;

— історія рекомендацій.

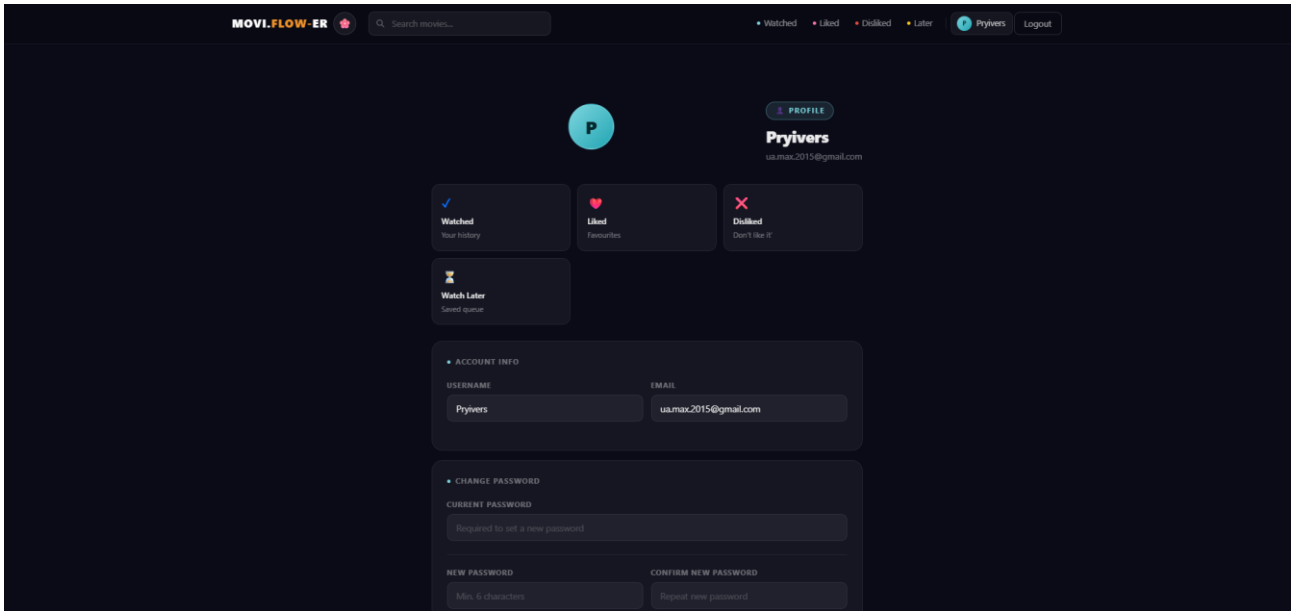


Рисунок Г.5 – Сторінка профілю

Користувач може:

— редагувати особисті дані;

— змінювати пароль;

— переглядати статистику вподобань;

— видаляти фільми зі списку переглянутих.

Профіль забезпечує централізоване управління персональними даними та історією взаємодії із системою.

У верхній частині інтерфейсу розміщено стилізований графічний елемент у вигляді квітки. Даний елемент виконує такі функції:

— формування візуальної ідеї системи;

— створення унікального стилю застосунку;

— підвищення впізнаваності інтерфейсу;

— покращення загального користувацького досвіду.

Квітка є функціональною кнопкою, оскільки може виконувати роль вибору емоції. Її використання підвищує естетичну привабливість вебзастосунку та створює цілісність дизайну.

Сторінка «Тести» призначена для покращення вибору фільму на основі емоційного стану користувача у вигляді короткого тестування, Рисунок Г.6.

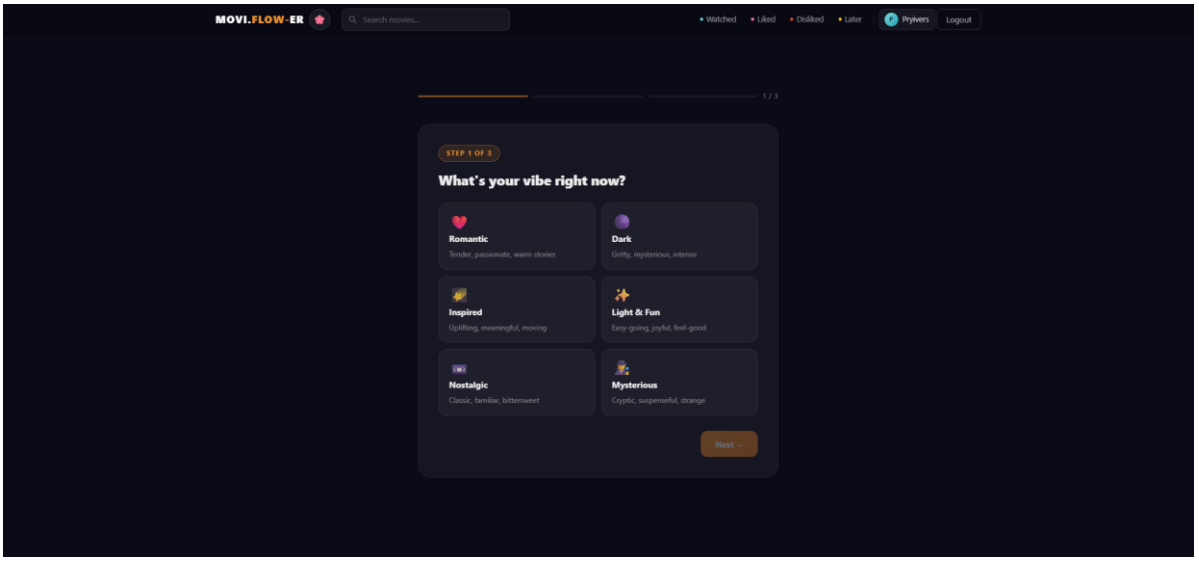


Рисунок Г.6 – Сторінка тестів

На даній сторінці може відображатися:

— етапи тестування;

— прогрес тестування;

Сторінка використовується авторизованим користувачем для покращення пошуку фільму. Тестова сторінка не призначена для звичайних користувачів та може бути доступною лише для авторизованих.

Отже, розроблена система підбору фільмів забезпечує повний цикл взаємодії користувача з рекомендаційним сервісом, а саме від авторизації до отримання персоналізованих рекомендацій та аналізу історії переглядів. Інтерфейс системи є логічно структурованим, функціонально завершеним та зручним для використання.

Система демонструє стабільність роботи, коректність обробки даних та відповідність поставленим технічним вимогам. Реалізований механізм рекомендацій забезпечує адаптивність до вподобань користувача, що підвищує ефективність підбору фільмів та покращує користувацький досвід.

Таким чином, програмний продукт є цілісним, функціонально обґрунтованим та готовим до практичного використання.

ДОДАТОК Д

(обов’язковий)

ПРЕЗЕНТАЦІЙНІ МАТЕРІАЛИ

Рисунок Д.1 – Слайд: Титульна сторінка»

Рисунок Д.2 – «Слайд: Актуальність теми»

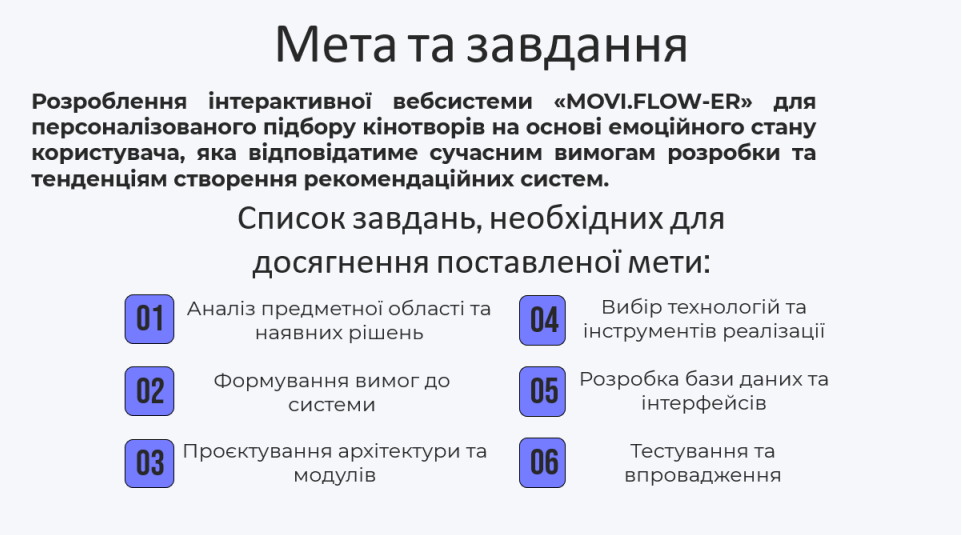


Рисунок Д.3 – «Слайд: Мета та завдання»

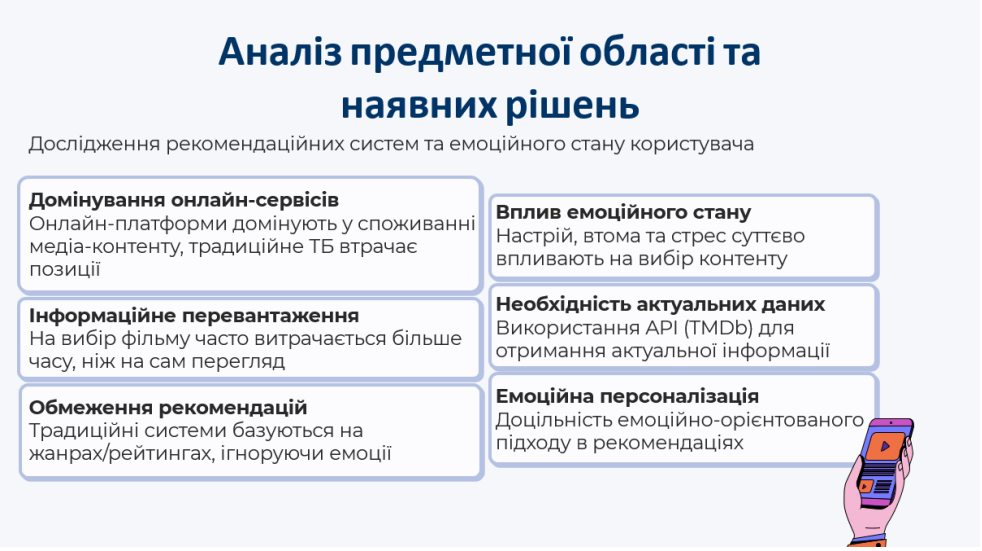


Рисунок Д.4 – «Слайд: Аналіз предметної області»

Рисунок Д.5 – «Слайд: Порівняльний аналіз аналогів»

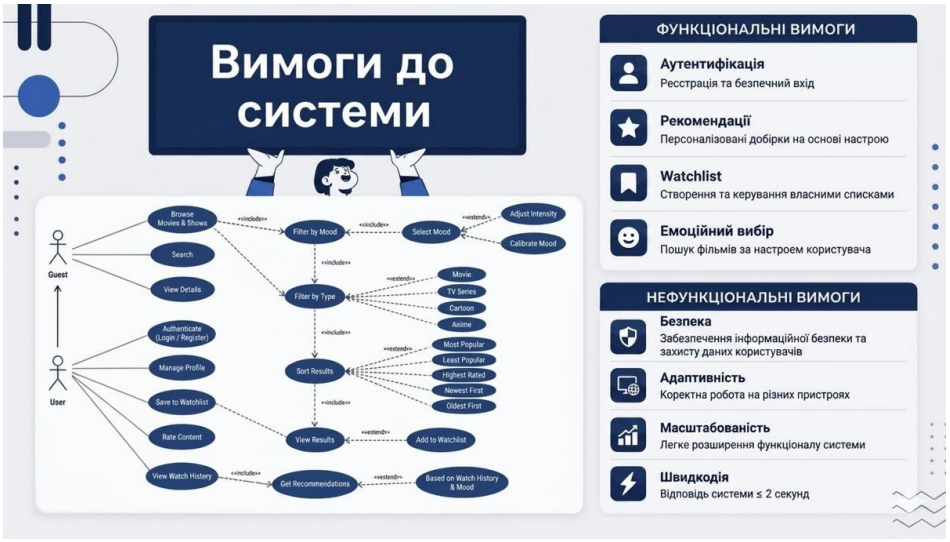


Рисунок Д.6 – «Слайд: Вимоги до системи»

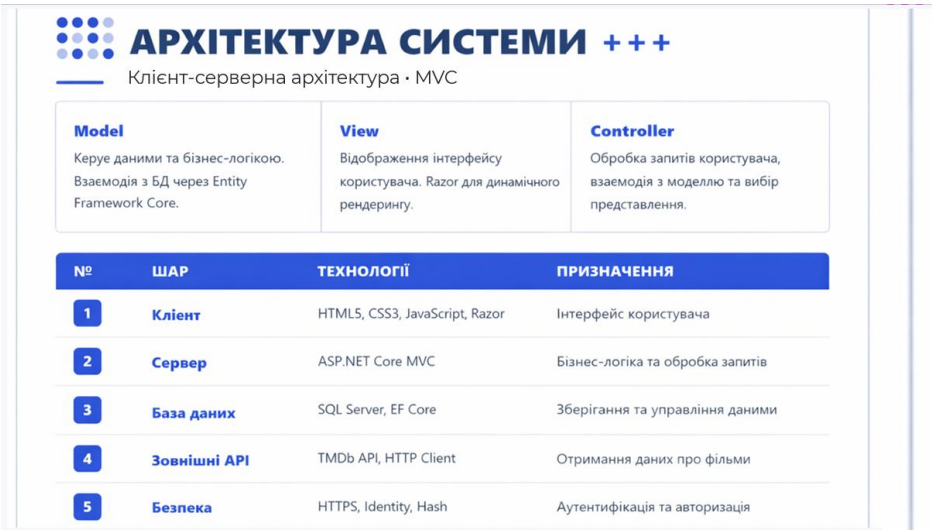


Рисунок Д.7 – «Слайд: Архітектура системи»

Рисунок Д.8 – «Слайд: Декомпозиція та інтерфейси»

Рисунок Д.9 – «Слайд: Проєктування модулів та даних»

Рисунок Д.10 – «Слайд: Аналіз та вибір технологій»

Рисунок Д.11 – «Слайд: Програмна реалізація та тестування»

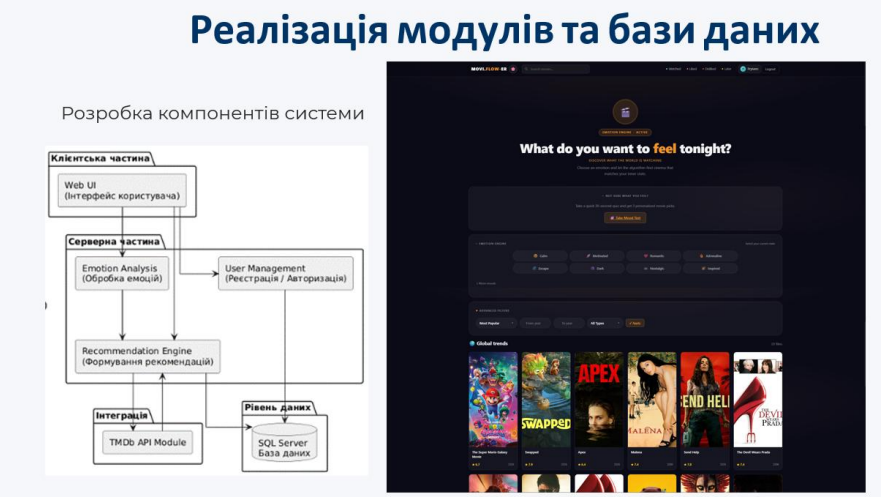


Рисунок Д.12 – «Слайд: Реалізація модулів та бази даних»

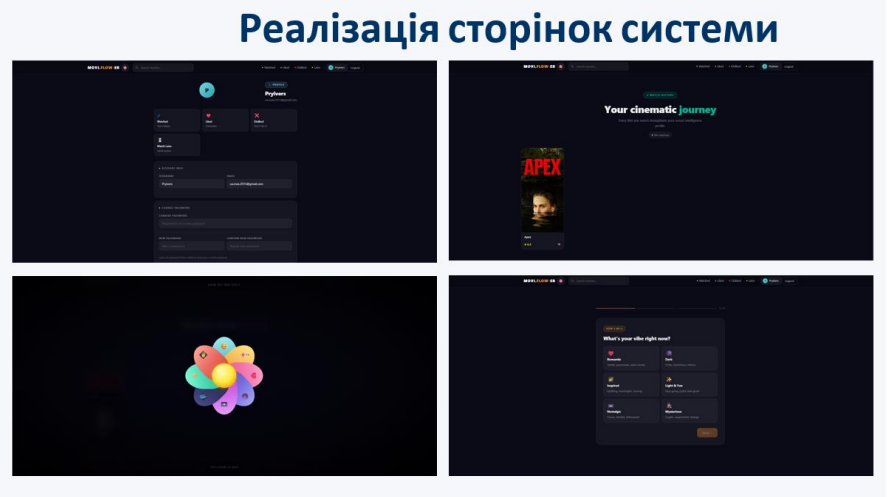


Рисунок Д.13 – «Слайд: Реалізація сторінок системи»

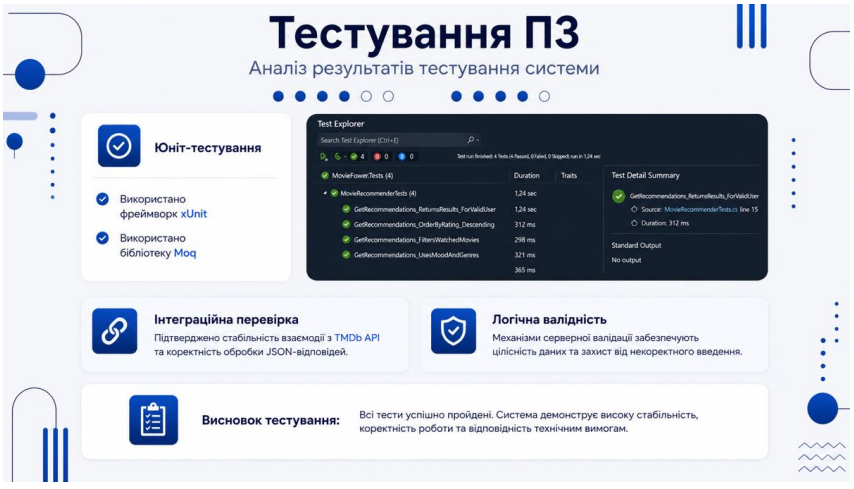


Рисунок Д.14 – «Слайд: Тестування ПЗ»

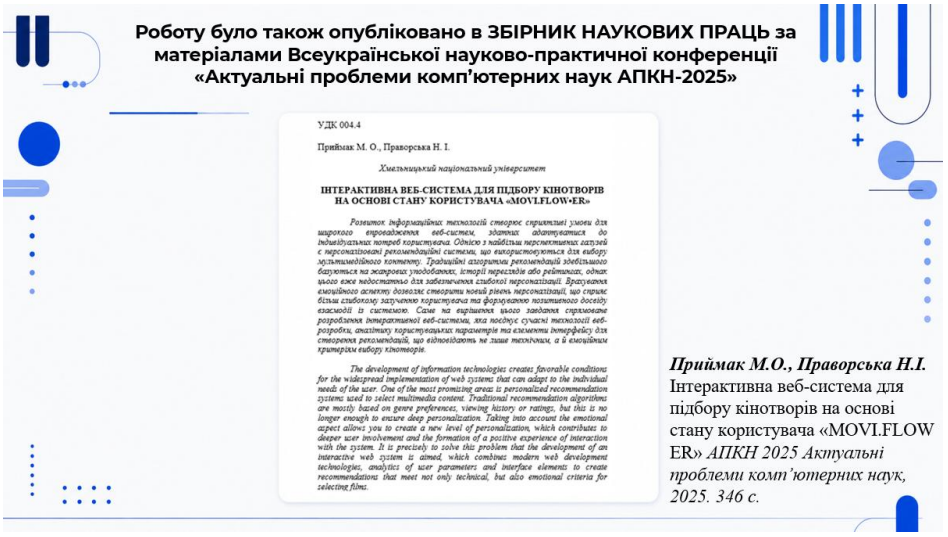


Рисунок Д.15 – «Слайд: Збірник наукових праць»

Рисунок Д.16 – «Слайд: Висновки»

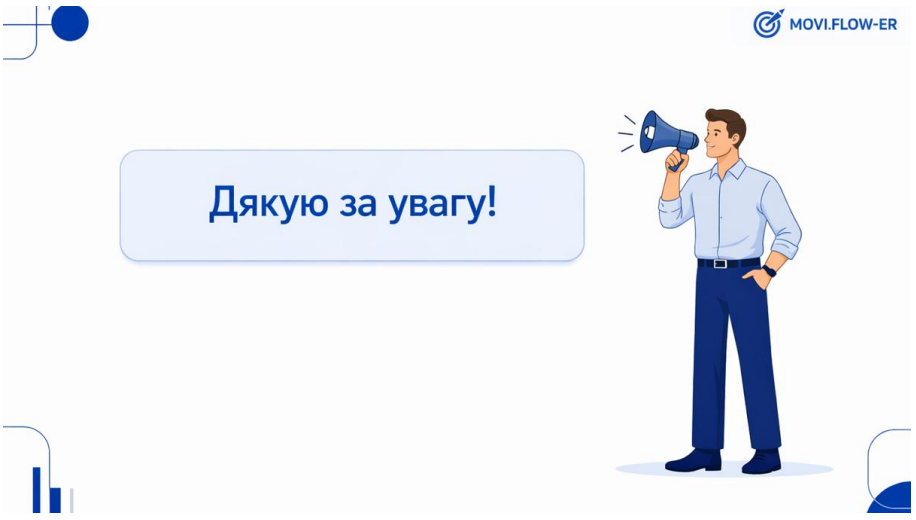


Рисунок Д.17 – «Слайд: Дякую за увагу!»

ДОДАТОК Е

(обов’язковий)

АНТИПЛАГІАТ

Thu May 21 11:30:01 EEST 2026, Форкун Юрій Вікторович, Хмельницький національний університет, ХНУ

Anti-Plagiarism (http://ap.km.ua) v-16.718

Максимальне співпадіння з одним документом 2.0%

Словники перевірки: UA, US, RU. **Помилок в документах: 12%**

ID: 271873 Назва: БКР Інтерактивна система для підбору фільмів Додано в БД: 2026-05-21 Автора: Максим ПРИЙМАК Керівники: канд. пед. наук, доцент Наталія ПРАВОРСЬКА Консультанти: Опоненти:	Документ		Сумарний збіг по Базі Даних	
	Символи	Лексеми	Символи	Лексеми
	111240	818	2900 (3%)	38 (5%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

Протокол аналізу звіту подібності науковим керівником

Заявляю, що я ознайомився (-лась) з Повним звітом подібності, який був згенерований Системою виявлення і запобігання плагіату щодо роботи:

Автор: Максим ПРИЙМАК

Співавтор:

Назва: Інтерактивна система для підбору фільмів на основі емоційного стану користувача «MOVI.FLOW-ER»

Науковий керівник: канд. пед. наук, доцент Наталія ПРАВОРСЬКА

Підрозділ: Кафедра інженерії програмного забезпечення

Коефіцієнт подібності 1:5.43%

Коефіцієнт подібності 2:1.1%

Мікропробіли: 22

Заміна букв: 14

Інтервали: 33

Білі знаки: 0

Дата створення звіту: 2026-05-21 10:04:27.0

Після аналізу Звіту подібності констатую наступне:

☑ **Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається.**

☐ **Запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Таким чином робота повертається на доопрацювання.**

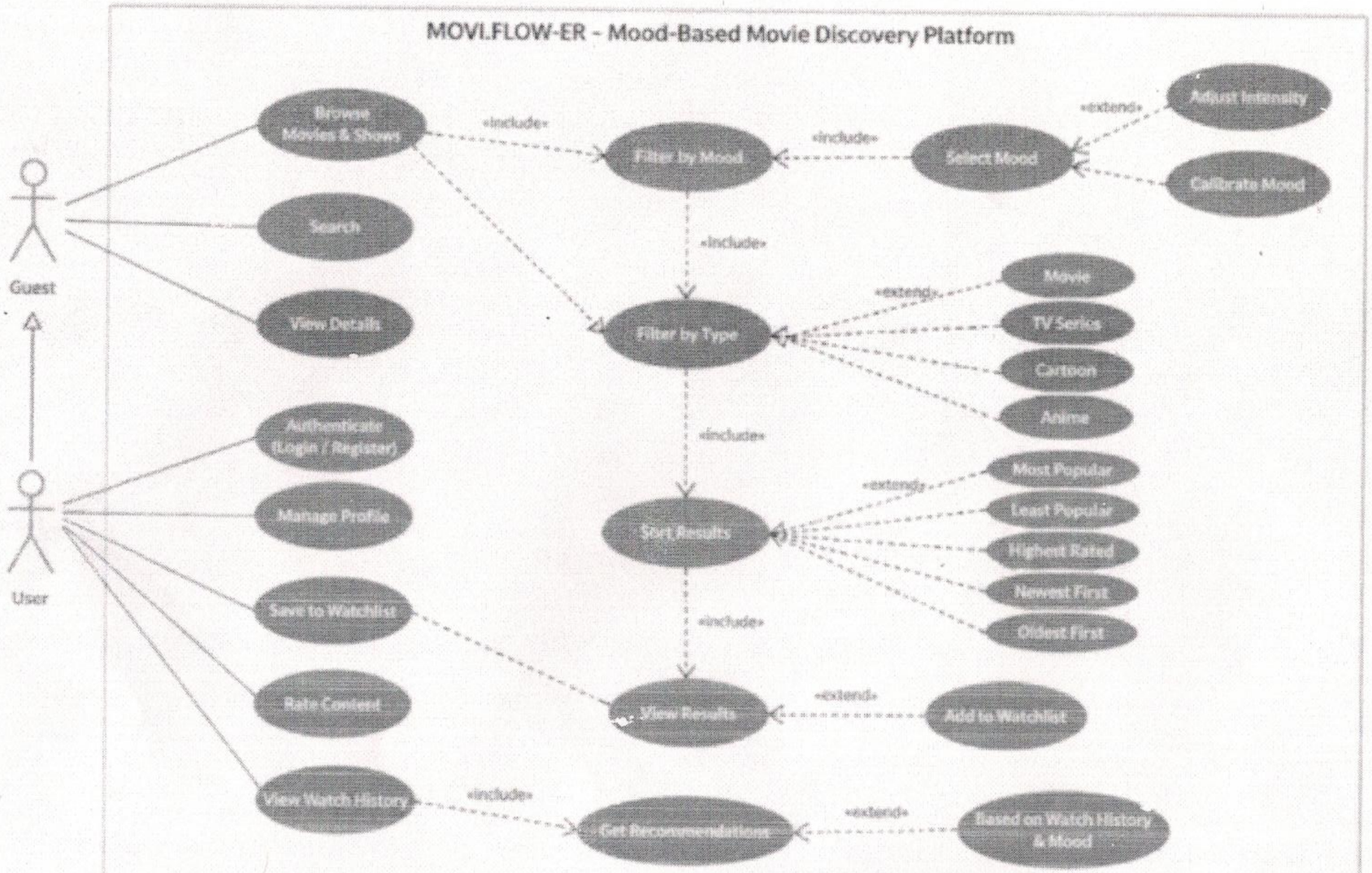
☐ **Виявлено запозичення і плагіат або навмисні текстові спотворення (маніпуляції), як передбачувані спроби укриття плагіату, які роблять роботу невідповідною вимогам законодавства (Ст. 32. ЗУ Про вищу освіту, пункт 3.1, Ст. 42. ЗУ Про освіту) та вимог НАЗЯВО (Критерій 5), а також кодексу етики і процедурам. Таким чином робота не приймається.**

Обгрунтування:

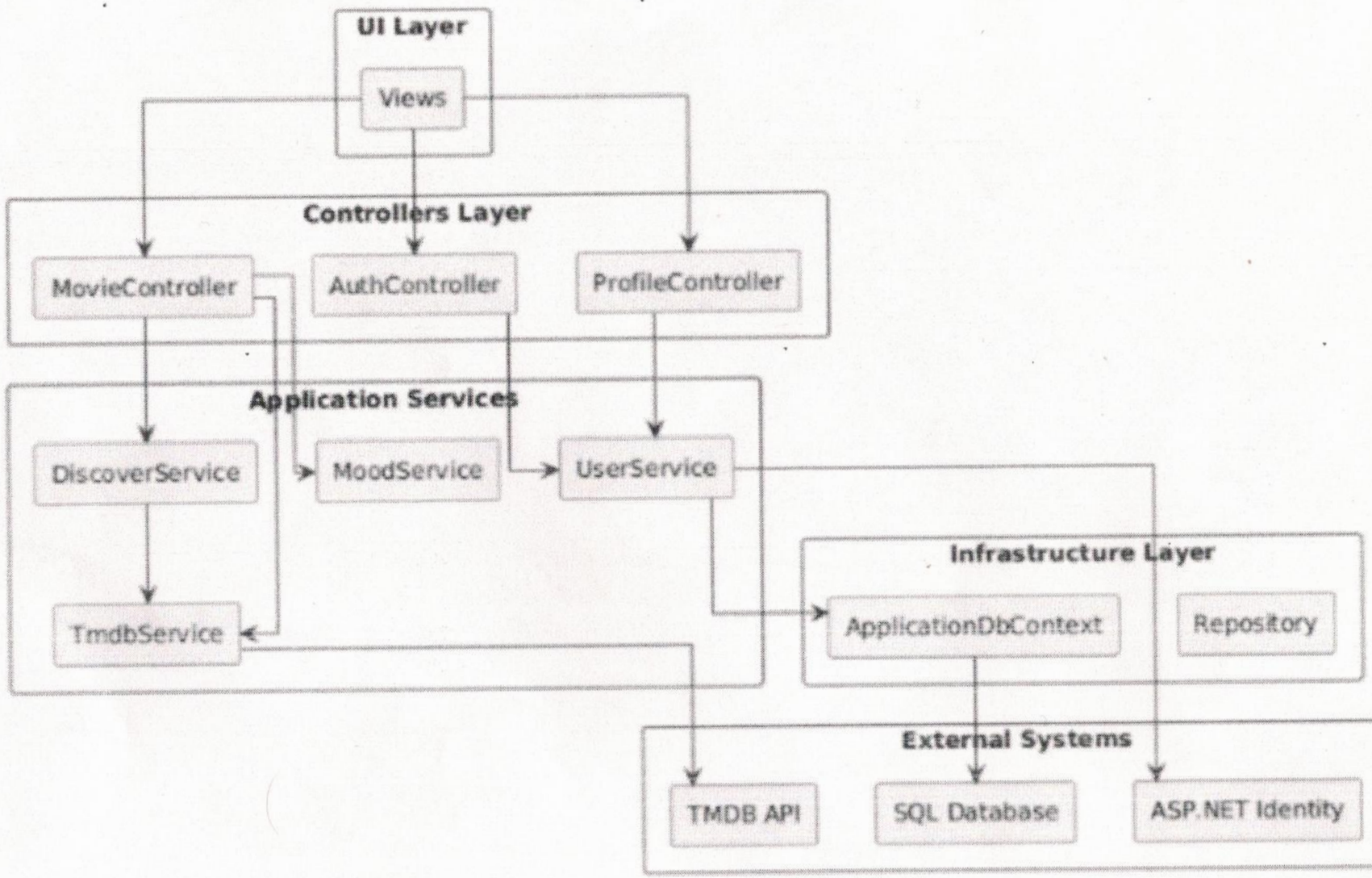
Дата експерт

25.05.2026

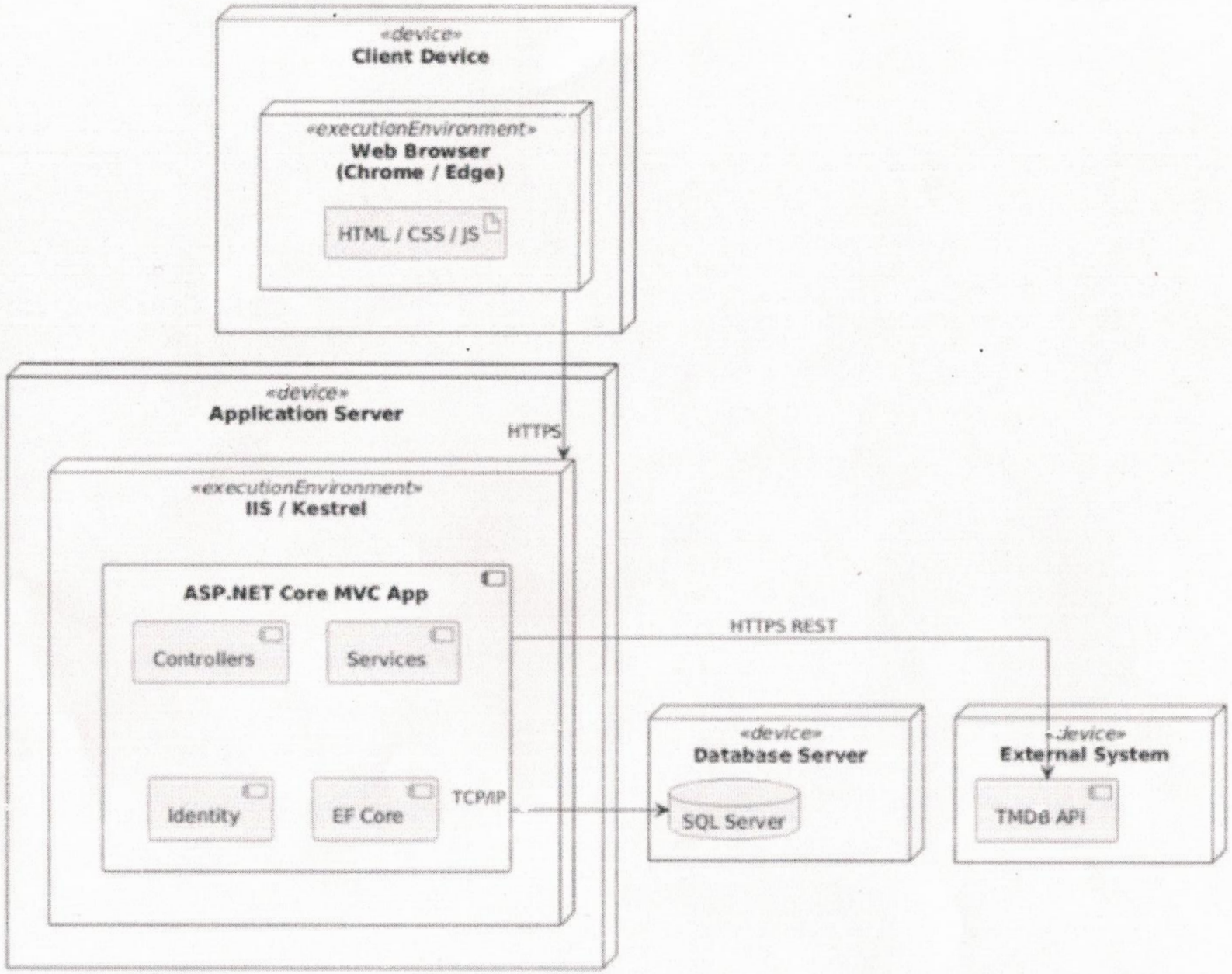
ГРАФІЧНІ МАТЕРІАЛИ



Зм.	Арк.	№ докум.	Підпис	Дата	КвРІПЗ.2201107.01.13.Е8		
Розробив		Приймак М.О.		27.05.	Діаграма варіантів використання	Літера	Маса	Масштаб
Керівник		Праворська Н.І.		27.05.				
Консульт.						Аркуш 1	Аркушів 3	
Н. Контр.		Яшина О. М.		27.05.		ХНУ, ІПЗ-22-1		
Зав. каф.		Бездратюк Л.П.		27.05.				



					КвРІПЗ.2201107.01.13.Е8	
						Літера / Маса / Масштаб
Зм.	Арк.	№ докум.	Підпис	Дата	Діаграма зв'язків модулів	
Розробив		Приймак М.О.		27.05.		Аркуш 2 / Аркушів 3
Керівник		Праворська Н.І.		27.05.		
Консульт.						
Н. Контр.		Яшина О. М.		27.05.		ХНУ, ІПЗ-22-1
Зав. каф.		Бедратюк Л.Д.		27.05.		



					КвРІПЗ.2201107.01.13.Е8		
Зм.	Арк.	№ докум.	Підпис	Дата			
Розробив		Приймак М.О.		17.05	Діаграма розгортання	Літера	Маса / Масштаб
Керівник		Праворська Н.І.		17.05			
Консульт.						Аркуш 3	Аркушів 3
Н. Контр.		Яцина О. М.		17.05.		ХНУ, ІПЗ-22-1	
Зав. каф.		Бадратов Л.П.		17.05.			

СУПРОВІДНІ ДОКУМЕНТИ

Завідувачу кафедри інженерії програмного
забезпечення проф. Леоніду БЕДРАТЮКУ
здобувача вищої освіти
Приймака Максима Олександровича
факультет ІТ, IV курс, група ІПЗ-22-1

ЗАЯВА

З правилами чинного Положення про систему забезпечення академічної доброчесності в Хмельницькому національному університеті, згідно з яким виявлення академічного плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту і застосування заходів дисциплінарної та академічної відповідальності, ознайомлений. Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на наявність академічного плагіату оповіщений та надаю свою згоду на обробку й збереження університетом моєї роботи в інституційному репозитарії Хмельницького національного університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-обчислювального комплексу StrikePlagiarism та/або програмно-технічного засобу AntiPlagiarism і використання роботи для виявлення академічного плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення текстових збігів у роботах.

Робота надається для перевірки в електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

02.09.2026
дата

підпис

Anti-Plagiarism (http://ap.km.ua) v-16.718

Максимальне співпадіння з одним документом 2.0%

Словники перевірки: UA, US, RU. **Поеклок в документах: 12%**

ID: 271873 Назва: БКР Інтерактивна система для підбору фільмів Додано в КД: 2026-05-21 Автора: Максим ПРИЙМАК Керівник: канд. пед. наук, доцент Наталя ПРАВОРСЬКА Консультанти: Оцінювати:	Документ		Сумарний збіг по Базі Даних	
	Символи	Лексеми	Символи	Лексеми
	111240	818	2900 (3%)	38 (5%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

Протокол аналізу звіту подібності науковим керівником

Заявляю, що я ознайомився (-лась) з Повним звітом подібності, який був згенерований Системою виявлення і запобігання плагіату щодо роботи:

Автор: Максим ПРИЙМАК

Співавтор:

Назва: Інтерактивна система для підбору фільмів на основі емоційного стану користувача «MOVI.FLOW-ER»

Науковий керівник: канд. пед. наук, доцент Наталія ПРАВОРСЬКА

Підрозділ: Кафедра інженерії програмного забезпечення

Коефіцієнт подібності 1:5.43%

Коефіцієнт подібності 2:1.1%

Мікропробіли: 22

Заміна букв: 14

Інтервали: 33

Білі знаки: 0

Дата створення звіту: 2026-05-21 10:04:27.0

Після аналізу Звіту подібності констатую наступне:

☑ **Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається.**

☐ **Запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Таким чином робота повертається на доопрацювання.**

☐ **Виявлено запозичення і плагіат або навмисні текстові спотворення (маніпуляції), як передбачувані спроби укриття плагіату, які роблять роботу невідповідною вимогам законодавства (Ст. 32. ЗУ Про вищу освіту, пункт 3.1, Ст. 42. ЗУ Про освіту) та вимог НАЗЯВО (Критерій 5), а також кодексу етики і процедура м. Таким чином робота не приймається.**

Обгрунтування:

Дата
25.05.2026

експерт

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ
освітнього ступеня «Бакалавр»

Дипломник Приймак Максим Олександрович

Тема Інтерактивна система для підбору фільмів на основі емоційного стану користувача «MOVI.FLOW-ER»

Спеціальність 121 – Інженерія програмного забезпечення

Обсяг кваліфікаційної роботи:

Кількість листів креслень 3; кількість сторінок записки 59.

1.Короткий зміст пояснювальної записки та прийнятих рішень У кваліфікаційній роботі проведено дослідження предметної області інтерактивних систем підбору кінофільмів та рекомендаційних платформ. Було виконано аналіз існуючих рішень у сфері стрімінгових сервісів і систем рекомендацій, визначено їх переваги та недоліки, а також обґрунтовано актуальність створення власного програмного забезпечення з урахуванням емоційного стану користувача. Було реалізовано систему рекомендацій фільмів, функції авторизації та реєстрації користувачів, формування персоналізованих списків, оцінювання фільмів, історії переглядів та механізмів емоційного підбору контенту. Також проведено модульне тестування системи за допомогою xUnit та Moq, результати якого підтвердили коректність роботи програмного забезпечення та його готовність до використання.

2. Висновок про відповідність роботи поставленому завданню Кваліфікаційна робота виконана відповідно до поставленого завдання та повністю відповідає вимогам до бакалаврських кваліфікаційних робіт. Усі поставлені цілі та завдання були досягнуті у повному обсязі.

3. Характеристика виконання кожного розділу роботи, ступінь використання останніх досягнень науки і техніки та передових методів роботи У вступі обґрунтовано актуальність теми кваліфікаційної роботи, визначено мету, завдання, об’єкт та предмет дослідження. У першому розділі виконано аналіз предметної області, досліджено сучасні стрімінгові сервіси та рекомендаційні системи, розглянуто проблеми інформаційного перевантаження користувачів та необхідність персоналізованого підбору контенту. Також сформовано функціональні та нефункціональні вимоги до програмного забезпечення. У другому розділі спроєктовано архітектуру системи, побудовано UML-діаграми, ER-модель бази даних та визначено структуру взаємодії між компонентами системи. Було обґрунтовано використання клієнт-серверної архітектури та шаблону MVC. У третьому розділі виконано практичну реалізацію програмного продукту із використанням сучасного стеку технологій ASP.NET Core MVC, Entity Framework Core, SQL Server та TMDb API. Реалізовано основний функціонал системи, інтерфейс користувача, механізми безпеки та взаємодію із зовнішніми API. Також проведено модульне тестування системи, результати якого підтвердили стабільність та коректність роботи програмного забезпечення.

4. Позитивні сторони роботи Тематика кваліфікаційної роботи є актуальною, оскільки сучасні системи рекомендацій контенту потребують більш глибокого рівня персоналізації та врахування емоційного стану користувачів. Позитивною стороною роботи є використання сучасних технологій та архітектурних підходів для створення вебзастосунку. Автором реалізовано зручний інтерфейс користувача, систему персоналізованих рекомендацій, інтеграцію із зовнішнім API та модульне тестування програмного забезпечення. Також слід відзначити якісне графічне оформлення презентаційних матеріалів, логічну структуру пояснювальної записки та практичну цінність розробленої системи.

5. Негативні сторони роботи У роботі рекомендаційна система реалізована переважно на основі вподобань та емоційного стану користувача, однак доцільним було б розширення алгоритмів рекомендацій із використанням методів машинного навчання. Також перспективним напрямом розвитку системи може бути реалізація багатокористувацьких рекомендацій та інтеграція із мобільними платформами.

6. Оцінка графічного оформлення та пояснювальної записки Графічне оформлення виконано відповідно до тематики кваліфікаційної роботи та містить необхідні UML-діаграми, схеми архітектури системи, ER-модель бази даних та ілюстрації інтерфейсу користувача. Пояснювальна записка оформлена згідно вимог чинних стандартів, матеріал викладено послідовно, структуровано та зрозуміло.

7. Відгук про кваліфікаційну роботу в цілому Кваліфікаційна робота виконана на достатньо високому рівні та демонструє належний рівень теоретичної підготовки й практичних навичок автора у сфері розробки програмного забезпечення. У процесі виконання роботи автор проявив вміння аналізувати предметну область, проєктувати архітектуру програмних систем, працювати із сучасними технологіями веброзробки та реалізовувати повноцінний програмний продукт. Робота має практичну цінність, а отримані результати можуть бути використані для подальшого розвитку систем рекомендацій контенту.

8. Інші зауваження Суттєвих зауважень до кваліфікаційної роботи не виявлено.

9. Оцінка кваліфікаційної роботи Кваліфікаційна робота виконана у повному обсязі, відповідає поставленим вимогам та заслуговує на оцінку «відмінно».

РЕЦЕНЗЕНТ Янсієв Віталій Миколайович, к. ф.-м. н., доцент, доцент кафедри КІТС, ХНУ

“27” травня 2026 р. _____
(підпис)

Зберегти як PDF | Назад

SemanticAI for Education

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

першого освітнього рівня «Бакалавр»

Студента: Максим ПРИЙМАК
Група: ІП3-22-1

Тема: *«Інтерактивна система для підбору фільмів на основі емоційного стану користувача «MOVI.FLOW-ER»»*

Спеціальність: 121 – Інженерія програмного забезпечення

Короткий зміст пояснювальної записки

У вступі до кваліфікаційної роботи акцентується на актуальності розробки інтерактивної вебсистеми для підбору фільмів на основі емоційного стану користувача. Мета роботи полягає в створенні системи, яка забезпечить персоналізований підбір контенту, враховуючи психологічні аспекти вибору. Завдання, що стоять перед розробкою, включають аналіз предметної області, дослідження існуючих платформ, формування вимог, розробку архітектури системи, вибір технологій реалізації та тестування.

Відповідність отриманих результатів роботи поставленим завданням

Завдання, сформульовані у вступі, включають: 1. Провести аналіз предметної області персоналізованих вебсистем. 2.Дослідити існуючі платформи підбору мультимедійного контенту. 3. Сформувати функціональні та нефункціональні вимоги до системи. 4. Розробити архітектуру системи та взаємодію її основних компонентів. 5. Спланувати структуру бази даних та модулів системи. 7. Обрати технології та інструменти реалізації. 7. Реалізувати програмну частину проекту. 8. Виконати візуальну частину даної системи. 9. Провести тестування функціональності та стабільності роботи проекту. 10. Оцінити результат та перспективи подальшого розвитку вебсистеми.

Результати, викладені у висновках, свідчать про те, що: 1. Аналіз предметної області проведено, виявлено основні проблеми та потреби користувачів. 2. Існуючі платформи досліджені, виявлено їхні переваги та недоліки. 3. Функціональні та нефункціональні вимоги сформульовані чітко. 4. Архітектура системи розроблена, враховуючи всі компоненти. 5. Структура бази даних та модулів спланована. 6. Технології реалізації обрані відповідно до вимог. 7. Програмна частина реалізована, система готова до використання. 8. Тестування проведено, результати підтверджують працездатність системи.

Отже, результати роботи **повністю відповідають** поставленим завданням.

Оцінка розділів

Розділ 1: Аналіз предметної області

У цьому розділі представлено змістовний аналіз предметної області, що охоплює структурні та функціональні особливості. Описано проблеми, які вирішує програмне забезпечення, а також визначено користувачів та їх інформаційні потреби. Однак, деякі аспекти потребують уточнення, зокрема, відсутні конкретні приклади існуючих систем та їх функціональних можливостей. Загалом, розділ демонструє глибоке розуміння теми, але потребує більшої структурованості та деталізації.

Розділ 2: Проєктування програмного забезпечення

Розділ містить детальний опис вибору архітектури, структури даних та модулів. Однак, відсутні порівняльні таблиці та чіткі схеми, що ускладнює сприйняття інформації. Вибір архітектури обґрунтовано, але не завжди чітко пов'язано з вимогами. Загалом, розділ є логічним, але потребує покращення в плані візуалізації та деталізації.

Розділ 3: Програмна реалізація та тестування

У цьому розділі описано реалізацію бази даних та програмних модулів, а також методи тестування. Однак, відсутні фрагменти коду та графіки, які б покращили візуалізацію інформації. Загалом, розділ є структурованим, але потребує уточнень та доповнень для підвищення якості.

Позитивні сторони

Кваліфікаційна робота демонструє оригінальність у підході до розробки персоналізованої системи підбору контенту, що враховує емоційний стан користувача. Висока якість рішень, зокрема у виборі технологій та архітектури, свідчить про глибоке розуміння предметної області. Зручність для користувача підкреслюється акцентом на інтуїтивно зрозумілому інтерфейсі та швидкому отриманні результатів.

Недоліки

Незважаючи на позитивні аспекти, робота має недоліки, зокрема недостатню деталізацію в описі деяких розділів, відсутність конкретних прикладів та візуалізацій, що ускладнює сприйняття інформації. Деякі висновки можуть бути більш структурованими, а аргументація потребує додаткових прикладів.

Відгук в цілому

Кваліфікаційна робота є актуальною та має практичну значущість, оскільки відповідає сучасним вимогам до персоналізованих цифрових сервісів. Зміст роботи відповідає темі та завданню, новизна ідей та рішень є очевидною. Об'єктивність та обґрунтованість викладених матеріалів підтверджується проведеним аналізом. Програмний продукт демонструє працездатність та відповідність технічному завданню, а також використовує сучасні технології.

Оцінка кваліфікаційної роботи

Кваліфікаційна робота заслуговує оцінки **добре**. Вона виконана в повному обсязі з дотриманням основних вимог, хоча є деякі недоліки, які потребують доопрацювання. Здобувач володіє матеріалом, грамотно викладає суть роботи, але може потребувати підказок при відповідях на детальні запитання.

Рекомендації

Рекомендується доопрацювати роботу, зокрема уточнити недоліки в описі розділів, додати візуалізації та приклади, що підвищить якість та зрозумілість матеріалу. У разі врахування цих рекомендацій, робота має потенціал до впровадження та подальшого розвитку.

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ
КАФЕДРИ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатами звіту/звітів перевірки роботи, продукованими програмно-технічним засобом (ами), на наявність текстових збігів.
Назва кваліфікаційної роботи: «Інтерактивна система для підбору фільмів на основі емоційного стану користувача «MOVI.FLOW-ER»»
Автор: Приймак Максим Олександрович
Освітня програма: Освітньо-професійна програма «Інженерія програмного забезпечення»
Спеціальність: 121 – Інженерія програмного забезпечення
Науковий керівник: Праворська Наталія Іванівна, канд. пед. наук, доцент
Після аналізу звіту/звітів зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається до захисту.	**відповідає**
2	Виявлені запозичення не є академічним плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота приймається до захисту, але має бути відкоригована.	
3	Виявлені запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Виявлені запозичення частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнуті. Робота може бути допущена до захисту після того, як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття текстових запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
5	Інше:	

Підтвердження:
Запозичення, виявлені у роботі, є законними і не є плагіатом, оскільки:

1) у тексті кваліфікаційної роботи системою перевірки на плагіат Anti-Plagiarism виявлено схожість з деякими документами у частині загальновживаних обов'язкових словосполучень у стандартних бланках, у структурі змісту, назвах назв розділів/підрозділів, рамках форм, у назвах та URL-адресах публікацій переліку джерел посилання;
2) запозичення, виявлені у тексті роботи, є фрагментарними.

Максимальний обсяг запозичень, визначений системою Anti-Plagiarism, складає 2.0 % з одного джерела. Загальна сумарна подібність у базі даних складає 3 % за символами та 5 % за лексемами. Крім того, за результатами додаткового аналізу коефіцієнт подібності 1 становить 5.43 %, коефіцієнт подібності 2 – 1.1 %. Кількість мікропробілів становить 22, маніпуляцій з інтервалами – 33., білих знаків не було виявлено. З урахуванням наведених обґрунтувань, відповідне характеру теми і свідчить на користь кваліфікаційної роботи.

Дата 27.05.2026

Завідувач кафедри _____ Леонід БЕДРАТЮК

Гарант освітньої програми _____ Леонід БЕДРАТЮК

Керівник кваліфікаційної роботи _____ Наталія ПРАВОРСЬКА