КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ

автоматизації і інформаційних технологій

(факультет)

інформаційних технологій

(кафедра)

ПОЯСНЮВАЛЬНА ЗАПИСКА

ДО АТЕСТАЦІЙНОЇ ВИПУСКНОЇ РОБОТИ
НА ЗДОБУТТЯ ОСВІТНЬОГО РІВНЯ «БАКАЛАВР»

на тему: «Розроблення веб-додатку з формування списку фільмів»

Рябенко Євгеній Вячеславович

(прізвище, ім’я та по батькові студента повністю)

Київ 2023 р.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ

автоматизації і інформаційних технологій
(факультет)

інформаційних технологій
(кафедра)

ЗАТВЕРДЖУЮ
Завідувач кафедри ІТ
д.т.н., професор Цюцюра С.В.
_____
«___» _____20___ року

ПОЯСНЮВАЛЬНА ЗАПИСКА

ДО АТЕСТАЦІЙНОЇ ВИПУСКНОЇ РОБОТИ НА ЗДОБУТТЯ ОСВІТНЬОГО РІВНЯ «БАКАЛАВР»

на тему: «Розроблення веб-додатку з формування списку фільмів»

Виконав:студент 4-го курсу, групи КН-41

Спеціальності: 122 «Комп`ютерні науки .

Спеціалізація: «Інформаційні управляючі системи та технології»
(шифр і назва напряму підготовки, спеціальності)

Рябенко Є.В.
(прізвище та ініціали)

Керівник к.т.н., доц. Горда О.В.
(прізвище та ініціали)

Рецензент к.т.н., доц. Доля О.В.
(прізвище та ініціали)

Київ, 2023 р.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ

Факультет: __автоматизації і інформаційних технологій
Кафедра:__ __інформаційних технологій
Освітній рівень: «бакалавр» за ОПП
Спеціальність: _122 «Комп`ютерні науки»
Спеціалізація:__ Інформаційні управляючі системи та технології

ЗАТВЕРДЖУЮ
Завідувач кафедри ІТ
д.т.н., професор Цюцюра С.В.
_____
„___” _____2023 року

З А В Д А Н Н Я
ДО ВИКОНАННЯ АТЕСТАЦІЙНОЇ ВИПУСКНОЇ РОБОТИ
НА ЗДОБУТТЯ ОСВІТНЬОГО РІВНЯ «БАКАЛАВР»

_____Рябенко Євгеній Вячеславович_____
(прізвище, ім’я, по батькові)

1. Тема роботи: Розроблення веб-додатку з формування списку фільмів
керівник роботи: к.т.н., доц. Горда Олена Володимирівна
затверджені наказом ректора КНУБА № 1811/2 від « 17» листопада 2022 р.
2. Термін подачі студентом роботи до захисту: 01.червня 2023 .
3. Вихідні дані до роботи _____
4. Зміст пояснювальної записки: Вступ 1. Аналіз предметної області та постановка задачі. 2. Розробка інформаційного забезпечення і моделювання БД. 3. Розробка програмного забезпечення та приклади функціоналу. Тестовий приклад роботи програмного продукту. 4. Техніко-економічне обґрунтування розробки підсистеми (Бізнес-план)
5. Перелік презентаційно-інформаційних слайдів:

6. Консультанти розділів атестаційної випускної роботи

Розділ	Прізвище, ініціали та посада консультанта, представника комісії	дата	підпис
Техніко-економічне обґрунтування розробки підсистеми (Бізнес-план)	д.т.н. проф. Цюцюра С.В.		
Прийом програмного продукту	к.т.н., доц. Єрукаєв		

7. Дата видачі завдання: 15 лютого 2023 р.

КАЛЕНДАРНИЙ ПЛАН

Види робіт та їх зміст	Дата виконання
Р. 1. Аналіз предметної області та постановка задачі	Лютий 2023 р.
Р. 2. Розробка інформаційного забезпечення та моделювання БД	Лютий 2023 р.
Р. 3. Розробка програмного забезпечення. Приклади функціоналу	Березень 2023 р.
Р. 4. Техніко-економічне обґрунтування розробки підсистеми (Бізнес-план)	Квітень 2023 р.
Оформлення роботи	Травень 2023 р.
Направлення роботи на рецензування	Травень 2023 р.
Попередній захист роботи на кафедрі	Червень 2023 р.

Бакалавр _____ Рябенко Є.В.
(підпис) (прізвище та ініціали)

Керівник _____ Горда О.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Рябенко Є.В. «Розроблення веб-додатку з формування списку фільмів».

Атестаційна випускова робота бакалавра за спеціальністю: 122 «Комп'ютерні науки», спеціалізація: «Інформаційні управляючі системи і технології проектування». – Київський національний університет будівництва і архітектури. – Київ, 2023.

В атестаційній випусковій роботи розроблена підсистема автоматизації і управління власним списком фільмів, з керуванням БД на основі вибору SQLite

Ключові слова: інформаційні технології, база даних, концептуальна база даних, веб-додаток

SUMMARY

Ryabenko E.V. "Development of a web application for generating a list of movies". Attestation final work of the bachelor's degree in the specialty: 122 "Computer Science", specialization: "Information technologies of designing". - Kyiv National University of Construction and Architecture. - Kyiv, 2023.

In the certification graduation work, a subsystem for automating and managing your own list of movies, with database management based on SQLite selection, was developed

Key words: information technology, database, conceptual database, web application

Зміст

ВСТУП

У сучасному світі розваги відіграють вирішальну роль у нашому повсякденному житті. Одним з найпоширеніших видів розваги є перегляд фільмів, і у кожного є улюблений фільм, який він із задоволенням переглядає знову і знову. Однак може бути важко відстежити всі ваші улюблені фільми. Наш веб-додаток задовольнить цю потребу.

Мета нашого веб-додатку — полегшити кіноманам складання та ведення власного списку улюблених фільмів. Користувачі можуть додавати та видаляти фільми зі свого списку, давати їм оцінки та писати стислі рецензії за допомогою нашої зручної платформи. Користувачі також можуть миттєво знаходити свої улюблені фільми, шукаючи їх за назвою, жанром, режисером або актором.

Предмет дослідження: Веб додаток фокусується на царині кіно та його впливі на людей. Він визнає важливість фільмів як форми розваги, мистецтва та культурного самовираження. Вивчаючи вподобання та вибір користувачів, Веб додатокпрагне отримати уявлення про різноманітні смаки та інтереси кінолюбителів. Ця інформація може бути використана для покращення користувацького досвіду, рекомендації відповідних фільмів та створення спільноти кіноманів.

Веб додаток використовує різні методи для досягнення своїх цілей. Ці методи включають: Реєстрація та аутентифікація користувачів; Інтеграція з базами даних фільмів; Створення та організація списківпорядку. Користувачі можуть класифікувати свої списки за темами, жанрами чи особистими вподобаннями.

Використовуючи ці методи, Веб додаток прагне стати цінним інструментом для кінолюбителів, спрощуючи процес створення списків і сприяючи спільній любові до кіно.

1. Аналіз предметної області та постановка

1.1Аналіз та дослідження проблеми

1.1.1 Аналіз проблеми

У сучасному інтернеті ми немаємо можливості нормально зберігати свої фільми, сериалі, мультфільми, тощо.

У сучасномі світі існує безліч фільмів які користувачі колись переглядали. Ціль
нашого дадутку полягає в тому, щоб дати користувачам можливість зберігати їх для себе, або переглянути відгуки інших користувачів на фільм або серіал. Також було б чудово мати додаток який зможе виконувати функцію соціальної мережі для любителів кінофільмів.

Щоб допомогти користувачам зберігати списки ми створили веб додаток, на зразок соціальної мережі де є можливість не тільки робити свій список, але й переглядати списки інших користувачів. [1]

Наш веб-додаток відповідає цим потребам, надаючи користувачам можливість легко зберігати, організовувати та спілкуватися щодо їхніх улюблених фільмів. Ми розуміємо, що велика кількість відеоматеріалів доступна онлайн, і важко втримати все це в пам'яті. Наш додаток дозволяє створювати власні списки, відзначати переглянуті твори, додавати відгуки та оцінки, а також взаємодіяти з іншими користувачами, які мають схожі інтереси.

Завдяки швидкому розвитку кінематографії та великій кількості фільмів, серіалів та мультфільмів, випущених кожного року, наш додаток допоможе вам легко зорієнтуватися в цьому морі розваг та створити власні списки для подальшої зручної навігації та спілкування з іншими любителями кіно.

1.1.2 Формування проблеми

Багато людей люблять дивитися фільми, і вони часто мають великий список улюблених фільмів. Однак люди зараз не можуть легко створити свій

власний список фільмів і керувати ним. Це може зайняти багато часу та дратувати, а також може призвести до розрізненого та переповненого списку фільмів.

Наша мета — створити веб-програму, яка дозволить користувачам створювати власні списки улюблених фільмів і керувати ними. За допомогою цієї програми користувачі зможуть швидко додавати фільми до своїх списків, шукати фільми за назвою, режисером, жанром або іншими критеріями, а також класифікувати свої фільми за кількома факторами, такими як жанр, рік і рейтинг.

Крім того, користувачі зможуть ділитися своїми списками з друзями та родиною, а також оцінювати та переглядати фільми.

Для любителів кіно відсутність доступного та простого у користуванні додатку зі списком фільмів викликає ряд проблем.

Ці проблеми складаються з:

1) Дезорганізація: без спеціальної платформи для керування улюбленими фільмами користувачам, можливо, доведеться покладатися на паперові списки, нотатки чи електронні таблиці, щоб відстежувати свої улюблені фільми. Це може призвести до захараценого та невпорядкованого списку фільмів, у якому важко орієнтуватися.
2) Труднощі з пошуком нових фільмів: без повної та актуальної бази даних фільмів користувачам може бути важко знайти нові та цікаві фільми для перегляду. Це може обмежити їхній вибір перегляду та завадити їм відкривати нові фільми.
3) Неефективне керування: додавання, категоризація та оновлення списків фільмів вручну може бути трудомістким і неефективним процесом. Це може перешкодити користувачам регулярно керувати своїми списками фільмів, що призведе до застарілих і неповних списків.[3]

1.2 Визначення цілей дослідження

1.2.1 Опис цілей дослідження

Проблема, яку ми намагаємося вирішити за допомогою нашого веб-додатку, полягає в тому, щоб надати людям платформу для створення та керування власним персональним списком улюблених фільмів. Щоб створити успішну та зручну програму, ми маємо вирішити кілька проблем і міркувань:

1) Управління даними. Одним із ключових завдань створення програми для списку фільмів є те, як керувати даними. Нам потрібно розглянути, як зберігати інформацію про фільм, включаючи назву, режисера, рік випуску, жанр і рейтинг, і як зробити цю інформацію легко доступною та доступною для пошуку для користувача.
2) Інтерфейс користувача. Іншим завданням є створення інтуїтивно зрозумілого та легкого у використанні інтерфейсу користувача. Користувачі повинні мати можливість швидко та легко додавати фільми до свого списку, шукати фільми за назвою чи іншим критерієм, а також переглядати та редагувати свій список фільмів.
3) Інтеграція із зовнішніми API: щоб надати користувачам повну базу даних фільмів, нам може знадобитися інтеграція із зовнішніми API, такими як API IMDb, для отримання інформації про фільми. Це вимагає ретельного розгляду того, як обробляти запити API, як кешувати дані та як обробляти помилки.
4) Безпека: ми повинні переконатися, що програма безпечна та захищає дані користувачів. Це включає впровадження безпечних механізмів автентифікації та авторизації, шифрування конфіденційних даних і дотримання найкращих практик щодо зберігання та обробки даних.
5) Продуктивність: із зростанням бази користувачів і збільшенням обсягу відеоданих продуктивність може стати проблемою. Нам потрібно оптимізувати додаток, щоб забезпечити швидке завантаження, мінімізувати навантаження на сервер і ефективно обробляти одночасні запити.

1.2.2 Дерево цілей

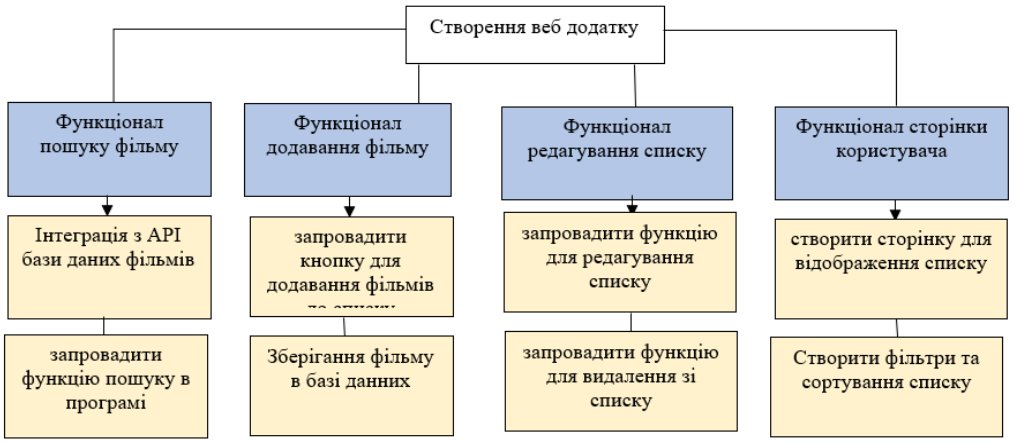


Рис. 1.1 Дерево цілей веб-додатку

1.3 *Порівняння подібних вже існуючих інструментів.*

1) IMDb (Internet Movie Database) - IMDb є однією з найпопулярніших баз даних фільмів в Інтернеті, яка дозволяє користувачам створювати власні списки улюблених фільмів. Користувачі можуть додавати фільми до своїх списків і впорядковувати їх будь-яким способом. Вони також можуть оцінювати фільми та писати рецензії. Однак одна проблема з IMDb полягає в тому, що для деяких користувачів це може бути непосильним, оскільки на сайті є дуже багато інформації.
2) Letterboxd – це сайт соціальних мереж для любителів кіно, який дозволяє користувачам стежити за іншими користувачами та переглядати їх списки, а також створювати списки своїх улюблених фільмів. Крім оцінювання та рецензування фільмів, учасники можуть спілкуватися один з одним через ряд соціальних елементів. Той факт, що Letterboxd може мати не так багато користувачів, як деякі інші платформи, означає, що користувачам може бути складніше знайти людей, за якими слід слідкувати.

3) Flixster – веб-сайт Flixster пропонує рекомендації та огляди фільмів. Користувачі також можуть складати списки своїх улюблених фільмів на сайті. Сервіс дозволяє користувачам оцінювати фільми та надавати рецензії, а потім пропонує фільми на основі цих оцінок. У минулому Flixster стикався з різними проблемами, такими як права власності та модифікації інтерфейсу, які могли зробити його менш зручним для користувачів.

1.3.1 Проблема існуючих сервісів:

Існує кілька ймовірних причин, чому нам може знадобитися новий онлайн-додаток для створення списків улюблених фільмів.

По-перше, користувальницький інтерфейс і користувацький досвід поточних служб все ще можуть мати потенціал для інновацій. Нова програма може, наприклад, зосередитися на тому, щоб зробити користувачам простішим і інтуїтивно зрозумілішим додавати фільми до своїх списків, або вона може запропонувати нові підходи до перегляду та перегляду даних фільмів.

По-друге, може виникнути потреба в службі, орієнтованій на певні групи або ніші кіноманів. Нова програма може, наприклад, зосереджуватися на інді-фільмах або міжнародних фільмах або мати на увазі певний віковий діапазон чи демографічну групу.

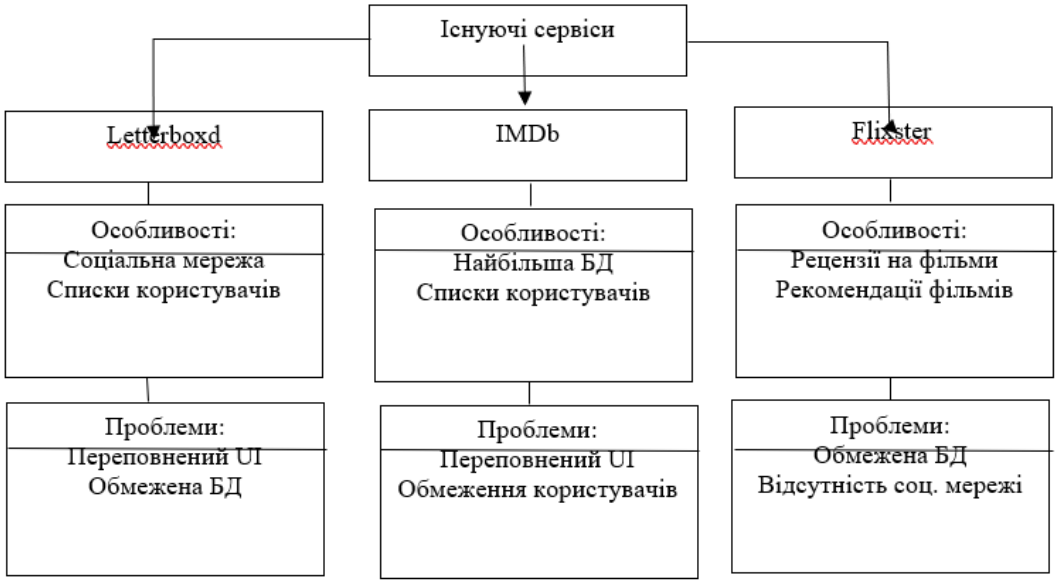


Рис. 1.2 Схема порівняння існуючих сервісів

Існуючі сервіси мають багато проблем, одні з них це:

1) Обмежені можливості налаштування: користувачі можуть не мати достатнього вибору, щоб налаштувати свої списки так, як вони хочуть, за допомогою деяких існуючих служб. Наприклад, вони можуть не дозволяти користувачам створювати унікальні теги або класифікувати фільми за різними критеріями, що може бути обмеженням для деяких користувачів.
2) Низька якість даних: користувачі, які бажають зробити обґрунтований вибір щодо того, які фільми додати до своїх списків, можуть дратувати, оскільки певні постачальники можуть не містити найточнішої чи найновішої інформації про фільми. Крім того, деякі служби можуть бути відсутніми або містити обмежену інформацію про певні фільми, через що користувачам може бути складно складати вичерпні списки.
3) Мінімальні соціальні функції: деякі служби можуть мати соціальні функції, які дозволяють користувачам стежити за іншими користувачами та взаємодіяти з ними, але вони можуть не надавати достатньо функцій, щоб заохочувати змістовні дебати про фільми чи відчуття спільності. Ті, хто бажає спілкуватися з іншими ентузіастами кіно та отримувати рекомендації чи ідеї від інших користувачів, можуть вважати це недоліком.
4) Користувачі можуть не захотіти публічно ділитися своїми вибраними фільмами або захочуть зберегти свої списки в таємниці через проблеми конфіденційності. Наразі доступні служби можуть не забезпечувати належного контролю конфіденційності або не можуть чітко вказати, як користувачі можуть керувати відображенням своїх списків.
5) Залежність від платформи: певні поточні служби можуть бути доступні лише через певні платформи, наприклад мобільні пристрої чи певні веб-браузери. Користувачі, які бажають отримати доступ до своїх списків із кількох пристроїв чи платформ або бажають використовувати різні браузери чи операційні системи, можуть зіткнутися з проблемами.

Загалом, безумовно, є можливості для вдосконалення існуючих сервісів, які допомагають людям створювати власні списки улюблених фільмів. Новий веб-додаток міг би вирішити деякі з цих проблем і надати користувачам більш зручний, настроюваний і соціальний досвід для створення та обміну своїми

списками фільмів.

1.4 Аналіз сучасних вимог до веб-додатків (схема)

Оскільки все більше організацій і окремих осіб почали намагатися бути присутніми в Інтернеті, створення веб-додатків набуло важливості. У результаті поточні вимоги до веб-додатків ускладнюються та відрізняються, що відображає попит на безпечні, прості у використанні та доступні на кількох платформах додатки. У цьому есе ми розглянемо деякі з важливих умов для сучасних веб-програм.

Безпека

Безпека є однією з найважливіших проблем та вимог сучасних онлайн-додатків. Конфіденційні дані користувачів мають бути захищені від несанкціонованого доступу, порушень даних і кібератак у світлі зростання кіберзагроз. Заходи безпеки для веб-додатків включають шифрування, безпечні процедури входу, безпечні платіжні шлюзи та системи для резервного копіювання та відновлення даних. Щоб зменшити небезпеку вразливості, розробники також повинні переконатися, що їхні програми мають найновіші оновлення безпеки та протоколи. [3]

UI інтерфейс

Зручний інтерфейс, який пропонує споживачам природні та зручні умови роботи, є ще одним важливим критерієм для сучасних онлайн-додатків. Розробники програм повинні пам’ятати про користувачів, створюючи свої продукти, пам’ятаючи про такі речі, як зручна навігація, швидке завантаження та адаптивний дизайн.

1) Домашня сторінка: домашня сторінка веб-програми має мати зрозумілий і простий дизайн, який дозволяє користувачам легко переходити до різних розділів програми. Він повинен містити панель пошуку, кнопку входу/реєстрації та кнопку доступу до збережених фільмів користувача.
2) Сторінка реєстрації/входу: Сторінки реєстрації та входу мають бути простими у використанні, з чіткими інструкціями для користувачів щодо створення облікового запису або входу за допомогою існуючого облікового

запису. Сторінка має містити поля для імені користувача, адреси електронної пошти та пароля.

3) Пошук фільмів: функція пошуку фільмів повинна дозволяти користувачам шукати фільми за назвою, режисером, актором або жанром. Результати пошуку мають відображати мініатюру плаката фільму, назву фільму та рік його виходу. Користувачі повинні мати можливість натиснути на фільм, щоб переглянути докладнішу інформацію про нього.

4) Додати до вибраного: користувачі повинні мати можливість додавати фільм до свого списку вибраного зі сторінки інформації про фільм. Кнопка «Додати до вибраного» має бути помітною, і натискання на неї повинно негайно додати фільм до списку вибраного користувача.

Доступність

Іншою важливою вимогою для сучасних онлайн-додатків є доступність, яка гарантує, що люди з вадами можуть легко отримати доступ до програми та використовувати її. Щоб гарантувати, що їхні додатки доступні для людей з обмеженими можливостями, розробники повинні дотримуватися стандартів веб-доступності, наприклад тих, що встановлені в Рекомендаціях щодо доступності веб-вмісту (WCAG). Це передбачає включення клавіатурної навігації в дизайн програми, використання розбірливих і чітких шрифтів і пропонування альтернативного тексту для фотографій.

Продуктивність

Іншою важливою необхідністю сучасної веб-розробки є продуктивність веб-додатків. Розробники повинні переконатися, що їхні додатки оптимізовані для швидкості та ефективності, оскільки споживачі потребують швидших і чуйніших додатків. Це передбачає зменшення розміру сторінки, скорочення часу завантаження та покращення часу відповіді сервера. Щоб належним чином масштабувати свої додатки відповідно до зростаючої кількості користувачів, розробники також повинні гарантувати, що їхні програми можуть обробляти великі рівні трафіку. [17]

Пристойне програмне забезпечення для списку фільмів має бути зручним та інтуїтивно зрозумілим, мати чіткі інструкції та легку навігацію. Користувачі повинні мати можливість швидко й просто додавати фільми до свого списку,

змінювати свій список і шукати певні фільми.

Іншим аспектом, який може вплинути на попит на продуктивність, є якість і повнота бази даних фільмів. Додаток має містити величезну актуальну бібліотеку фільмів для вибору з повною інформацією про кожен фільм, як-от режисер, акторський склад, дата виходу та жанр.

Крім того, додаток має містити інструменти, які допомагають користувачам упорядковувати та фільтрувати списки фільмів, наприклад опцію класифікувати фільми за жанром, датою чи рейтингом.

Нарешті, додаток має бути розроблено таким чином, щоб підтримувати продуктивність і мотивацію. Це може включати такі елементи, як гейміфікація, соціальний обмін і призи за виконання завдань або досягнення цілей. Загалом, добре продумана програма зі списком фільмів може допомогти користувачам бути більш продуктивними та ефективними у створенні й організації своїх улюблених фільмів. Забезпечуючи зручний інтерфейс, повну базу даних і корисні можливості категоризації та сортування, такий додаток може допомогти користувачам заощадити час і енергію, а також отримати задоволення від процесу пошуку та курування власних колекцій фільмів.

Масштабованість

Сучасні онлайн-програми також повинні бути масштабованими, особливо якщо вони призначені для інтенсивного трафіку або значної кількості користувачів. Для того, щоб ефективно масштабувати веб-додатки та задовольняти зростаючі вимоги користувачів без шкоди для безпеки чи швидкості, це необхідно. Це передбачає створення додатків, які можуть обробляти великі обсяги трафіку, розподіляючи навантаження між кількома серверами та впроваджуючи методи кешування, щоб зменшити навантаження на сервери.

Інтеграція

Плавна інтеграція сучасних веб-додатків з іншими системами та службами також є обов’язковою. Щоб надати користувачам повніший досвід, це передбачає підключення до сторонніх API, баз даних та інших онлайн-програм. Сумісність додатків розробників з різними апаратними засобами та операційними системами.

Управління даними

Іншою важливою необхідністю сучасних онлайн-додатків, особливо тих, які обробляють конфіденційні дані користувача, є керування даними. Розробники повинні використовувати методи керування даними, які гарантують безпеку, конфіденційність і правильність даних користувача. Щоб гарантувати відновлення даних у разі порушення даних або збою системи, це включає застосування шифрування даних, використання методів безпечного зберігання даних і створення стратегій резервного копіювання та відновлення даних.

Відповідність

Залежно від сектора та бази користувачів веб-програми також мають відповідати низці законів і стандартів. Розробники повинні переконатися, що їхні програми відповідають чинному законодавству, наприклад Стандарту безпеки даних індустрії платіжних карток і Загальному регламенту захисту даних (GDPR) (PCI DSS). Додатковими критеріями відповідності є інструкції щодо веб-доступності та забезпечення того, що програму можуть використовувати люди з вадами.

Підсумовуючи, існує потреба в безпечних, зручних, доступних і високопродуктивних програмах, що відображається в різноманітності та складності поточних потреб веб-програм. Щоб виправдати очікування сучасних споживачів, розробники повинні враховувати ці фактори під час розробки та створення онлайн-програм.

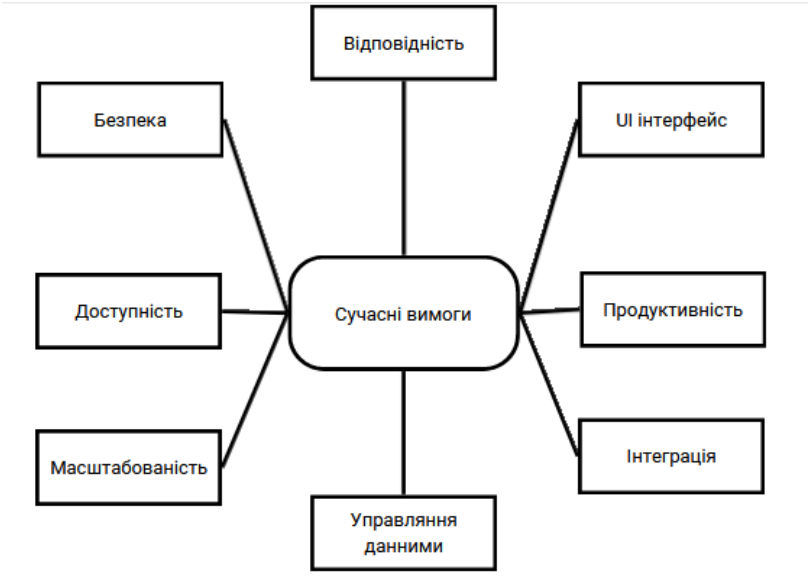


Рис. 1.3 Схема сучасний вимог до веб-додатків

1.5 Аналіз особливостей веб-додатка (схема)

Для любителів кіно, які бажають стежити за своїми улюбленими фільмами, онлайн-програма, яка дозволяє користувачам створювати власні списки улюблених фільмів, є чудовим ресурсом. У цьому есе ми розглянемо характеристики однієї такої онлайн-програми та розглянемо, як вона може забезпечувати споживачам зручну та приємну роботу.

Керування обліковими записами для користувачів

Керування обліковими записами користувачів – це початкова функція веб-додатку для складання списку улюблених фільмів. Користувачі повинні мати можливість реєструватися, входити в систему та керувати даними свого облікового запису. Це включає в себе можливість змінювати налаштування конфіденційності, пароль і інформацію профілю. [5]

Можливість керувати обліковими записами користувачів є ключовим компонентом кожної онлайн-програми. Створення облікових записів для користувачів повинно бути швидким і простим, в ідеалі з можливістю входу в соціальні мережі. Після створення облікового запису вони повинні мати можливість входити в систему та керувати даними свого облікового запису. Це включає можливість змінювати свій пароль, контролювати налаштування конфіденційності та оновлювати інформацію свого профілю, наприклад ім’я та адресу електронної пошти.

Управління обліковим записом користувача є важливим елементом будь-якого онлайн-застосування, оскільки дозволяє користувачам створювати обліковий запис і зберігати інформацію про свій профіль. Процедура створення облікового запису повинна бути швидкою і простою, з можливістю використовувати вхід в соціальну мережу для прискорення процесу. Після входу в систему користувачі повинні мати можливість змінювати інформацію про профіль, таку як їх ім'я, адреса електронної пошти та зображення профілю.

Функціональність Пошуку

Онлайн-додаток для складання списку улюблених фільмів також повинен мати можливість пошуку. Користувачі повинні мати можливість пошуку фільмів за назвою, жанром, режисером або актором. Результати пошуку мають містити відповідні дані про кожен фільм, такі як назва, короткий зміст, рейтинг і рік випуску.

Функція пошуку — ще одна важлива функція веб-програми для створення списку улюблених фільмів. Користувачі повинні мати можливість шукати фільми за назвою, жанром, режисером або актором. Результати пошуку мають відображати релевантну інформацію про кожен фільм, таку як назва, рейтинг, рік випуску та короткий опис. Користувачі також повинні мати можливість фільтрувати результати пошуку на основі різних критеріїв, таких як рейтинг або рік випуску, щоб допомогти їм знайти фільми, які їх цікавлять.

Фунціонал оцінки та перегляду

Додавання системи оцінювання та перегляду є ще одним елементом, який може покращити взаємодію з користувачем. Користувачі повинні мати можливість ранжувати фільми, які вони переглянули, і надсилати рецензії зі своїми думками. Користувачі повинні мати можливість оцінювати фільми за шкалою від 1 до 10, а рецензії мають мати обмеження кількості символів, щоб вони були короткими та по суті.

Система оцінювання та рецензування може покращити роботу веб-програми для створення списку улюблених фільмів. Користувачі повинні мати можливість оцінити фільми, які вони бачили, і залишити відгук зі своїми думками про фільм. Система оцінювання повинна дозволяти користувачам оцінювати фільми за шкалою від 1 до 10, а система рецензування має мати обмеження на кількість символів, щоб огляди були лаконічними та зосередженими. Користувачі також повинні мати можливість переглядати середню оцінку та читати відгуки інших користувачів.

Рис. 1.4 Схема особливостей веб-додатку

1.6 Постановка задачі

Цей проект має на меті створити веб-програму, яка дозволяє користувачам керувати власним унікальним списком улюблених фільмів. Користувачі повинні мати можливість шукати фільми, додавати їх до своїх списків і, якщо потрібно, видаляти їх за допомогою зручного інтерфейсу програми. Програмне забезпечення також має надавати параметри для категоризації та фільтрації списку відповідно до інших факторів, таких як жанр, рейтинг і рік випуску. Мета полягає в тому, щоб створити користувальницький інтерфейс (UI), який є чуйним та інтуїтивно зрозумілим, здатним керувати величезними обсягами відеоданих і може запропонувати споживачам зручну роботу. Щоб гарантувати, що дані користувача зберігаються та безпечно доступні, програма також має містити потужні можливості керування даними та безпеки.
Любителі кіно, які бажають відстежувати всі свої улюблені фільми в одному місці, повинні обслуговуватися онлайн-програмою. Усі, хто має підключення до Інтернету, мають мати змогу користуватися програмою, і вона має працювати з різними веб-переглядачами.

Для цього програма має використовувати поточні інструменти веб-розробки, як-от HTML, CSS, JavaScript, і серверну мову програмування, як-от Python. Щоб отримати інформацію про фільм, як-от назву, опис, жанр, режисера, актора та оцінки користувачів, програма також має використовувати

відомі бази даних фільмів, як-от IMDb або The Movie Database (TMDb). [6]

Функції програми мають дозволити користувачам реєструвати обліковий запис, входити в систему та зберігати список улюблених фільмів на багатьох пристроях. Крім того, програма повинна мати функцію пошуку, щоб користувачі могли швидко знаходити певні фільми та отримувати рекомендації на основі їхньої інформації.

Програма має бути адаптивною, тобто працювати на ПК, планшетах і мобільних пристроях будь-якого розміру. Він також повинен мати зручний і зрозумілий інтерфейс.

Додаток також може мати інструменти рекомендацій, які надають пропозиції щодо фільмів на основі смаків користувача та історії переглядів, щоб забезпечити більш індивідуальний досвід. Наприклад, програма може запропонувати фільми, які за жанром, стилем або режисером можна порівняти з улюбленими фільмами користувача.

Крім того, соціальні елементи, які дозволяють користувачам надсилати електронною поштою або публікувати свої списки фільмів друзям і родині в соціальних мережах, повинні бути додані в програму для покращення використання. Додаток також може мати систему оцінювання та перегляду, яка дозволить користувачам оцінювати та обговорювати фільми з іншими користувачами. [7]

Додаток має використовувати ефективні механізми перевірки та перевірки даних, щоб гарантувати якість і повноту даних фільму. Наприклад, щоб уникнути помилок і гарантувати збереження лише точних даних, програма повинна перевіряти введені користувачем дані. Щоб переконатися, що дані, які показуються користувачам, є точними, програма також має підтверджувати правдивість даних про фільми, отриманих з інших баз даних. Програма має відповідати стандартним практикам веб-розробки, таким як модульний і багаторазовий код, контроль версій і тестування, щоб гарантувати масштабованість і зручність обслуговування програми. Крім того, додаток має бути створено для вирішення будь-яких проблем із масштабуванням, які можуть виникнути через збільшення даних користувача або трафіку.

Додаток має містити службу підтримки або довідковий центр, який може

запропонувати допомогу та швидко вирішити технічні проблеми, щоб надати підтримку та відповісти на будь-які проблеми, з якими можуть зіткнутися користувачі.

Загалом, є кілька перешкод, які потрібно подолати, щоб створити цю веб-програму, зокрема переконатися, що вона придатна для використання, безпечна, точна, масштабована та придатна для обслуговування. Але за умови належного дизайну та розробки програма може запропонувати кіноманам корисну послугу та допомогти їм знайти та впорядкувати улюблені фільми весело та практично.

1.7 Класифікація фільмів

Бойовик: фільми, у яких багато фізичних навантажень, бойові сцени та сцени, що викликають адреналін. Приклади: «Міцний горішок», «Матриця» та «Термінатор 2: Судний день».

Комедії: фільми, які мають на меті розсмішити людей за допомогою гумору, дотепних діалогів і ситуативної комедії. Приклади включають "Похмілля", "Подружки нареченої" та "Шон мертвих".

Драма: фільми, які зосереджені на реалістичних персонажах і ситуаціях, часто мають справу зі складними емоціями та стосунками. Приклади: «Хрещений батько», «Список Шиндлера» та «Форрест Гамп».

Романтика: фільми, які досліджують теми кохання, стосунків і емоційних зв’язків. Приклади включають "The Notebook", "Notting Hill" і "The Fault in Our Stars".

Трилери: фільми, які створюють напругу та напругу, часто містять кримінальний або таємничий сюжет. Приклади включають «Silence of the Lambs», «Se7en» і «Gone Girl».

Жахи: фільми, які мають на меті налякати й шокувати глядачів за допомогою надприродних, паранормальних або психологічних засобів. Приклади: «Екзорцист», «Сяйво» та «Психо».

Наукова фантастика: фільми, які досліджують футуристичні або

альтернативні реальності, часто залучаючи передові технології, космічні подорожі та позаземне життя. Приклади: «Той, що біжить по лезу», «Термінатор» і «Зоряні війни».

Фентезі: фільми з магічними або міфічними елементами, дія яких часто відбувається у вигаданому світі або заснована на серії книг. Приклади включають «Володар перснів», «Гаррі Поттер» і «Хроніки Нарнії».

Анімація: фільми, у яких використовуються методи анімації, щоб розповісти історію, часто розраховані на молодшу аудиторію, але також можуть зацікавити дорослих. Приклади: «Історія іграшок», «У пошуках Немо» та «Король Лев».

Документальні фільми: фільми, які містять фактичний опис реальних подій, людей або місць, часто з конкретним фокусом або повідомленням. Приклади включають "Фаренгейт 9/11", "Акт вбивства" та "Марш пінгвінів".

Надаючи користувачам можливість класифікувати свої улюблені фільми за цими категоріями, веб-програма може допомогти їм знайти нові фільми, які їм можуть сподобатися, і створити персоналізований список рекомендацій.

2. Розробка інформаційного забезпечення та моделювання БД

2.1 Вибір системи керування базами даних (схема)

Під час розробки веб-проекту, який вимагає постійного зберігання даних, вибір системи керування базами даних (СУБД) є життєво важливим рішенням, яке може значно вплинути на продуктивність і масштабованість програми. Деякі з основних змінних, які слід враховувати при виборі СУБД, це структура даних і складність, масштабованість, безпека, надійність і вартість.
Для нашого веб-додатку, який дозволяє клієнтам створювати власний список улюблених фільмів, ми оцінили кілька різних альтернатив СУБД, зокрема MySQL, PostgreSQL і SQLite. Зрештою, ми вирішили використовувати SQLite з таких причин:

1) Структура даних і складність: для нашої веб-програми потрібна відносно проста структура даних із лише кількома таблицями для зберігання інформації користувача та списку улюблених фільмів. Легка та проста архітектура SQLite робить його добре придатним для такого роду програм.
2) Масштабованість: хоча SQLite може бути не таким масштабованим, як інші варіанти СУБД, такі як MySQL або PostgreSQL, ми визначили, що наша веб-програма навряд чи створить великий обсяг трафіку або потребуватиме величезних обсягів зберігання даних. Тому масштабованість не була критичним фактором у нашому рішенні.
3) Безпека: SQLite пропонує надійні функції безпеки, включаючи шифрування та безпечні протоколи зв’язку. Це було важливим фактором для нашої веб-програми, оскільки ми хочемо забезпечити захист даних користувачів.
4) Надійність: SQLite має перевірену історію надійності та стабільності, завдяки великій спільноті розробників і користувачів, які роблять внесок у його постійний розвиток і підтримку. Це було важливим фактором для нас,

оскільки ми хочемо переконатися, що наш веб-додаток завжди доступний і функціонує належним чином.

5) Вартість: SQLite — це СУБД з відкритим вихідним кодом, що означає, що її можна використовувати та розповсюджувати безкоштовно. Це було ключовим фактором для нашого проекту, оскільки ми маємо обмежені ресурси та хочемо, щоб витрати були якомога нижчими.

6) Простота використання: SQLite відомий своєю простотою використання, що робить його ідеальним вибором для розробників, які тільки починають керувати базами даних. Він має простий та інтуїтивно зрозумілий синтаксис, що означає, що розробники можуть швидко та легко створювати та керувати базами даних без необхідності вивчати багато складних команд. [10]

7) Швидкість: SQLite також відомий своєю швидкістю та продуктивністю, особливо коли йдеться про операції читання. Це пояснюється тим, що SQLite зберігає дані в одному файлі, що означає, що йому не потрібно спілкуватися з окремим сервером кожного разу, коли виконується запит. Це може призвести до швидшого часу відповіді та покращення загальної продуктивності.

8) Цілісність даних: SQLite розроблено як самодостатню СУБД, що означає, що вона забезпечує цілісність даних, виконуючи перевірки та підтвердження даних під час їх вставлення чи оновлення. Це може допомогти запобігти пошкодженню даних і забезпечити точність і послідовність даних у базі даних. Загалом ми виявили, що SQLite є ідеальним вибором для нашого веб-проекту через його простоту, безпеку, стабільність і економічну ефективність. Хоча це може бути не найбільш масштабоване рішення, ми вважаємо, що воно більш ніж підходить для наших цілей і дозволить нам створити високоякісну взаємодію з користувачем. Ми обрали SQLite як СУБД для нашої веб-програми, тому що вона пропонувала просте та легке рішення, яке відповідало нашим потребам у зберіганні та управлінні даними. Він забезпечував надійні функції безпеки, був надійним і стабільним, а також забезпечував відмінну продуктивність і портативність. Крім того, простота використання та сильна підтримка спільноти зробили його привабливим варіантом для нашої команди розробників. [11]

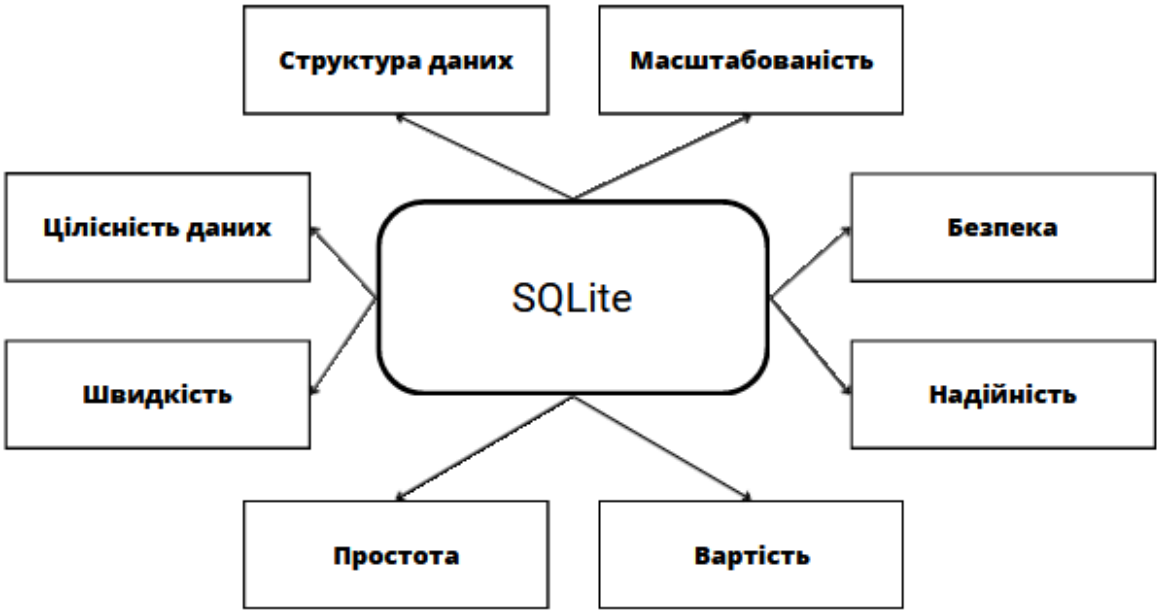


Рис. 2.1 Схема СУБД

2.2 Вибір фреймворку для створення UI (схема)

Вибір фреймворку для створення інтерфейсу користувача може бути важким завданням, оскільки на ринку є так багато можливостей. Коли ми створювали наш веб-додаток, який дозволяє людям створювати власний список улюблених фільмів, нам довелося врахувати численні критерії, перш ніж вибрати структуру. Ось деякі важливі аспекти, які ми врахували:

1) Ефективність. Наш додаток мав бути швидким і чуйним, навіть якщо він мав справу з великими обсягами даних. Нам потрібен був фреймворк, який міг би впоратися з цим, не сповільнюючи роботу програми.
2) Масштабованість: ми хотіли переконатися, що наша програма може обслуговувати все більшу кількість користувачів і функцій. Нам потрібна була масштабована структура, яка могла б легко вмістити нові функції.
3) Досвід розробника: ми хотіли структуру, з якою було б легко працювати, мати гарну документацію та сильну спільноту. Це забезпечить ефективну роботу наших розробників і отримання допомоги за потреби.

Після розгляду всіх цих факторів ми вирішили вибрати React як наш фреймворк. Ось деякі з причин, чому ми вибрали React:

1) Продуктивність: React відомий своєю високою продуктивністю завдяки своїй віртуальній DOM. Це означає, що програма може швидко оновлювати та відтворювати зміни, навіть якщо мається на увазі велика кількість даних.
2) Масштабованість: модульна архітектура React дозволяє легко масштабувати додаток і додавати нові функції. Це також полегшує підтримку кодової бази в міру зростання програми. [9]
3) Досвід розробника: React має велику й активну спільноту з великою кількістю доступних ресурсів і документації. Він також має велику екосистему бібліотек і інструментів, які роблять розробку швидшою та легшою.
4) Архітектура на основі компонентів: Архітектура на основі компонентів React дозволяє легко розбивати складні інтерфейси користувача на менші компоненти, які можна багаторазово використовувати. Це не тільки робить розробку швидшою та легшою, але й полегшує підтримку та оновлення програми.
5) Односторонній потік даних: React слідує односторонньому потоку даних, що означає, що дані протікають в одному напрямку від батьківських компонентів до дочірніх компонентів. Це полегшує налагодження та керування станом програми. [14]
6) Екосистема React: React має велику екосистему бібліотек та інструментів, які роблять розробку швидшою та легшою. Наприклад, ми використовували Redux для управління станом і Axios для викликів API.
7) Сильна спільнота: React має потужну та активну спільноту, яка надає багато ресурсів, підтримки та оновлень. Це гарантує, що фреймворк залишається актуальним і актуальним, а розробники завжди мають доступ до останніх функцій і найкращих практик.

Підсумовуючи, React був найкращим вибором для нашої веб-програми, яка допомагає людям створювати власний список улюблених фільмів, завдяки її високій продуктивності, масштабованості, досвіду розробників, компонентній архітектурі, односторонньому потоку даних, підтримці мобільної розробки, сильній екосистеми та активної спільноти. Ці фактори

дозволили нам створити високоякісну, адаптивну та придатну для обслуговування програму, яка забезпечує безперебійну роботу як на веб-, так і на мобільних платформах. [15]

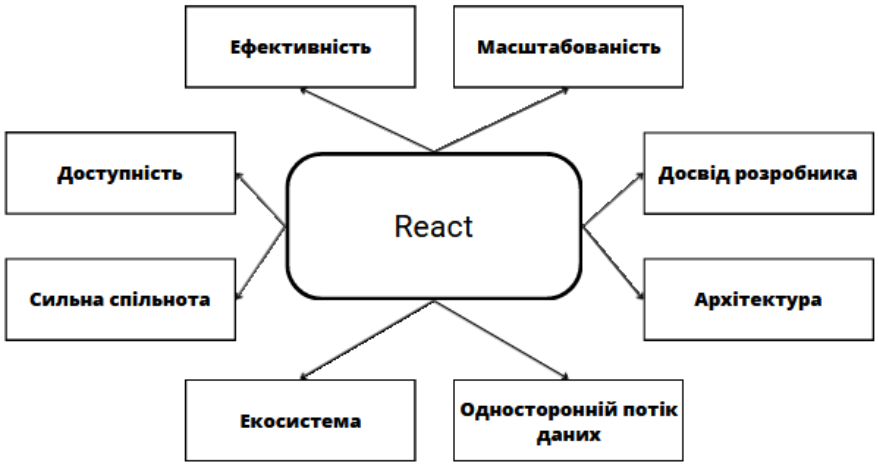


Рис. 2.2 Схема фреймворку UI

2.3 Побудова принципової схеми роботи веб-додатка

Діаграма ілюструє взаємодію між зовнішнім (на стороні клієнта) і серверним (на стороні сервера) компонентами веб-додатку, а також взаємодію з веб-сервером і базою даних.

Ось розбивка компонентів:

Інтерфейс (на стороні клієнта):

Інтерфейс користувача: частина програми, з якою взаємодіють користувачі.

Надсилає HTTP-запити до серверної частини.

Обробляє введені користувачем дані та відображає інформацію. [16]

Сервер (на стороні сервера):

Веб-сервер: обробляє вхідні HTTP-запити та направляє їх до відповідного серверного коду.

Містить логіку програми та обробку.

Зв’язується з базою даних для зберігання та пошуку даних.

Інтерфейс користувача:

Візуальне представлення програми, з якою взаємодіють користувачі.

Дозволяє користувачам переглядати улюблені фільми, додавати нові фільми, видаляти фільми та виконувати інші дії. [12]

Веб-сервер:

Отримує HTTP-запити від інтерфейсу.

Направляє запити до відповідного серверного коду для обробки.

Надсилає HTTP-відповіді назад у зовнішній інтерфейс.

База даних:

Зберігає дані про улюблені фільми користувачів.

Підтримує такі операції, як створення, читання, оновлення та видалення записів фільмів.

Зазвичай комунікаційний потік відбувається в такій послідовності:

Користувач взаємодіє з інтерфейсом користувача (frontend).

Інтерфейс надсилає запит HTTP на веб-сервер (бекенд).

Веб-сервер обробляє запит, взаємодіє з базою даних, якщо це необхідно, і надсилає відповідь HTTP назад інтерфейсу.

Інтерфейс отримує відповідь і відповідно оновлює інтерфейс користувача. На цій схематичній діаграмі представлено загальний огляд компонентів, задіяних у веб-програмі для керування улюбленими фільмами. Варто зазначити, що фактичні деталі реалізації можуть відрізнятися залежно від технологій і фреймворків, які використовуються.

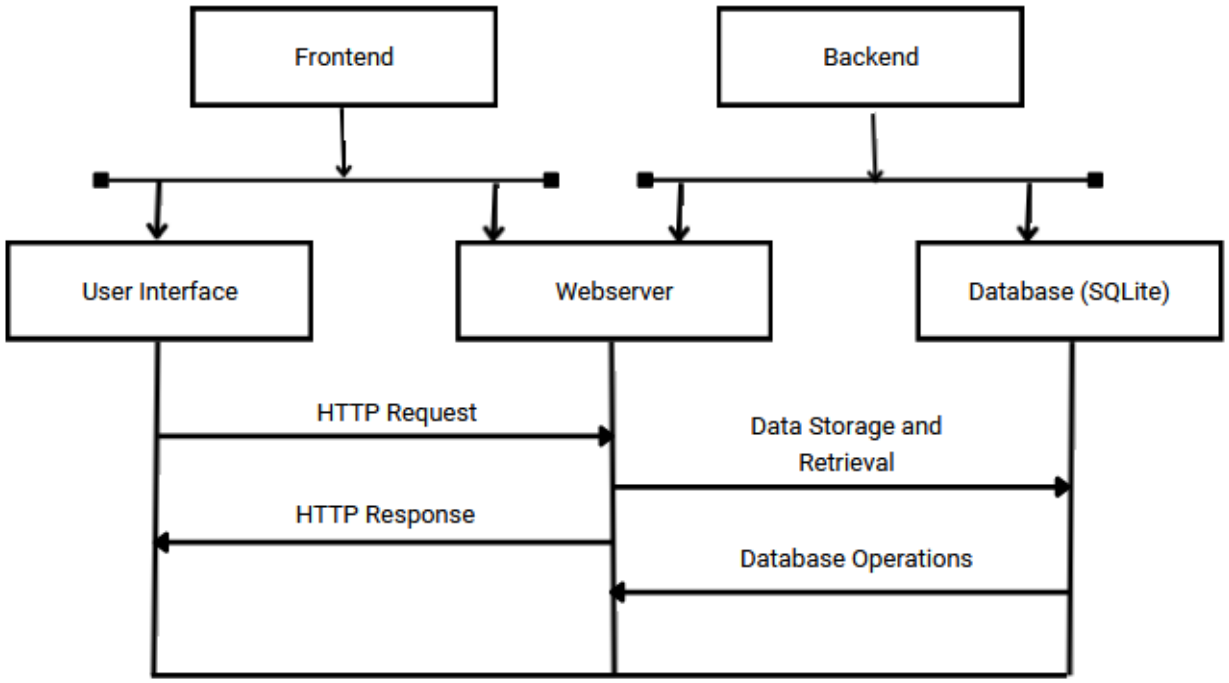


Рис. 2.3 Схема роботи веб-додатка

2.4 Опис UI інтерфейсу веб-додатка

Інтерфейс користувача (UI) веб-додатку розроблений, щоб забезпечити інтуїтивно зрозумілий і зручний досвід для користувачів, щоб керувати своїми улюбленими фільмами. Він містить різні елементи та функції, які дозволяють користувачам створювати, переглядати, редагувати та видаляти фільми зі свого списку. [16]

Заголовок:

1) Розділ заголовка зазвичай з’являється у верхній частині інтерфейсу користувача та містить логотип програми разом із навігаційним меню.
2) Логотип служить елементом брендування програми.
3) Навігаційне меню дозволяє користувачам отримувати доступ до різних розділів програми, таких як домашня сторінка, список вибраного, функція пошуку та налаштування облікового запису.

Домашня сторінка:

1) Домашня сторінка служить цільовою сторінкою програми та містить короткий огляд її функцій.

2) Він може включати вітальне повідомлення, опис мети програми та кнопку із закликом до дії, щоб спонукати користувачів створити свій список улюблених фільмів.

Список обраних:

1) У цьому розділі відображається наявний список улюблених фільмів користувача.
2) Кожен запис фільму зазвичай містить назву фільму, постер або мініатюру, а також додаткові відомості, такі як рік випуску, жанр і рейтинг.
3) Список може бути представлений у вигляді сітки або у вигляді прокручування, залежно від кількості фільмів.
4) Користувачі можуть переглядати список, щоб знайти певні фільми.

Додати фільм:

1) Ця функція дозволяє користувачам додавати нові фільми до списку вибраного.
2) Зазвичай він містить форму з полями для введення назви фільму, року випуску, жанру та будь-якої іншої відповідної інформації.
3) Користувачі можуть вводити деталі вручну або використовувати функцію пошуку, щоб отримати інформацію про фільм із зовнішньої бази даних або API.
4) Після введення даних користувачі можуть надіслати форму, щоб додати фільм до свого списку вибраного.

Редагування фільму:

1) Ця функція дозволяє користувачам редагувати деталі наявного фільму у своєму списку вибраного.
2) Вибравши окремий запис про фільм, користувачі отримають доступ до форми або модального вікна, де вони можуть змінити інформацію про фільм.
3) Форма попередньо заповнюється наявними деталями, що дозволяє користувачам вносити бажані зміни.
4) Користувачі можуть зберегти оновлену інформацію, щоб застосувати зміни до запису фільму.

Видалити фільм:

1) Ця функція дозволяє користувачам видаляти фільм зі списку вибраного.

2) Користувачі зазвичай можуть знайти опцію видалення, пов’язану з кожним записом фільму.
3) Після підтвердження фільм остаточно видаляється зі списку.

Функціональність пошуку:

1) Веб-програма може надавати функцію пошуку, яка дозволяє користувачам шукати певні фільми.
2) Користувачі можуть вводити ключові слова, наприклад назви фільмів або імена акторів, щоб знаходити відповідні фільми.
3) Результати пошуку можуть відображатися в окремому розділі або інтегруватися в список обраних.

Налаштування аккаунта:

1) У цьому розділі користувачі можуть керувати параметрами облікового запису.
2) Користувачі можуть оновлювати інформацію свого профілю, змінювати паролі та налаштовувати налаштування сповіщень.
3) Він також може включати параметри імпорту або експорту списку вибраного, що дозволяє користувачам створювати резервні копії своїх даних або переносити їх на інший пристрій.

Адаптивний дизайн:

1) Інтерфейс веб-додатку розроблений таким чином, щоб бути чуйним, забезпечуючи оптимальну взаємодію з користувачем на різних пристроях і розмірах екрана.
2) Він адаптує та налаштовує макет, розмір шрифту та інтерактивні елементи, щоб забезпечити безперебійну роботу на настільних комп’ютерах, планшетах і смартфонах.

Загалом інтерфейс користувача веб-додатку має на меті забезпечити естетично привабливе та інтуїтивно зрозуміле середовище для створення та керування списком улюблених фільмів. Він зосереджений на простоті, зручності використання та ефективній навігації для покращення загального досвіду користувача [17]

UseCase діаграмма:

Переглянути улюблені фільми: Дозволяє користувачеві переглядати

список улюблених фільмів. Цей варіант використання дозволяє користувачеві бачити фільми, які він додав до своїх улюблених.

Керування фільмами: Охоплює варіанти використання, пов'язані з керуванням фільмами у списку обраного користувача. Включає наступні підваріанти використання:

Додати фільм: Дозволяє користувачеві додати новий фільм до списку вибраного.

Редагувати фільм: Дозволяє користувачеві змінювати інформацію про фільм, що вже є у його списку обраного.

Видалити фільм: Дозволяє користувачеві видалити фільм зі списку вибраного.

Пошук фільмів: Дозволяє користувачеві шукати певні фільми. Цей варіант використання дозволяє користувачеві вводити ключові слова і шукати фільми за назвами, акторами або іншими відповідними критеріями.

Кожен з цих варіантів використання представляє певну взаємодію або функціональність, що надається користувацьким інтерфейсом веб-додатку. Діаграма варіантів використання дає загальний огляд основних дій, які користувачі можуть виконувати в додатку, полегшуючи розуміння ключових функцій і взаємодій, доступних для них.

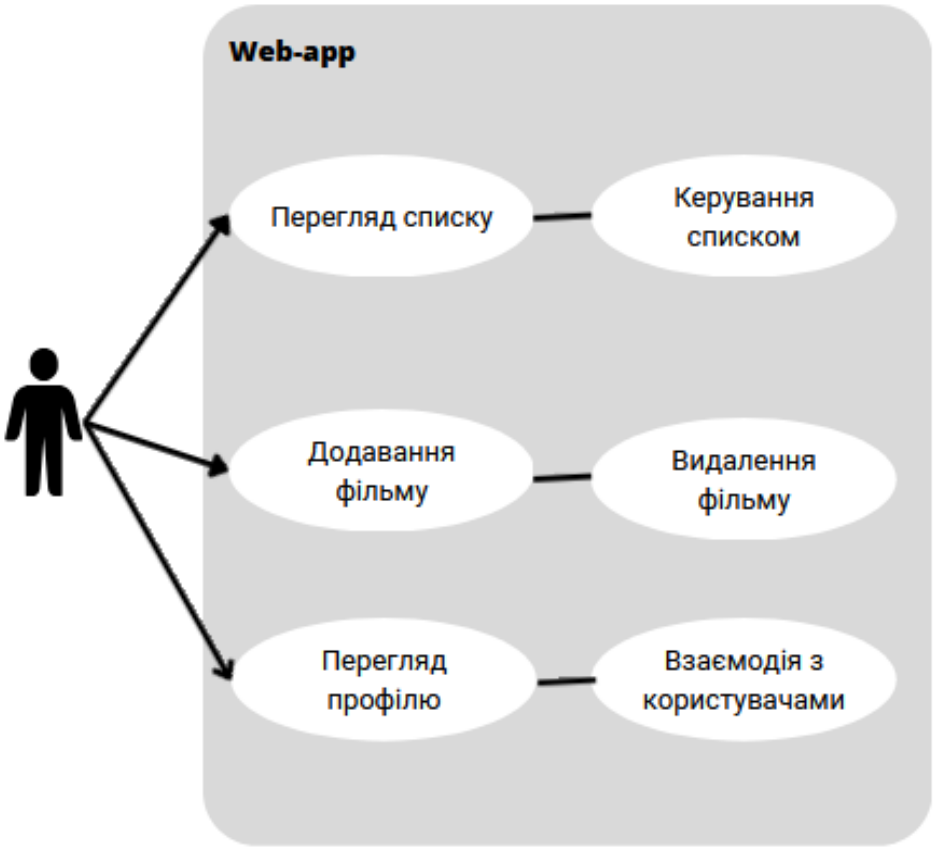


Рис. 2.4 Usecase діаграмма UI інтерфейсу

2.5 *Концептуальна модель БД*

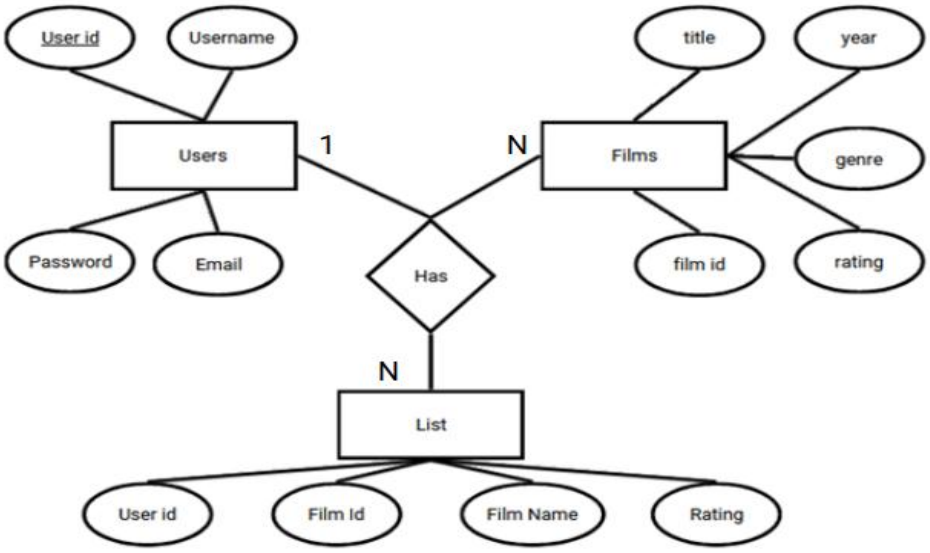


Рис. 2.3 Діаграма Чена концептуальної моделі БД

Пояснення моделі бази даних:

Концептуальна модель бази даних складається з трьох основних сутностей:

Користувач, Фільм та список. Ось розбивка кожної сутності:

Користувач:

1) Представляє користувача веб-додатку.
2) Містить такі атрибути, як user_id (первинний ключ), ім'я користувача, електронна пошта, пароль і профіль_зображення.
3) Атрибут user_id унікально ідентифікує кожного користувача в базі даних.
4) Один користувач може мати декілька обраних (зв'язок "один-до-багатьох" з сутністю "Обраний").

Фільм:

1) Представляє фільм у додатку.
2) Містить такі атрибути, як film_id (первинний ключ), назва, рік випуску, жанр та рейтинг.
3) Атрибут film_id унікально ідентифікує кожен фільм у базі даних.
4) Кожен фільм може бути пов'язаний з кількома користувачами через обрані

Список:

5) Представляє фільм у додатку.
6) Містить такі атрибути, як list_id, user_id (первинний ключ), назва, рік випуску, жанр та рейтинг.
7) Атрибут list_id унікально ідентифікує кожен фільм у базі даних.
8) Кожен список пов'язаний з одним користувачем

2.6 Логічна Модель БД:

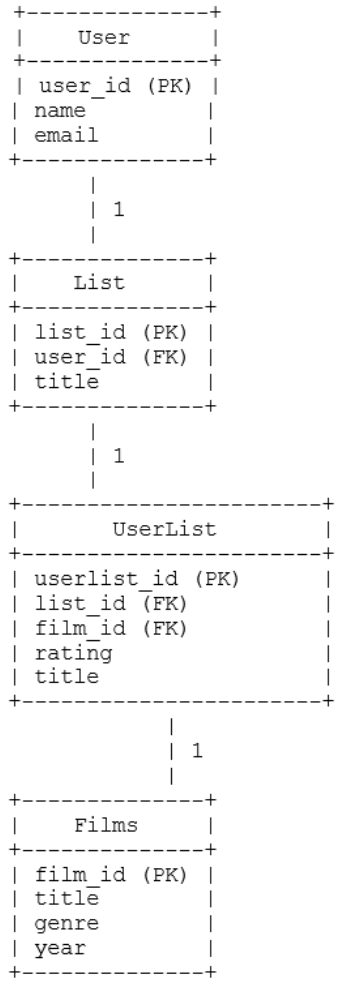


Рис. 2.4 Логічна модель БД

Пояснення логічної моделі:

Сутність User представляє користувачів додатку. Вона містить такі атрибути, як UserID (унікальний ідентифікатор), ім'я користувача, електронна пошта та пароль.

Сутність Film представляє окремі фільми. Вона містить такі атрибути, як FilmID (унікальний ідентифікатор), Назва, Режисер, Рік випуску та Тривалість.

Сутність List представляє списки улюблених фільмів, створених користувачами. Вона містить такі атрибути, як ListID (унікальний ідентифікатор), UserID (зовнішній ключ, що посилається на User.UserID) і Title.

Зв'язок між сутностями представлено лініями, що з'єднують пов'язані сутності. У цьому випадку між Користувачем і Списком існує зв'язок один-до-

багатьох (один користувач може мати кілька списків), на що вказує лінія, що з'єднує Користувача зі Списком. Також існує зв'язок "багато до багатьох" між Списком і Фільмом (список може містити кілька фільмів, а фільм може бути в кількох списках), позначений лінією, що з'єднує Список і Фільм.

Ця логічна модель надає огляд сутностей та їхніх зв'язків у базі даних. Вона допомагає візуалізувати структуру бази даних, не зосереджуючись на конкретних деталях реалізації або типах даних. [20]

2.7 Фізична модель БД:

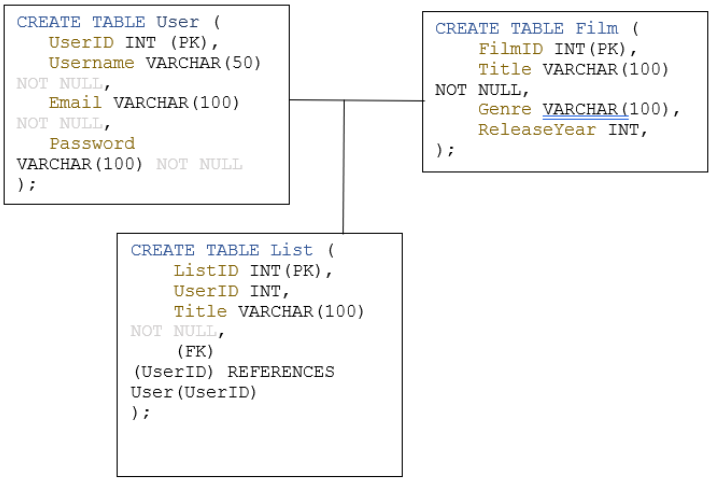


Рис.2.5 Схема фізичної моделі БД

Пояснення фізичної моделі:

Таблиця User створюється зі стовпчиками для UserID (первинний ключ), імені користувача, адреси електронної пошти та пароля. Ідентифікатор користувача визначено як тип даних INT, а інші стовпці - як VARCHAR з відповідною довжиною. Для забезпечення цілісності даних застосовуються обмеження, такі як NOT NULL.

Таблиця Film створюється зі стовпчиками для FilmID (первинний ключ), назви, режисера, року випуску та тривалості. Ідентифікатор фільму має тип INT, а інші стовпці мають типи даних VARCHAR або INT відповідно.

Таблицю List створено зі стовпцями ListID (первинний ключ), UserID (зовнішній ключ, що посилається на User.UserID) і Title. Стовпці ListID та UserID визначено як типи даних INT, а стовпець Title - як VARCHAR

Ця фізична модель представляє деталі реалізації бази даних за допомогою

синтаксису SQL. Вона включає конкретні типи даних, первинні ключі, зовнішні ключі та зв'язки, необхідні для створення та управління базою даних для веб-додатку, орієнтованого на створення списків улюблених фільмів.

2.8 Цілістність та захист БД

Перевірка даних:

Впровадьте належні механізми перевірки даних, щоб гарантувати, що до бази даних вносяться лише достовірні та очікувані дані. Це включає перевірку даних, введених користувачем у такі поля, як ім'я користувача, адреса електронної пошти, пароль, інформація про фільм тощо.

Використовуйте перевірку на стороні сервера та на стороні клієнта для виявлення та запобігання збереженню в базі даних зловмисних або некоректних даних. [10]

Застосовуйте обмеження, такі як перевірка типу даних, перевірка довжини та формату, щоб забезпечити цілісність та узгодженість збережених даних.

Безпечна автентифікація та авторизація:

Впроваджуйте безпечні механізми автентифікації, такі як хешування та збереження паролів, для захисту облікових даних користувачів, що зберігаються в базі даних.

Впроваджуйте надійні політики паролів і заохочуйте користувачів обирати складні паролі.

Впровадьте контроль доступу на основі ролей (RBAC), щоб гарантувати, що лише авторизовані користувачі можуть мати доступ до бази даних та змінювати її.

Обмежте доступ до конфіденційних операцій і даних бази даних лише авторизованим адміністраторам або привілейованим користувачам.

Шифрування:

Розгляньте можливість шифрування конфіденційних даних у базі даних, таких як паролі користувачів, щоб забезпечити додатковий рівень захисту.

Використовуйте методи шифрування, такі як AES (Advanced Encryption Standard), для шифрування та розшифрування конфіденційних даних під час їх

зберігання та отримання з бази даних.

Регулярне резервне копіювання:

Регулярно створюйте резервні копії бази даних для захисту від втрати даних, спричиненої апаратними збоями, помилками програмного забезпечення або порушеннями безпеки.

Зберігайте резервні копії даних у безпечних та окремих місцях, щоб забезпечити їхню доступність та цілісність у разі катастрофи.

Контроль доступу:

Налаштуйте належні механізми контролю доступу, щоб обмежити прямий доступ до бази даних з несанкціонованих джерел.

Використовуйте брандмауери, заходи мережевої безпеки та списки контролю доступу (ACL) для контролю та моніторингу доступу до сервера бази даних.

Використовуйте безпечні протоколи, такі як HTTPS, для передачі даних між веб-додатком і сервером бази даних. [13]

Аудит та ведення журналів бази даних:

Впровадьте механізми аудиту та реєстрації для відстеження та моніторингу дій з базою даних, таких як входи користувачів, модифікації даних та спроби доступу.

Регулярно переглядайте та аналізуйте журнали бази даних, щоб виявити будь-які підозрілі дії або потенційні порушення безпеки.

Ведіть аудиторський журнал, щоб фіксувати всі критичні операції та зміни, внесені до бази даних.

Регулярні оновлення та виправлення:

Оновлюйте програмне забезпечення бази даних та пов'язані з нею компоненти найновішими виправленнями та оновленнями безпеки.

Будьте в курсі вразливостей безпеки та рекомендацій, пов'язаних з програмним забезпеченням бази даних, і оперативно застосовуйте необхідні патчі або виправлення.

Дуже важливо поєднувати ці заходи з іншими кращими практиками безпеки додатків і адміністрування серверів, щоб забезпечити загальний захист і цілісність веб-додатку і пов'язаних з ним компонентів.

3. **Розробка програмного забезпечення. Приклади функціоналу**

3.1 Обгрунтування вибору інструментарію

Обґрунтування вибору React та Django як інструментів для розробки веб-додатку, що допомагає людям створювати список своїх улюблених фільмів, можна підсумувати наступним чином:

React для Front-End розробки: React - це популярна JavaScript-бібліотека для створення користувацьких інтерфейсів. Вона дозволяє створювати високоінтерактивні та чуйні користувацькі інтерфейси, що робить її придатною для динамічних веб-додатків, таких як списки фільмів. Компонентна архітектура React сприяє повторному використанню коду, що полегшує управління та підтримку додатку з часом. Його віртуальний DOM ефективно оновлює та рендерить компоненти, забезпечуючи більш плавний користувацький досвід. [8]

Повторне використання компонентів: Компонентна архітектура React сприяє повторному використанню та модульності. Компоненти можуть бути легко повторно використані у всьому додатку, що скорочує час та зусилля розробки.

Віртуальний DOM: Віртуальний DOM React ефективно оновлює та рендерить лише необхідні компоненти, що призводить до вищої продуктивності у порівнянні з маніпуляціями з реальним DOM. Це призводить до більш плавного користувацького досвіду, особливо при роботі зі складними та інтерактивними елементами інтерфейсу.

Багата екосистема: React має широку екосистему бібліотек та інструментів, таких як Redux для управління станами, React Router для навігації та Material UI для готових компонентів інтерфейсу користувача. Ці ресурси можуть значно пришвидшити розробку та покращити функціональність і естетику веб-додатку. [7]

Django для Back-End розробки: Django - це потужний веб-фреймворк на Python, який відповідає архітектурному шаблону Model-View-Controller

(MVC). Він надає надійний набір інструментів і функцій для швидкої розробки, що робить його добре придатним для створення складних веб-додатків. Django пропонує вбудовану підтримку моделей даних, автентифікації та управління базами даних, які є важливими для створення додатку зі списком фільмів, що дозволяє користувачам зберігати та керувати своїми улюбленими фільмами. Крім того, сильний акцент Django на безпеці та масштабованості гарантує, що додаток може обробляти зростаючу базу користувачів і захищати конфіденційні дані користувачів. [13]

Швидка розробка: Підхід Django "з батарейок" надає повний набір інструментів і функцій, таких як ORM (об'єктно-реляційне відображення) для управління базами даних, система автентифікації та вбудований інтерфейс адміністратора. Це прискорює процес розробки, усуваючи необхідність створювати ці загальні функціональні можливості з нуля.

Синтаксис Python: Django створено за допомогою Python, широко використовуваної та зручної для початківців мови програмування. Чистий і зрозумілий синтаксис Python полегшує розробникам написання та підтримку коду, зменшуючи ймовірність помилок і підвищуючи загальну ефективність розробки.

Безпека: Django за замовчуванням включає численні заходи безпеки, такі як захист від поширених веб-уразливостей, таких як міжсайтовий скриптинг (XSS) та підробка міжсайтових запитів (CSRF). Він також пропонує вбудовані механізми автентифікації та авторизації користувачів, що полегшує реалізацію безпечного керування користувачами у додатку списку фільмів. [20]

Ефективність повного стеку розробки: Поєднуючи React для фронтенд-розробки та Django для бекенд-розробки, ми можемо використовувати сильні сторони обох технологій. Декларативний та компонентний підхід React покращує розробку користувацького інтерфейсу, в той час як комплексний фреймворк Django спрощує роботу на стороні сервера. Таке поєднання забезпечує ефективну розробку та більш плавну інтеграцію між фронтенд- та бекенд-компонентами додатку.

Безшовна інтеграція: React та Django можна легко інтегрувати для створення повностекового веб-додатку. REST-фреймворк Django можна використовувати

для створення надійного API, який взаємодіє з фронтендом React, забезпечуючи ефективний обмін даними та оновлення в реальному часі.

Знайомство розробників: React і Django є широко розповсюдженими і добре задокументованими фреймворками. Багато розробників вже знайомі з однією або обома цими технологіями, що скорочує час навчання і полегшує збір команди розробників для створення веб-додатків.

Підтримка спільноти та екосистема: React та Django мають великі та активні спільноти розробників. Це означає, що існує велика кількість документації, навчальних посібників та підтримки спільноти, що дозволяє легко знаходити рішення спільних проблем та бути в курсі найкращих практик. Широка екосистема, що оточує React та Django, також пропонує широкий спектр сторонніх бібліотек та пакетів, які можна використовувати для покращення функціональності та ефективності веб-додатків.

Масштабованість та продуктивність: Віртуальний DOM React та оптимізація продуктивності Django дозволяють додатку працювати з великою кількістю користувачів та ефективно керувати операціями з базами даних. Обидва фреймворки добре зарекомендували себе у роботі з веб-сайтами та додатками з високим трафіком, гарантуючи, що додаток для перегляду фільмів може масштабуватися відповідно до кількості користувачів без шкоди для продуктивності.

Горизонтальне масштабування: І React, і Django розроблені для роботи з додатками з високим трафіком і можуть бути горизонтально масштабовані на декількох серверах, щоб впоратися зі зростаючим користувацьким навантаженням.

Кешування та оптимізація продуктивності: Django надає механізми кешування, які можуть покращити продуктивність даних, до яких часто звертаються. Крім того, віртуальний DOM React та ефективний рендеринг допомагають мінімізувати непотрібні оновлення, забезпечуючи оптимальну продуктивність веб-додатку.

Таким чином, вибір React та Django як інструментів для розробки веб-додатку забезпечує потужне поєднання інтерактивності інтерфейсу та функціональності бекенду. Сильні сторони кожного фреймворку доповнюють

один одного, в результаті чого створюється ефективний [11]

3.2 Загальна Архітектура та файлова структура

3.2.1 Загальна архітектура:

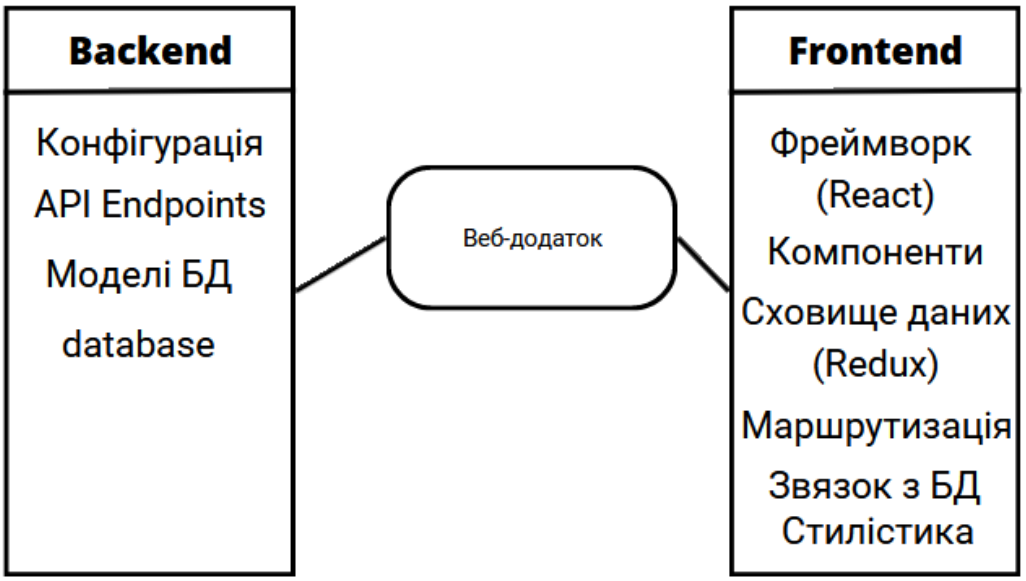


Рис. 3.1 Загальна схема архітектури веб-додатку

Front-End (React):

Компоненти: Створюйте багаторазові компоненти для різних частин інтерфейсу, таких як список фільмів, інформація про фільм, автентифікація користувачів та форми.

Керування станами: Використовуйте бібліотеку управління станами, таку як Redux або React Context API, щоб керувати глобальним станом додатку, включаючи автентифікацію користувача та список його улюблених фільмів.

Інтеграція з API: Спілкуйтеся з внутрішнім API для отримання та надсилання даних за допомогою HTTP-запитів, наприклад, для пошуку фільмів, додавання фільмів до обраного та автентифікації користувачів.

Маршрутизація: Налаштуйте маршрутизацію за допомогою React Router,

щоб керувати навігацією між різними сторінками або поданнями в додатку.

Стилістика: Застосовуйте CSS або використовуйте бібліотеку CSS-in-JS, таку як styled-components, для стилізації компонентів і створення естетично привабливого інтерфейсу. [6]

Back-End (Django):

Моделі: Визначте моделі Django для представлення таких об'єктів, як фільми, користувачі та вибране. Встановіть зв'язки між моделями, наприклад, зв'язок "багато-до-багатьох" між користувачами та обраними фільмами.

Подання: Реалізуйте подання Django, які обробляють вхідні запити, взаємодіють з моделями і повертають відповідні відповіді. Сюди входять подання для пошуку фільмів, керування автентифікацією користувачів та обробки операцій з обраними фільмами.

Конфігурація URL: Визначення шаблонів URL-адрес у конфігурації URL-адрес Django для зіставлення вхідних запитів з відповідними поданнями.

Серіалізатори: Створення серіалізаторів за допомогою фреймворку Django REST для перетворення екземплярів моделі в JSON або інші формати для відповідей API.

Аутентифікація: Реалізуйте автентифікацію та авторизацію користувачів за допомогою вбудованої системи автентифікації Django або сторонніх пакетів, таких як автентифікація за допомогою токенів фреймворку Django REST або JWT (JSON Web Tokens).

База даних: Налаштуйте параметри бази даних у файлі налаштувань Django, вказавши тип бази даних та деталі підключення.

Кінцеві точки API: Створіть кінцеві точки API, які оброблятимуть запити на дані фільмів, автентифікацію користувачів та операції з обраними фільмами. [10]

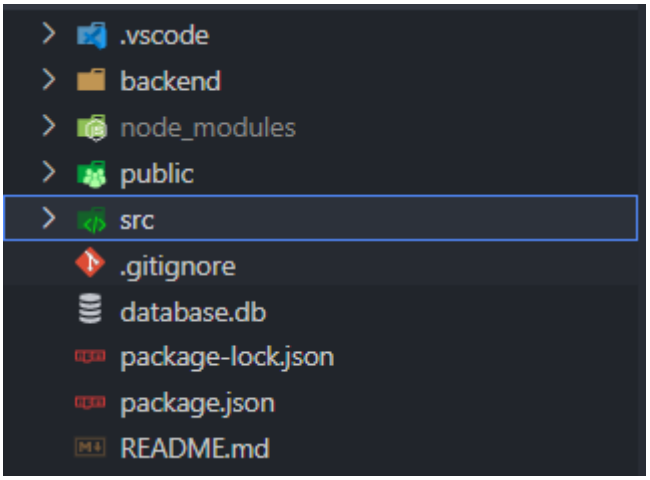
3.2.2 Файлова структура:

Схематична файлова структура:

```
backend/
  |- database.db             (База даних SQLite)
  |- migrations/             (Бекенд міграції)
 |- env.py           (головний файл для міграцій)
  |- venv/                   (допоміжні файли для збірки)
```

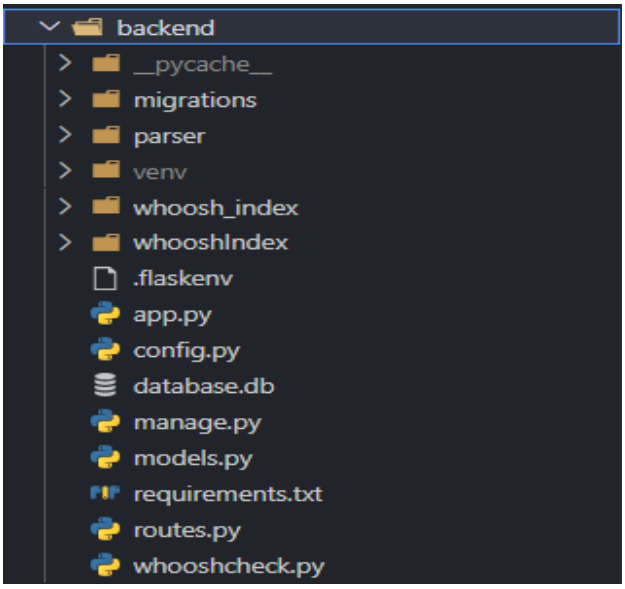
```
  |- parser/                (парсер списку фільмів)
  |- models.py              (Бекенд моделі)
  |- routes.py              (Бекенд маршрутизатор)
  |- app.py                 (кірневий файл бекенду)
  |- menage.py              (збірка проекту)
  |- config.py              (Бекенд конфігурація)
  |- requierements.txt      (Бекенд рекомендації)

Node_modules/
  |- bin/                 (npm модулі)

src/
  |- components/          (React components)
     |- Edit.js           (Компонент редагування фільму)
     |- Film.js           (Компонент фільму)
     |- AddFilm.js        (Компонент додавання фільму)
     |- Friends.js        (Компонент списку друзів користувача)
     |- FriendsSearch.js  (Компонент пошуку користувачів)
     |- Header.js         (Компонент заголовку сторінки)
     |- Modal.js          (Компонент модального вікна)
     |- Profile.js        (Компонент списку користувача)
     |- ProfileBar.js     (Компонент інформації про користувача)
     |- Home.js           (Головна сторінка)
     |- Login.js          (Сторінка Логіну)
     |- Register.js       (Сторінка Регістріції)

  |- redux/               (Бібліотека Redux)
     |- Actions/          (Redux Actions)
 |- filmsActions.js (Actions фільмів)
        |- authActions.js (Actions логіну)
     |- Reducers/         (Redux Reducers)
 |- filmsReducers.js(Reducer фільмів)
        |- authReducers.js (Reducer логіну)
     |- store.js          (Головне сховище даних Redux)

  |- App.js               (Головний компонент React)
  |- index.css            (Головний css React)
  |- setupTests.ts        (Тести проекту)
  |- index.js             (Кірневий файл проекту)
  |- logo.svg             (Логотип)
  |- images/              (Папка зображень)

public/
  |- index.html           (HTML шаблон проекту)
  |- favicon.ico          (Favicon for the app)
  |- assets/              (допоміжні матеріали)

.gitignore               (git файл)
package.json             (npm конфігурація)
package-lock.json        (npm автоматичний конфіг)
database.db              (База даних)
README.md                (документація)
```

Опис файлової структури:

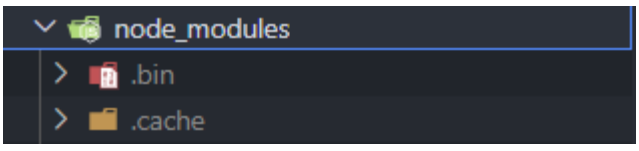


У файловій структурі ми можемо бачити поділення проекту на декілька частин, а саме:

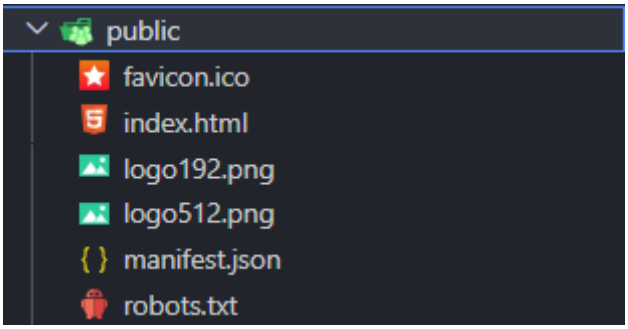
1) Backend: Одна з найголовніших частин проекту де зберігается весь функціонал бекенду та конфігурація підключення до БД



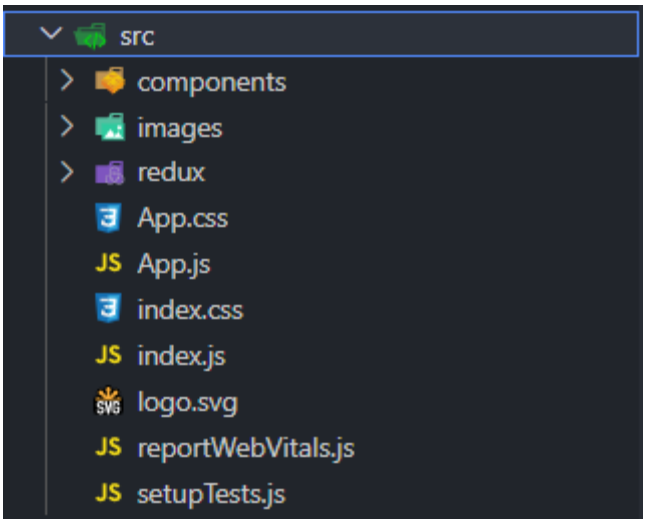
2) Папка node_modules служить збірником всік модулів який втсановлює npm для роботи фреймворку React, ця папка створюється та обновляється автоматично, не залежить від розробника.



3) Public: папка служить як сховище для всіх допоміжних файлів, таких як фотографії, аудіозаписи, логотипи і т.д.



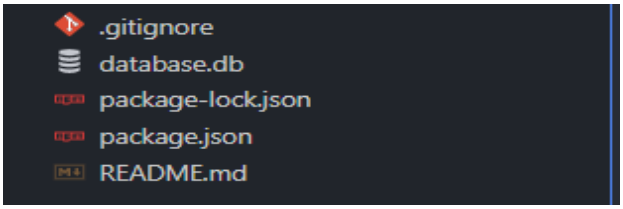
4) Src: Найголовніша папка проекту де зберігается весь функціонал Front-end який пыделючений до папки Backend:



Тут ми можемо бачити внутрішню структуру нашого веб-додатку де знаходятся корневі файли App.js та Index.js а також папки які відповідають за роботу программи: Components та Redux.

Також у нашій программі присутні Unit тести які зберігаються у файлі setupTests.js який автоматично тестує программу на цілістність збірки. (npm build)

5) Інші файли: сотаннє що залишилося в файловій структурі це файли конфігурації нашого дадутку та сама база даних яка знаходится в окремому файлі:



3.3 Структура функціональної частини та опис конкретних функцій

3.3.1 Приклад Backend Функціоналу

У файлі menage.py ми задаємо функціонал створення, оновлення для нашого проекту, це можна побачити на скріншоті:

```
def deploy():
    from app import create_app
    from flask_migrate import upgrade, migrate, init, stamp



    app = create_app()
    app.app_context().push()
    stamp()
    migrate()
    upgrade()


deploy()
```

Приклад створення моделі БД:

Ми можемо бачити як за допомогою оголошення классу User(db.Model) ми створюємо нову таблицю в БД після чого оголошуємо її залежності та поля, тут ми створюємо такі поля: Id, username, password, film, friends це все що потрібно бути в моделі користувача, у кожного користувача буде унікальний id через який можна робити залежність для інших таблиць БД. [10]

Фунція def friend_add() створює залежність для майбутнього функціоналу додавання в друзі.

```python
class User(db.Model):
    __tablename__ = "database"
    __searchable__ = ["username"]
    __analyzer__ = StemmingAnalyzer()
    id = db.Column(db.String(32), primary_key=True, unique=True, default=get_uuid)
    username = db.Column(db.String(22), unique=True, nullable=False)
    password = db.Column(db.Text, nullable=False)
    film = db.relationship('User_List', backref='database', lazy=True)
    friends = db.relationship('User',
                              secondary = friends,
                              primaryjoin = (friends.c.user_id == id),
                              secondaryjoin = (friends.c.friend_id == id),
                              lazy = 'dynamic'
                             )


    def friend_add(self, user):
        if not self.is_friend(user):
            self.friends.append(user)

    def friend_remove(self, user):
        if self.is_friend(user):
            self.friends.remove(user)

    def is_friend(self, user):
        return self.friends.filter(friends.c.friend_id == user.id).count() > 0



class User_List(db.Model):
    __tablename__ = "user_list"
    id = db.Column(db.String(32), primary_key=True, unique=True, default=get_uuid)
    added_at = db.Column(db.DateTime, default=datetime.datetime.now)
    name_ru = db.Column(db.String)
    name_en = db.Column(db.String)
    poster_url = db.Column(db.String)
    user_rating = db.Column(db.Float)
    status = db.Column(db.String)
    user_id = db.Column(db.String, db.ForeignKey('database.id'))

```

Також на скріншоті ми бачимо створення другої таблиці, а саме User_list де ми будуємо список фільмів користувача, як можна спостерігати, поле User_id відносится до таблиці User і зчитує id користувача.

Авторизація:

У нашому веб-додатку використовується сучасна система авторизації з допомогою JWT токена, її використання можна побачити на скріншоті:

```python
app.config['JWT_SECRET_KEY'] = "eyJhbGciOiJIUzI1NiJ9.eyJSb2xl
app.config['JWT_ACCESS_TOKEN_EXPIRES'] = timedelta(hours=4)
```

Тут ми створюємо та зберігаємо токен для авторизації а також задаємо таймер через який ми будемо оновлювати його для максимальної безпеки авторизації.

```python
@app.before_request
def indexing():
    flask_whooshalchemy3.search_index(app, User)
    flask_whooshalchemy3.search_index(app, Animes)



@app.after_request
def refresh_expiring_jwts(response):
    try:
        exp_timestamp = get_jwt()["exp"]
        now = datetime.now(timezone.utc)
        target_timestamp = datetime.timestamp(now + timedelta(hours=12))
        if target_timestamp > exp_timestamp:
            access_token = create_access_token(identity=get_jwt_identity())
            data = response.get_json()
            if type(data) is dict:
                data["access_token"] = access_token
                response.data = json.dumps(data)
        return response
    except (RuntimeError, KeyError):
        # No JWT == return original response
        return response
```

HTTP запити:

Одна з найголовніших частин розробки бекенду це API endpoints, за допомогою яких наш додаток спілкується з базою даних. Всього є декілька видів запитів: GET (отримати данні), POST (відправити данні), PUT (оновити данні) та DELETE (видалити дані)

Розлянемо на прикладі нашого доатку декілька запитів:

1) GET:

```python
@app.route("/profile/<username>", methods=["GET"])
@jwt_required()
def my_profile(username):
    current_user = get_jwt_identity()
    user = User.query.filter_by(username=username).first()
    if user is None:
        return {"Error": "This user doesn`t exist"}, 412
    userlist = User_List.query.filter_by(user_id=user.id).all()
    filmlist = [{"Name": film.name, "Rating": str(film.user_rating), "Status": film.status, "Current_User": current_user} for film in userlist]
    response_body = jsonify(filmlist)

    return response_body
```

Тут ми можемо бачити фунціонал пошуку користувача на шлях “/profile/username” де usernmae це Query параметр який відправляє користувач при пошуку.

Також ми можемо бачити валідацію яка може показати нам помилку у випадку якщо кристувача не знайдено.

При успішному запиті ми відправляємо з бекенду данні які отримує Frontend і вімальовує їх на сторінці. [13]

2) POST:

```python
@app.route('/add', methods=["POST"])
@jwt_required()
def adder():
    username = get_jwt_identity()
    name = request.json.get("name")
    user_rating = request.json.get("user_rating")
    status = request.json.get("status")

    user = User.query.filter_by(username=username).first()

    new_film = User_List(name=name,user_rating=user_rating,status=status,user_id=user.id)

    film_exist = User_List.query.filter_by(name=new_film.name,user_id=new_film.user_id).first() is not None

    if film_exist:
        return jsonify({"error": "You already added this film"}),409

    db.session.add(new_film)
    db.session.commit()


    return jsonify({
                'Name': new_film.name,
                'Rating': new_film.user_rating,
                'Status': new_film.status,
                'user_id': new_film.user_id,
    })
```

Тут ми можемо бачити функціонал додавання фільму до свого списку, на бекенд відправляється дані у форматі JSON які він зчитує та зберігає у змінну request після чого якщо дані вірні зберігає в базу даних.

У випадку якщо фільм який хоче додати користувач вже є у нього в списку то бєкенд поверне нам помилку да допомогою валідації.

3) DELETE:

```python
@app.route('/remove', methods=['DELETE'])
@jwt_required()
def remover():
    username = get_jwt_identity()
    name = request.json.get('name')
    user = User.query.filter_by(username=username).first()


    film_exist = User_List.query.filter_by(name=name,user_id=user.id).first()

    db.session.delete(film_exist)
    db.session.commit()

    response_body = {"Removed": film_exist.name}


    return response_body

```

Тут ми можемо бачити фуенціонал видалення фільму, це відбувається за допомогою Id, який бекенд отримує і двиляє за ним фільм з БД списку користувача

4) PUT:

```python
@app.route('/edit', methods=['PUT'])
@jwt_required()
def editor():
    username = get_jwt_identity()
    user = User.query.filter_by(username=username).first()
    name = request.json.get('name')
    new_name = request.json.get('new_name')
    new_status = request.json.get('new_status')
    new_rating = request.json.get('new_rating')

    film = User_List.query.filter_by(name=name,user_id=user.id).update(dict(name=new_name, status=new_status, user_rating=new_rating))

    db.session.commit()

    response_body = {
                    "success": "Good Job"
                    }
    return response_body
```

Тут ми можемо бачити схожий функціонал з методом DELETE але в цьому випадку користувач відправляє не тільки ID а і всю модель фільму для оновлення інформації.

3.1.1 *Приклади Frontend функціоналу:*

Front-end функціонал ми можемо поділити на дві великі частини: Components, в якому знаходятся всі компоненти нашої программи та Redux, де знаходятся всі дані які ми отримуємо від бекенду.

React компоненти:

Розглянемо на прикладі головної сторінки додатку (Home page):

```javascript
const [page, setPage] = React.useState(0);
const [modalActive, setModalActive] = React.useState(false);
const [filmTitle, setfilmTitle] = React.useState('');
const [currentfilm, setCurrentfilm] = React.useState('');

let filmId;

React.useEffect(() => {
  dispatch(loadSearch);
  console.log(films);
}, []);

const handleSearch = (e) => {
  e.preventDefault();
  dispatch(loadSearch(currentfilm));
};

const handleAdd = () => {
  dispatch(getSinglefilm(filmId));
};

const handleModal = (title) => {
  setModalActive(true);
  setfilmTitle(title);
};
```

В цій частині ми можемо бачити такий фунціонал:

React.UseState це реакт-хук який відповідає за локальне сховище данних,

у нашому випадку тут зберігається назви фільмів які ми отримали за допомогою метода dispatch. [21]

React.UseEffect це також хук який відповідає за рендер сторінки під час завантаження або перезавантаження сайту, тут відбувається dispatch(loadSearch) за допомогою якого ми шукаємо фільм по якого назві, а сам функціонал пошуку відбувається у функції handleSearch.
Функція handleModal відповідає за відкриття модального вікна для додовання фільму після пошуку.

```
return (
  <section className="home">
    <h1 className="home__title">Знайти Фільм</h1>
    <form className="search__form" onSubmit={handleSearch}>
      <input
        value={currentfilm}
        onChange={(e) => setCurrentfilm(e.target.value)}
        type="search"
        placeholder="введіть правильну назву фільму"
        required
        className="film__search"
      />
      <input className="search__submit green-btn" value="Поиск" type="submit" />
    </form>
    <div className="all__film__wrapper">
      {films.length == 0 ? (
        <div className="empty__wrapper">
          <h3 className="empty__msg">Будь ласка, введіть правильну назву фільму</h3>
        </div>
      ) : (
        films.map((film, id) => (
          <div key={film.mal_id} className="all__film__col">
            <div className="all__film__body">
              <div className="all__film__image">
                <img src={film.Poster} alt="film" />
                <button onClick={() => handleModal(film.Name_Ru)} className="add__film">
                  <img src={plus} alt="+" />
                </button>
                <div className="film__text__container">
                  <div className="all__film__title">{film.Name_Ru}</div>
                </div>
              </div>
            </div>
          </div>
        ))
      )}
    </div>
    <Modal title={filmTitle} active={modalActive} setActive={setModalActive} />
  </section>
);
```

На скрішоті ми можемо бачити рендер головної сторінки React в своїх компонентах генерує спеціальний HTML код який може використовувати весь функціонал фреймворку це також називается JSX.

На цьому прикладі ми можемо бачити тег <input> який і виконує пошук і відправляє дані зпочатку у Redux а потім він вже відправляє на бекенд де дані зберігаються у БД. У випадку якщо фільм не знайдено в БД то ми отримуємо повідоплення про це.

Функція films.map() генерує весь список фільмів який знайшов пошук та показує їх на екрані, пвсля чого користувач може додати його до свого списку. [21]

Внутрішні компоненти React:

Ми можемо бачити тег <Modal> це є внутрішною компонетною яку ми додали,

в неї ми передаємо дані, а саме назву фільму після чого відкриється модальне вікно. Саму розмітку компоненти ми можемо бачити на скріншоті:

```
return (
  <div className={active ? 'modal active' : 'modal'} onClick={() => setActive(false)}>
    <div onClick={(e) => e.stopPropagation()} className="modal__content">
      <div className="modal__title">
        {title} <img width="20" height="20" src={star} alt="*" />
      </div>
      <form onSubmit={handleSubmit} className="film__add">
        <select
          value={status}
          onChange={handleInputChange}
          name="status"
          id="1"
          className="film__add__select">
          <option value="none" className="film__add__option">
            Выбери статус
          </option>
          <option value="Watched" className="film__add__option">
            Просмотренно
          </option>
          <option value="Planning" className="film__add__option">
            Буду смотреть
          </option>
        </select>
        <select···
        </select>
        <input type="submit" value="Добавить" className="film__add__submit" />
      </form>
    </div>
  </div>
);
```

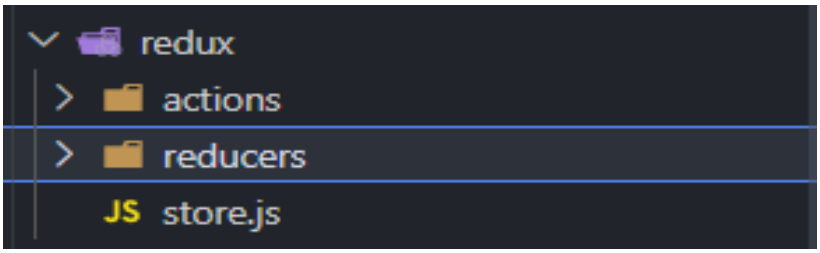
Тут користувач обирає статус на оцінку фільму після чого ці дані пи передаємо у функцію яка перетворює дані у формат JSON та відправляє на бекенд за допомогою useDispatch()

```
const navigate = useNavigate();
const dispatch = useDispatch();
const [data, setData] = React.useState({
  username: 'admin',
  name: '',
  user_rating: '',
  status: '',
});

const { name, user_rating, status } = data;
```

Бібліотека Redux:

Redux це бібліотека React за допомогою якої ми можемо керувати даними які нам приходять з бекенду та зберігати їх, структура бібліоетеки поділяється

на двві частини, Actions та Reducers та головний файл store.js де зберігається всі данні.



У файлі **Actions** ми можемо бачити такий функціонал

```
export const loadSearch = (searchfilm) => (dispatch) => {

    dispatch(setLoaded(false))
    axios({
        url: `/search?q=${searchfilm}`,
        method: "POST",
        headers: {
            'Content-Type': 'application/json;charset=utf-8',
        },
        data: {
            'search_film': 'True',
            'limit': 20
        }

    }).then(({ data }) => {
        console.log(data)
        dispatch(getCatalog(data))

    }).catch((error) => {
        console.log(error)
    });
}
```

Тут відбувається POST запит на бекенд для пошуку фільму, ми відправляємо параметр searchfilm за допомогою якого бекенд шукає співпадіння в БД якщо запит вдалий то ми отримуємо список за допомогою dispach, якщо запит не відбувся то ми отрамаємо помилку.

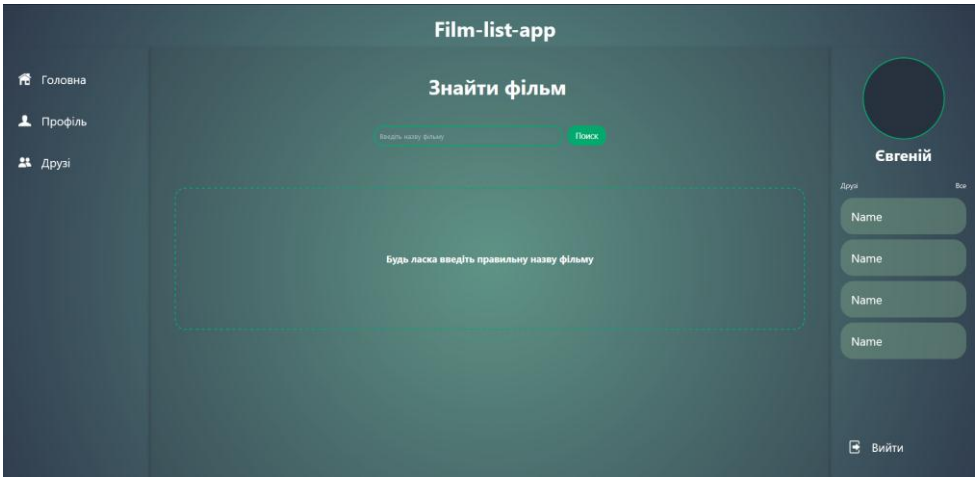
В headers ми визначаємо правильний формат запиту а саме application/json формат, якщо відправити не в цьому форматі ми отримуємо повідомлення з помилкою.

У файлі **Reducers** ми отримуємо, оновлюємо наші данні:

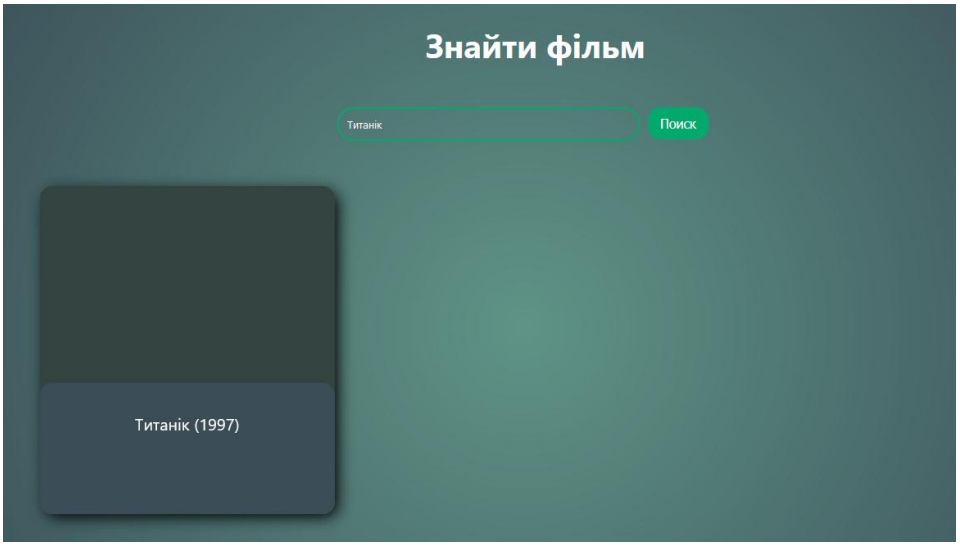
```
const initialState = {
    films: [],
    currentPage: 50,
    perPage: 10,
    totalCount: 0,
    isLoaded: false,
}

const AllfilmReducer = (state = initialState, action) => {
    switch (action.type) {
        case 'CATALOG_GET':
            return {
                ...state,
                films: action.payload,
                currentPage: action.payload,
                isLoaded: true,
            }
        case 'USERS_GET':
            return {
                ...state,
                users: action.payload,
                isLoaded: true,
            }
        case 'SET_LOADED':
            return {
                ...state,
                isLoaded: action.payload,
            }
        case 'SET_CURRENT_PAGE':
            return {
                ...state,
                currentPage: action.payload,
            }
        case 'SET_SEARCH_film':
            return {
                ...state,
                seatchfilm: action.payload,
            }
        default:
            return state;
    }
}
```

Через фунціонал switch case ми можемо керувати файлом і отримувати ті данні які нам потрібно.

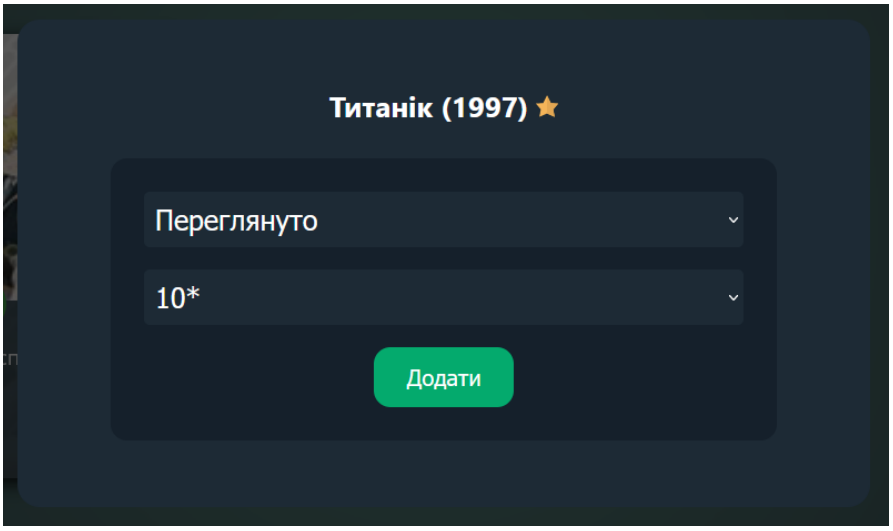
3.4 Реалізація інтерфейсу



На скріншоті ми можемо побачити гловну сторінку нашого веб-додатку яка описана в компоненті Home.jxs. Ця сторінка містить декілька внутрішнік компонентів, таких як навігація, меню профілю, та найголовніша частина нашоно проекту- пошук фільмів.

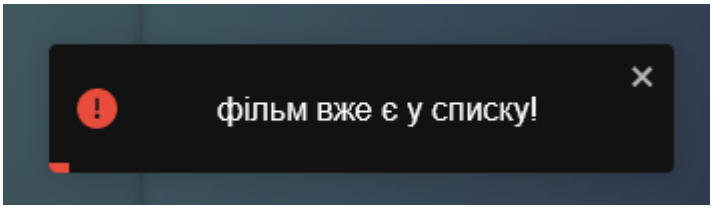


На скріншоті ми можемо бачити функціонал пошуку на прикладі фільму Титанік



Після пошуку у користувача є можливість додати фільм у свій власний список. Він має обрати статус фільму (у нашому випадку переглянуто) та оцінку фільму, після цього якщо цього фільму у нього немає у списку його буде успішно додано.

У випадку якщо фільм є дублікатом в списку користкувача ми отримаємо повідомлення про помилку:



4. Техніко-економічне обґрунтування розробки підсистеми (Бізнес-план)

4.2. ПРОЕКТОВАНИЙ ПРОДУКТ АБО ВИД ПОСЛУГ

Які потреби повинен задовольнити продукт проекту

Наш веб-додаток, має на меті задовольнити потреби кіношанувальників у кількох аспектах. Перш за все, він пропонує простий і зручний спосіб стежити за історією переглядів та складати унікальний список улюблених фільмів. Крім того, наш веб-сайт дозволяє користувачам шукати нові фільми на основі їх рейтингів і смаків. Також, у нашому продукті є соціальна мережа, де кіномани можуть приєднуватися, обмінюватися списками фільмів і вести розмови про них.

Особливості та відмінні риси нашого продукту, які роблять його привабливим для користувачів порівняно з конкурентами, включають наступне:

1) Можливість створення власних списків фільмів і їх сортування за жанром, режисером і роком випуску.
2) Оцінки фільмів, що базуються на вподобаннях користувачів, а також індивідуальні рекомендації фільмів.
3) Наявність соціальної мережі, де користувачі можуть спілкуватися з друзями, стежити за іншими користувачами та брати участь в бесідах про фільми.
4) Рекомендації щодо фільмів, підготовлені нашою командою кіноекспертів, які вважають ці фільми вартими перегляду.
5) Наявність преміум-членства, яке надає персоналізовані пропозиції щодо фільмів, ранній доступ до нових релізів та унікальний контент.
6) Можливість створення персоналізованих списків фільмів на основі тематики або жанрів.
7) Можливість спілкування користувачів між собою і створення відчуття спільноти, що об'єднує їх захоплення фільмами через різноманітні громадські заходи, такі як покази фільмів і дебати.

Отже, наш продукт проекту відповідає потребам користувачів, надаючи їм

зручний спосіб стежити за переглядами, шукати нові фільми та спілкуватися з іншими кіноманами. Його особливості і функціональні можливості виокремлюють його серед конкурентів та роблять його привабливим вибором для шанувальників кіно.

Наявність патентів або авторських свідоцтв на продукт проекту

Відсутні*

Попередня оцінка ціни реалізації продукту проекту і витрат на його виробництво

Технічне завдання (ТЗ) являє собою детальний документ, у якому описано, хто і які види робіт зі створення сайту виконує. ТЗ необхідне для того, щоб уявлення про проєкт у свідомості замовника і розробників максимально збігалися. Саме тому ТЗ має бути якомога докладнішим, однозначним і об'єктивним. Ціна розробки залежить від факторів: - Наявності вихідної інформації. - Кількості вимог замовника до проєкту.	від 3500 грн
Вартість дизайну. Що більше вікон/сторінок різної структури потрібно створити, то вищою буде вартість дизайну. Ціна може розраховуватися декількома способами: 1. Відштовхуючись від вартості за одне вікно/сторінку. 2. Відштовхуючись від кількості витрачених на роботу годин. За умови вибору першої схеми, щоб розрахувати вартість, варто врахувати такі моменти: - Кожен пункт меню – це 1 сторінка. - Вартість дизайну контентної частини будь-якої статті розраховується за 1 блок.	від 1200 грн за вікно/сторінку
Вартість програмування (розробки основної бізнес логіки). Для повноцінної роботи проєкту може знадобитися додатковий функціонал, не передбачений «базовою» версією CMS (англ. Content Management System – Система керування вмістом). Ціна буде сформована з урахуванням усіх внесених до проєкту доробок. Вартість 1 години роботи програміста залежить від його рівня кваліфікації та досвіду. З переходом на новий рівень цінник збільшується у 1,5-2 рази. До того ж замовлення послуг фахівця у веб-студії обходиться удвічі дорожче порівняно із залученням до роботи програміста на фрілансі.	від 450 грн/год

Наповнення.	від 250 грн/год
Ведення проєкту і консультації клієнта. Що частіше і довше за часом буде відбуватися спілкування з виконавцем під час робіт, то більшою буде сума у рахунку, який він виставить, закінчивши проєкт. Тому варто заздалегідь продумати список вимог до майбутнього продукту і обговорити їх з розробником перед початком.	від 600 грн/год

Загальна вартість проекту – 130 тис. грн.

Зважаючи на використання власних коштів і можливість додаткової залучення інвестицій, передбачається, що проект буде реалізований. Очікується, що продукт проекту принесе прибуток в розмірі 200 000 грн протягом першого року з потенціалом значного зростання у наступні роки, враховуючи дослідження ринку та прогнозоване зростання.

Наш веб-додаток має кілька переваг та якісних показників, включаючи:

1) Інтуїтивно зрозумілий дизайн та зручний інтерфейс, що спрощують використання.
2) Персоналізовані пропозиції фільмів на основі відгуків користувачів та списки фільмів, рекомендованих нашою командою кіноекспертів.
3) Можливість створювати персоналізовані списки фільмів на основі тематики або жанру.
4) Привілеї преміум-підписки, включаючи ранній доступ до нових релізів та індивідуальні пропозиції.
5) Соціальна мережа, де любителі кіно можуть спілкуватися та обговорювати фільми.

Наш веб-додаток є технічним продуктом, який потребуватиме постійного обслуговування та оновлень. Для цього буде створена спеціальна команда, яка буде забезпечувати технічну підтримку та безперебійну роботу програми. Також, буде надано обслуговування клієнтів для вирішення будь-яких питань або проблем, з якими можуть зіткнутися користувачі.

4.3 ОЦІНКА РИНКУ ЗБУТУ

Виявлення відсутніх даних:

а) Умови постачання, виробництва та продажу кінцевого продукту проекту:

Наш веб-додаток буде доступний онлайн з будь-якої точки світу. Він буде повністю розроблений та наданий онлайн, а також буде продаватися через платформи, такі як магазини програм і наш веб-сайт.

b) Потенційні суперники:

IMDb, Letterboxd і Rotten Tomatoes - лише кілька з сайтів, які ми визначили як потенційних конкурентів на ринку. Вони надають послуги, схожі на наш веб-додаток, наприклад списки фільмів і спеціальні пропозиції. Однак, ми вважаємо, що наша соціальна платформа та сервіс створення персоналізованих списків фільмів вирізнять нас серед конкурентів. Крім того, наш додаток буде простішим у використанні та більш інтуїтивно зрозумілим, ніж у наших конкурентів, що забезпечить кращий загальний досвід для наших користувачів.

Джерело інформації:

а) Власне дослідження:

Ми проведемо дослідження ринку, щоб дізнатися більше про звички, смаки та вимоги кіноманів. Також ми зіберемо інформацію про пропозиції та вартість наших конкурентів.

б) Місцеві підприємницькі палати та асоціації:

Ми також будемо звертатися до місцевих торгово-промислових палат, торгових асоціацій та інших груп, щоб отримати додаткову інформацію про ринок і галузь.

Аналіз даних:

а) У найближчій та довгостроковій перспективі, хто, чому, скільки та коли буде готовий придбати продукт проекту:

Ми передбачаємо, що кіномани, які шукають простий спосіб відстежувати свої улюблені фільми та знаходити нові, стануть основним цільовим ринком нашого веб-додатка. Ці клієнти будуть готові придбати продукт, щойно він стане доступним. Ми також спрямовуватимемося на шанувальників кіно, які бажають спілкуватися з іншими та відвідувати місцеві заходи. Наш продукт

матиме прийнятну ціну порівняно з іншими аналогічними продуктами на ринку.

b) Встановлення зразкової ціни продажу продукту проекту та умов конкурсу:

На основі наших досліджень і аналізу ми вважаємо, що продажна ціна нашого веб-додатка буде складати 5-10 доларів США на місяць. Ця ціна буде конкурентоспроможною з нашими конкурентами, і ми будемо пропонувати додаткові функції, такі як соціальна платформа та налаштований сервіс списків фільмів, щоб виділитися серед конкурентів.

Впровадження заходів без використання цих даних, на прикладі бізнесу: Ми зосередимося на створенні сильного бренду для нашого онлайн-додатка, що включатиме помітний логотип і запам'ятовувальний слоган. Ми також будемо використовувати соціальні мережі для просування та залучення користувачів. Витратимо кошти на пошукову оптимізацію (SEO), щоб поліпшити нашу присутність в Інтернеті та збільшити відвідуваність веб-сайту. Крім того, ми будемо співпрацювати з рецензентами фільмів і блогерами, щоб залучити увагу до нашого додатка і привернути нових користувачів.

4.4. КОНКУРЕНЦІЯ

Хто є найбільшим виробником аналогічного продукту

Щоб дослідити ринок та конкуренцію, необхідно вивчити провідні веб-сайти, програми або сервіси, що пропонують подібні характеристики до запропонованої веб-програми. Наприклад, IMDb, Letterboxd і Rotten Tomatoes відомі сайти, де користувачі можуть створювати списки своїх улюблених фільмів.

Ось інформація про ці веб-сайти та їх діяльність:

а) Об'єкти продажу:

IMDb є дочірньою компанією Amazon і входить до їхньої екосистеми реклами та електронної комерції. Вони продають рекламний простір на своєму веб-сайті, а також надають преміальні послуги, такі як IMDbPro, яка є передплатним сервісом для професіоналів з сфери розваг.

б) Доходи:

Доходи IMDb отримують від реклами та преміальних підписок. За даними Statista, їхні доходи в 2020 році склали приблизно 192 мільйони доларів США.

в) Впровадження нових моделей:

IMDb розширює свої послуги і нещодавно запустив IMDb TV, який є безкоштовним потоковим сервісом, що підтримується рекламою. Вони також пропонують оригінальний контент, такий як телешоу та документальні фільми.

г) Технічний сервіс:

IMDb має надійну технічну інфраструктуру і інвестує в поліпшення користувацького досвіду та функціональності. Вони також надають довідковий центр та канали підтримки для допомоги користувачам.

д) Рекламна компанія:

IMDb вкладає кошти в рекламу для просування своїх послуг та контенту. Вони використовують різні канали, такі як соціальні мережі, пошукові системи та програмну рекламу.

Що стосується продукту конкурентів:

а) Основні характеристики:

IMDb - це веб-сайт, який надає детальну інформацію про фільми, телевізійні програми та інші розважальні вироби. Він містить велику базу даних назв, акторського складу, знімальної групи, рецензій, рейтингів та трейлерів. Також в ньому присутні функції спільноти, такі як огляди, дошки оголошень і списки користувачів.

б) Рівень якості:

Як експерти, так і ентузіасти часто використовують IMDb як надійне джерело точної інформації. Однак якість інформації, створеної користувачами, наприклад, відгуки та оцінки, може розрізнятися.

в) Дизайн:

Дизайн IMDb є простим і практичним, але через великий обсяг інформації він може здаватися дещо переповненим. Недавно вони вдосконалили зручність використання свого дизайну.

г) Думка зацікавлених сторін:

Для зрозуміння сприйняття IMDb та вашої веб-програми важливо отримати відгуки та повідомлення від зацікавлених сторін та потенційних користувачів.

Можна проводити опитування, інтерв'ю або фокус-групи, щоб зібрати якісні та кількісні дані. Для оцінки настроїв також можна переглядати огляди в Інтернеті та відгуки у соціальних мережах.

4.5. СТРАТЕГІЯ МАРКЕТИНГУ

Ринкова стратегія, що передбачає проникнення на ринок, полягає встановленні адекватної ціни на послугу, з метою привернути максимальну кількість відвідувачів і завоювати значну частку ринку. Ця стратегія є виправданою для масштабного виробництва, яке ускладнює доступ на ринок для дрібних і середніх підприємств з обмеженими фінансовими можливостями.

У ціноутворенні може бути важко вибрати найкращу стратегію. Тому важливо врахувати різні аспекти, такі як конкуренція, цільовий ринок і ціннісна пропозиція. Нижче наведено деякі фактори, які слід врахувати:

Вибір підходу до встановлення цін: ви можете обрати одну з кількох стратегій ціноутворення, включаючи конкурентоспроможну ціну, ціноутворення на основі вартості і ціноутворення на основі витрат. Важливо переконатися, що обраний тарифний план відповідає готовності користувачів платити за надану послугу.

Розуміння фінансових цілей і очікувань допоможе встановити розумні цільові ціни і оцінити можливості їх поліпшення.

Реклама є важливим елементом для поширення інформації про вашу онлайн-програму та привертання споживачів. Ось деякі аспекти, які варто розглянути:

а) Організаційні стратегії реклами: ви можете використовувати різні канали реклами, такі як соціальні мережі, пошукові системи, програмну рекламу та впливовий маркетинг.

б) Сегментація аудиторії: визначте свою цільову аудиторію і розбийте її на сегменти, щоб акуратно налаштувати свою рекламну кампанію для кожного сегмента.

с) Змістовна стратегія: створюйте цікавий та корисний контент, який привертає увагу вашої аудиторії і стимулює їх брати участь в вашій онлайн-програмі. Розгляньте використання відео, інфографіки, блогів, вебінарів та

інших форматів контенту.

d) Точне визначення бюджету: встановлення реалістичного рекламного бюджету, що враховує ваші фінансові можливості і цілі.

e) Відстеження результатів: встановіть метрики та аналітику для вимірювання ефективності вашої рекламної кампанії. Аналізуйте дані, щоб виправити та оптимізувати ваші стратегії в майбутньому.

Загалом, успішна ринкова стратегія потребує глибокого розуміння вашої цільової аудиторії, конкурентного середовища та особливостей вашої послуги. Продумана стратегія ціноутворення та реклами допоможуть вам залучити більше користувачів і завоювати значну частку ринку.

4.6. ПЛАН ВИРОБНИЦТВА

Підстави для розробки

Документ, на підставі якого ведеться розробка

ISO/IEC 27001:2013

Найменування теми розробки

Інструмент визначення оптимального місцерозташування для підприємства.

Організація, що затвердила підставу розробки, і дата його затвердження

Дана тема виконується в рамках учбового процесу в Київському національному університеті будівництва і архітектури з предмету «Управління ІТ проєктами» у відповідності до завдання на курсову роботу від 1 березня 2023 року, та робочої програми дисципліни.

Призначення розробки

Критерії ефективності та якості програми

Оскільки фізичного виробництва немає, немає обмежень щодо потужності, які потрібно враховувати. Однак важливо забезпечити масштабованість програми для обслуговування зростаючої бази користувачів та попиту з часом. Це може вимагати інвестицій у додаткову потужність сервера, оновлення програмного забезпечення або збільшення команди

розробників для підтримки та вдосконалення програми.

Оскільки веб-додаток розміщено в Інтернеті, немає обмежень пропускної здатності в традиційному розумінні. Проте важливо забезпечити масштабованість програми для обслуговування зростаючої бази користувачів та попиту з часом. Це може вимагати інвестицій у додаткову потужність сервера, оновлення програмного забезпечення або збільшення команди розробників для підтримки та вдосконалення програми. Важливо оцінити очікуваний темп зростання бази користувачів і спланувати відповідно. Основне завдання програми - надання користувачам можливості складати свій власний список улюблених фільмів та ділитися ним з іншими користувачами.

Вимоги до програми:

- Функціональні характеристики:
- Забезпечення автозаповнення або підстановки даних у визначені поля для прискорення роботи;
- Розрахунок за різними критеріями;
- Використання фільтрів з налаштуванням користувачем;
- Налаштування параметрів роботи, таких як локалізація та колірна тема;
- Створення списків закладок та порівнянь;
- Збереження і перегляд історії операцій;

Ефективність та продуктивність:

- Швидка відповідь на запити користувачів для забезпечення зручного використання програми;
- Оптимізація використання ресурсів сервера для підтримки багатьох користувачів одночасно;
- Масштабованість системи для збільшення обсягу обробки даних при зростанні попиту.

Безпека:

- Забезпечення захисту персональних даних користувачів;
- Аутентифікація та авторизація користувачів для захисту від несанкціонованого доступу;
- Захист системи від зловмисних атак та вразливостей програмного забезпечення.

Зручність використання:

Інтуїтивний та зручний інтерфейс для забезпечення простоти навігації та взаємодії користувачів з програмою;

Доступність на різних платформах, таких як веб, мобільні пристрої тощо.

Підтримка та розвиток:

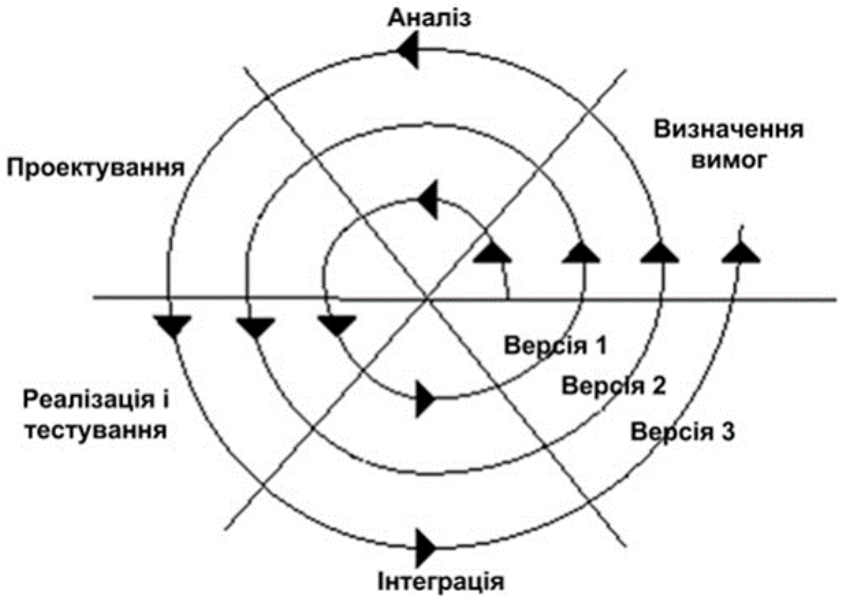
- Забезпечення постійної підтримки та оновлення програмного забезпечення для усунення помилок та вдосконалення функціональності;
- Здатність до легкого розширення та модифікації програми для додавання нових функцій і функціональності.

Це лише загальні вимоги, і конкретні деталі програми можуть варіюватись в залежності від вашої бізнес-моделі, цільової аудиторії та інших факторів. Рекомендується провести детальний аналіз вимог і розробити план реалізації з урахуванням цих факторів.

Дана ціна відповідає створенню ПЗ-ресурсу на 3-5 вікон з унікальним дизайном по 2-6 інформаційних блоків на сторінці та додатковим функціоналом на кшталт довідки, особистого кабінету тощо. Приблизна вартість одного такого «блоку» послуг – 11600грн. Продукт нараховуватиме близько 9 різних категорій-розділів, отже:
Загальна вартість: 135000 грн.

Модель життєвого циклу

Спіральна модель має кілька переваг. Перш за все, ітераційна розробка полегшує коригування проекту у разі змін у вимогах замовника. Крім того, вона дозволяє поступово інтегрувати окремі елементи до кінцевого продукту. Це сприяє зниженню ризиків під час реалізації проекту та забезпечує гнучкість в управлінні ним. Ітераційний підхід також підходить для повторного використання компонентів і дозволяє удосконалювати процес розробки.



Стадії розробки програмного продукту в спіральній моделі включають визначення тематики та основної мети проекту, розробку технічного завдання, прототипування, макетування та дизайн, програмування, наповнення контентом, тестування, здачу готового проекту, просування і технічний супровід.

Отже, спіральна модель життєвого циклу програмного продукту, яка базується на ітераційному підході, дозволяє ефективно розвивати продукт, забезпечуючи гнучкість, зниження ризиків і полегшення коригування проекту у процесі розробки.

4.7. ОРГАНІЗАЦІЙНИЙ ПЛАН

Комунікації в проекті

Організаційний план має включати посадові інструкції та обов'язки для кожної посади в організації, а також повноваження та критерії для експертів. Це допомагає чітко визначити ролі та належним чином розподілити обов'язки. Крім того, важливо визначити процедури найму та навчання нового персоналу для забезпечення відповідної підготовки.

Застосування страхових покриттів, включаючи правові та нормативні зобов'язання, такі як ліцензії та дозволи, також є важливим для забезпечення відповідності юридичним вимогам та захисту компанії. У визначенні

організаційної структури слід враховувати розмір бізнесу та очікувані темпи зростання.

Комунікація між відділами та спільна робота впливають на успіх компанії, тому важливо визначити структуру звітності, маршрути спілкування та заходи для формування командної роботи та збереження здорової робочої атмосфери. Організаційна стратегія також повинна включати пільги та винагороди, такі як діапазони заробітної плати, бонуси та соціальні пільги, для привернення та збереження талановитих співробітників.

Безпека праці має велике значення, тому важливо забезпечити безпеку працівників, зокрема шляхом використання ергономічних робочих станцій, навчання щодо технік безпеки та наявності протоколів для надзвичайних ситуацій.

Опис кадрового складу команди, включаючи кількість працівників, ролі та обов'язки, а також інформацію про набір та навчання, також є важливим. Утримання та розвиток співробітників також слід враховувати, зокрема за допомогою програм навчання та розвитку та можливостей для просування в компанії.

Таблиця 1

Посада	Обов'язки	Чисельність
Менеджер проекту	Відповідає за планування та виконання всіх завдань щодо проекту. Він чи вона координує роботу різних команд, спостерігає за всіма процесами та стежить за тим, щоб усе виконувалось у встановлені терміни та проект залишався в рамках бюджету.	1
IT-аналітик	Аналіз усіх організаційних процесів у компанії, вивчення вже існуючих інформаційних систем та пошук можливостей для оптимізації роботи.	1
Розробник	Створення програмного забезпечення з використанням різних мов програмування. Back-end розробники будують внутрішню логіку і функціональність продукту, а front-end розробники, своєю чергою, працюють над елементами, із якими взаємодіють користувачі.	2

QA-фахівець	Стеження за якістю продукту, його тестування та виявлення недоліків, які необхідно усунути.	1
UI та UX дизайнер	Зосередження на ефективній взаємодії між кінцевими користувачами та додатком. Проектування інтерфейсу таким чином, що продукт виглядає привабливо і зручно ним користуватися.	1
Фахівець з аналізу даних	Збір, перевірка та аналіз даних, що дозволяє приймати рішення щодо розвитку проекту на їх основі.	1

Спеціалісти будуть набиратися на конкурсній основі, з урахуванням їхніх професійних навичок та досвіду у взаємодії з клієнтами, здатності швидко вирішувати складнощі та відповідати на можливі запитання. Крім того, можливо залучення студентів, що навчаються на відповідних спеціальностях, для роботи в компанії.

4.8. ЮРИДИЧНИЙ ПЛАН

Форма організації

Даний IT-проект планує бути організованим як одноосібне володіння, що є організаційно-правовою формою підприємства, де одна особа володіє і керує ним, несе повний ризик збитків або одержує весь прибуток.

Вибір такої форми володіння має наступні переваги:

-Безпосереднє керування власником, що забезпечує свободу та оперативність дій.

-Невеликі початкові капіталовкладення.

- Власник отримує весь прибуток.

- Можливість закриття підприємства за потреби.

Однак, існують також недоліки одноосібного володіння:

- Труднощі з приверненням великих капіталів.

- Обмежена платоспроможність, що може впливати на доступ до кредитів в банках.

- Повна особиста відповідальність за борги.

- Відсутність спеціалізованого менеджменту.
- Невизначеність термінів функціонування.

Оскільки власник проекту буде отримувати весь прибуток, він зацікавлений в ефективній роботі. Також вибір цієї форми володіння обумовлений простотою оподаткування (оподатковується лише індивідуальним податком на дохід) та збереженням конфіденційності діяльності.

Джерелами капіталу для одноосібного володіння є особисті заощадження та позики від банків. Для розробки продукту планується використати власні кошти в розмірі 150 тис. грн.

Щодо форми власності, проект має статус приватного підприємства, яке засноване на власності фізичної особи. Законодавство України визнає власника як підприємця, і тому власність та управління майном не розділяються. Власник має право встановити розмір статутного капіталу на свій розсуд, який зазвичай не потребує додаткового підтвердження при реєстрації, але може вимагати певні документи. Для приватного підприємства рекомендується спрощена система оподаткування (єдиний податок), яка є оптимальною.

Власник має повне право самостійно визначати розмір статутного капіталу, який не обмежений законодавством. Втім, зазначений розмір повинен бути вказаний у статутних документах. Часто при реєстрації приватного підприємства не вимагається офіційне підтвердження формування статутного капіталу, але наявність відповідних документів може бути корисною. Наприклад, це може бути довідка з банку або акт приймання-передачі майна.

Система оподаткування є ще одним аспектом, який треба врахувати при виборі форми власності. Приватні підприємства часто використовують спрощену систему оподаткування, так званий єдиний податок. Ця система дозволяє спростити процедуру оподаткування та зменшити податкове навантаження на підприємця. Він обчислюється на підставі виручки або обсягу реалізованої продукції і може бути більш привабливим варіантом для малих і середніх підприємств.

У контексті проекту, зазначається, що основними джерелами капіталу для одноосібного володіння є особисті збереження та позики від банків. Власник проекту планує використати власні кошти у розмірі 150 тис. грн. для розробки продукту. Це дозволить зберегти контроль над підприємством і має певні фінансові переваги, оскільки власник отримує повний прибуток з проекту.

Враховуючи всі переваги та недоліки, а також особливості форми власності, здавалося би, що одноосібне володіння є вигідним вибором для даного ІТ-проекту "оцінки". Приватне підприємство дає власнику безпосереднє керування, свободу в управлінні та можливість оперативно реагувати на зміни. Крім того, простота оподаткування та збереження конфіденційності діяльності сприяють вибору цієї форми власності. Проте, варто врахувати й недоліки, такі як труднощі з приваблennям великих капіталів, повна відповідальність за борги та відсутність спеціалізованого менеджменту.

4.9. ОЦІНКА РИЗИКУ І СТРАХУВАННЯ

Управління ризиками проекту (Project Risk Management) є процесом, спрямованим на зниження можливості виникнення негативних умов для успішної реалізації проекту та мінімізацію непередбачуваних втрат. Цей процес включає пошук, прийняття і виконання управлінських рішень, що спрямовані на управління ризиками проекту.

Перший крок у плануванні ризиків - ідентифікація ризиків. Це включає визначення потенційних ризиків, які можуть вплинути на проект. Ідентифікація ризиків повинна проводитись на протязі всього життєвого циклу проекту і має на меті створити перелік подій ризику, які можуть мати як негативний, так і позитивний вплив на проект.

Після ідентифікації ризиків слід перейти до оцінки їх впливу. Це включає визначення імовірності виникнення ризику і його наслідків для проекту. Цей аналіз допомагає визначити, які ризики потребують розробки заходів управління, а які можуть бути прийняті без спеціальних заходів. Оцінка ризиків може здійснюватися як якісно (за допомогою категорій або рейтингів) так і кількісно (за допомогою числових показників і моделей).

Планування ризиків складається з наступних кроків:

- ідентифікація ризиків;
- оцінка ризиків;
- розробка заходів реагування на ті ризики, які цього вимагають.

Ідентифікація ризиків полягає у визначенні того, які ризики здатні вплинути на конкретний проект та інші пов’язані з ним проекти у рамках портфелів чи програм проектів. Ідентифікація ризиків має проводитись на всьому життєвому шляху проекту. Мета ідентифікації ризиків – скласти перелік подій ризику, які можуть вплинути на проект (вплив ризиків може бути негативним і позитивним). Ідентифікація ризиків – не разова дія, вона повинна проводитися регулярно.

Оцінка ризиків виконується з точки зору їх впливу на хід і результати проекту. Метою такого аналізу є визначення того, які події ризику вимагають розробки заходів реагування, а які – ні. Для того, щоб обґрунтовано вирішувати такі питання, слід пов'язати з кожним з проектних ризиків оцінки імовірності їх появи і наслідків для проекту та інших пов’язані з ним проектів, портфелів та програм. Оцінка ризиків може виконуватися за допомогою якісних рівнів або
кількісних показників.

Управління ризиками проекту включає:

- виявлення та ідентифікація передбачуваних ризиків;
- аналіз і оцінка ризиків;
- вибір методів управління ризиком;
- застосування обраних методів і прийняття рішень в умовах ризиків;
- реагування на наступ ризикового події;
- розробка і реалізація заходів зниження ризиків.
- контроль, аналіз та оцінка дій щодо зниження ризиків і вироблення рішень.

Серед методів управління ризиками найбільш поширені такі, як:

- розробка і реалізація стратегії управління ризиками;
- методи компенсації ризиків, що включають прогнозування зовнішнього середовища

- проекту, маркетинг проектів і продуктів проекту, моніторинг оточуючого середовища і
- створення системи резервів проекту;
- методи локалізації ризиків, які застосовуються для великих і складних проектів, що
- спеціальних груп аналітиків для оцінки ризиків;
- методи уникнення ризиків, що включають відмову від ризикованих проектів і ненадійних партнерів, страхування ризиків, пошук гарантів.

4.10. ФІНАНСОВИЙ ПЛАН

План прибутків і збитків

Рекомендується складати план прибутків і збитків в помісячному розрізі з урахуванням сезонних коливань та інших факторів, що впливають на величину прибутку. При складанні цього плану, корисно використовувати плани собівартості продукції на відповідний плановий період. У таблиці 2 наведений фрагмент такого плану, де враховано місячний розподіл виробництва продукту.

Таблиця 2

Фрагмент плану прибутків і збитків (тис. грн)

№ п/п	Показники	Місяці			
		квітень	травень	червень	Всього, I квартал
1.	Виручка (дохід) від реалізації продукції (без ПДВ і акцизів)	1526,0	1784,0	1897,0	5207,0
2.	Змінні витрати	342,0	412,0	245,0	999,0
3.	Маржинальний прибуток	412,0	321,0	487,0	1220,0
4.	Умовно-постійні витрати	55,0	55,0	55,0	165,0
5.	Прибуток до оподаткування	820,0	996,0	743,0	2559,0
6.	Податок на прибуток	40,0	49,8	37,1	160,2
7.	Чистий прибуток	780,0	946,2	705,9	1020,1

Баланс грошових витрат і надходжень включає розподіл загального прибутку, оплату податків, виплату боргів і процентів за кредити, а також розподіл прибутку між власниками підприємства та самим підприємством. Чистий прибуток, що залишається після врахування всіх витрат, може бути спрямований на винагороди для персоналу, благодійні внески або залишений на підприємстві для інвестицій та створення резервного фонду.

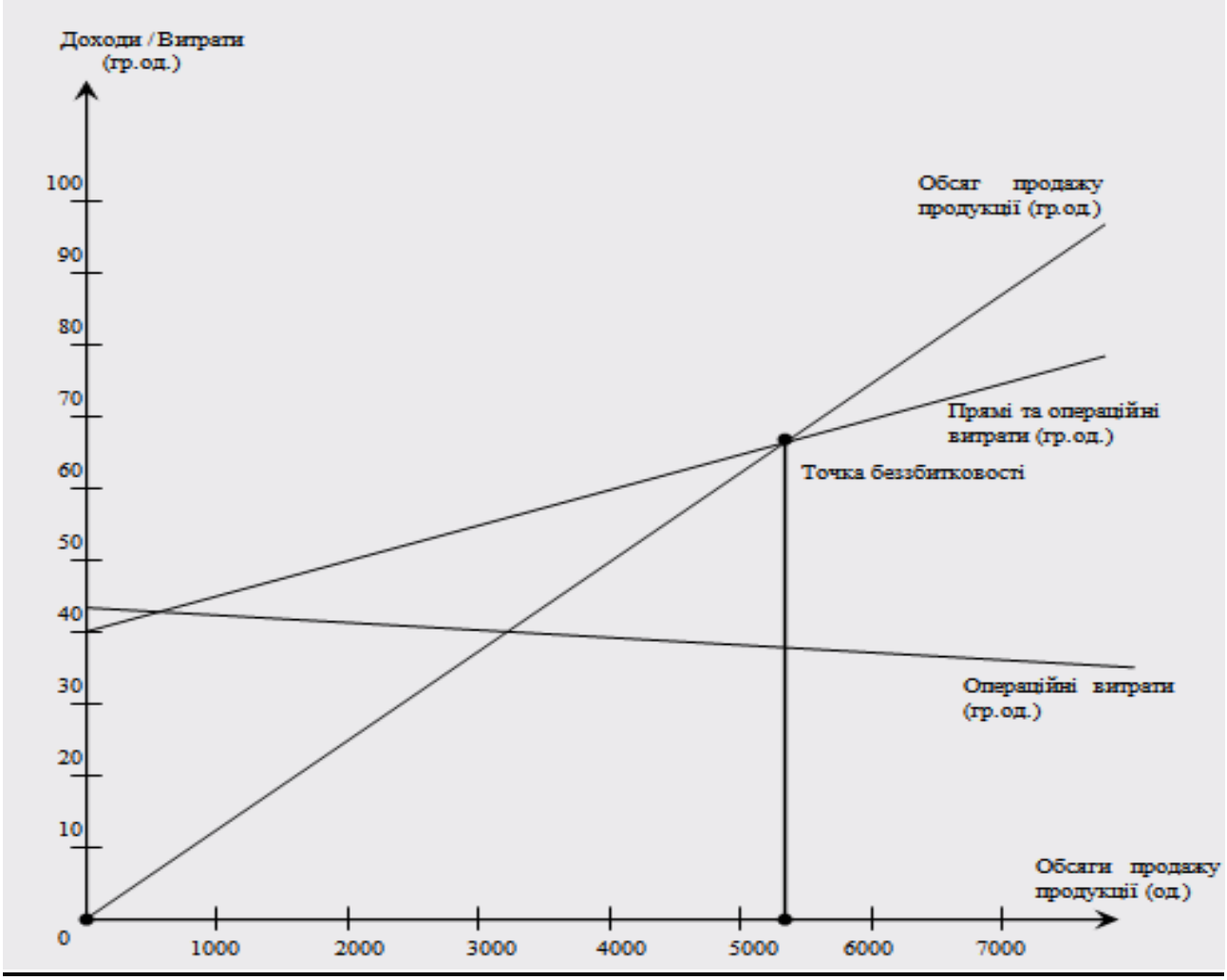
Графік досягнення беззбитковості показує, коли доходи від продажу продукції повністю покривають витрати на її виробництво й реалізацію, що призводить до відсутності прибутку або збитків. Точка беззбитковості визначає, скільки одиниць продукції має продати фірма, щоб покрити свої витрати. Продаж кожної наступної одиниці продукції приносить фірмі прибуток, а зменшення обсягів продажу нижче рівня беззбитковості означає збитки.

Для розрахунку точки беззбитковості необхідно знати ціну продажу одиниці продукції, прямі витрати на виробництво одиниці продукції та загальні операційні витрати. Обчислення точки беззбитковості здійснюється за певною формулою.

$$Тб = Зов/ (Цоп - Вп) ,$$ де

Зов – загальні операційні витрати фірми; Цоп – ціна одиниці продукції фірми; Вп — прямі витрати на одиницю продукції.

Точку беззбитковості можна визначити і графічним методом:



Підприємець може використовувати процес обчислення точки беззбитковості (ТБ) для моделювання різних ситуацій. Він має можливість змінювати ціну одиниці продукції, рівень прямих та операційних витрат і аналізувати, як такі зміни впливають на прибуток фірми. Цей процес дозволяє підприємцю оцінити варіанти й знайти оптимальні рішення для досягнення бажаного рівня прибутку.

4.11. СТРАТЕГІЯ ФІНАНСУВАННЯ

Для фінансування проекту загальною вартістю 135 тис. грн. передбачається використання різних джерел фінансових ресурсів. Першочерговим джерелом капіталу будуть власні кошти, в розмірі 150 тис. грн. Однак, при необхідності, можуть бути залучені й інші джерела фінансування. Нижче перераховані можливі джерела фінансування:

1) Власні засоби: Засновник проекту може використовувати свої особисті збереження для розробки та запуску додатку.
2) Кредити банків: Можна звернутися до банків і отримати кредит для фінансування проекту.
3) Залучення засобів партнерів: Можна залучити партнерів, які внесуть інвестиції або нададуть необхідні ресурси, наприклад, технічну підтримку
4) Залучення засобів акціонерів: Можна випустити акції компанії і залучити кошти від інвесторів.

Термін повернення вкладених засобів може залежати від різних факторів, включаючи успішність додатку та ринкові умови. Зазвичай, очікуваний термін повернення становить приблизно 2-3 роки після запуску додатку. Інвестори можуть отримати дохід від своїх вкладень у формі дивідендів або продажу своїх акцій у разі подальшого розвитку компанії.

ВИСНОВОК

Висновок полягає в тому, що додаток для створення списку улюблених фільмів має потенціал стати успішним та забезпечити прибуток. З урахуванням швидкого зростання ринку онлайн-кінематографії і попиту користувачів на додаткові можливості для організації своїх улюблених фільмів, цей продукт може знайти свою аудиторію.

Для досягнення успіху необхідно створити функціональний та зручний в користуванні додаток. Крім того, важливо знайти ефективний спосіб монетизації, такий як реклама, підписки або продаж преміум-версії з додатковими функціями.

Проте, існують ризики, які варто враховувати, зокрема конкуренція з іншими додатками, витрати на розробку та маркетинг, а також нестабільність ринку онлайн-кінематографії. Однак, при правильному плануванні та розвитку, додаток може отримати популярність серед широкої аудиторії та стати прибутковим.

Висновки

В результаті виконання дипломної роботи було успішно розроблено програмний продукт, який забезпечує безпечне та зручне керування власною інформацією. Для досягнення цієї мети було виконано наступні кроки:

1) Проведений аналіз наявних рішень щодо розробки Fullsatck веб додатку
2) Вибрано необхідні засоби та інструменти для розробки програмного продукту.
3) Визначено методи розробки.
4) Розроблено архітектуру програмного забезпечення.
5) Здійснено програмну реалізацію продукту для безпечного збереження інформації з використанням обраних технологій програмування.

Розроблений програмний продукт має наступний функціонал:

1) Гнучка БД на основі SQLite
2) Функціонал реєстрації та вторизації в системі
3) Функціонал пошку та додавання фільмів
4) Перегляд списку фільмів

В результаті дипломної роботи було створено програмний продукт, який може зможе допомогти користувачам створити свій власний список улюблених фільмів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Е. Шмидт. Как работает Google, 2015. – 384 с.
2. Н. Прасти. Блокчейн. Разработка приложений, 2016. – 256 с.
3. Л. Лелу. Блокчейн от А до Я. Все о технологии десятилетия, 2008. – 423 с.
4. А. Астрель. Книга шифрів. Таємна історія шифрів і їх розшифровки, 2007. – 446 с.
5. Ф. Бауэр. Расшифрованные секреты. Методы и принципы криптологии, 2007. – 550 с.
6. Офіційна документація React: https://reactjs.org/docs/
7. Офіційний блог React: https://reactjs.org/blog/
8. React Patterns: https://reactpatterns.com/
9. React Cookbook: https://reactrecipes.dev/
10. Офіційна документація Django: https://docs.djangoproject.com/
11. Офіційний блог Django: https://www.djangoproject.com/weblog/
12. Django for Beginners (книга): https://djangoforbeginners.com/
13. Django Girls Tutorial: https://tutorial.djangogirls.org/
14. S. Stoyan. Up & Running: Building Web Applications — Published by O’Reilly Media in Azure, 2016. – 222 с.
15. "Основи React.js" - Артемій Федосеєв
16. "Панування над React" - Адам Хортон
17. "Cookbook React: Створення динамічних веб-додатків з React за допомогою Redux, Webpack, Node.js та GraphQL" - Карлос Сант
18. "RESTful веб-сервіси Django: Найпростіший спосіб створення Python RESTful API" - Гастон С. Хіллар
19. "Two Scoops of Django 3.x: Кращі практики для веб-фреймворку Django" - Даніель Рой Грінфельд та Одрі Рой Грінфельд
20. "Django та інтеграція з JavaScript: AJAX та jQuery" - Джонатан Хейворд
21. "Вивчення React: Функціональна розробка веб-додатків з React та Redux" - Алекс Бенкс та Єва Порселло

Додаток А

Лістинг програмного коду:

```
/// <summary>
/// Модель користувача
/// </summary>

class User(db.Model):
    __tablename__ = "database"
    __searchable__ = ["username"]
    __analyzer__ = StemmingAnalyzer()
    id = db.Column(db.String(32), primary_key=True, unique=True, default=get_uuid)
    username = db.Column(db.String(22), unique=True, nullable=False)
    password = db.Column(db.Text, nullable=False)
    anime = db.relationship('User_List', backref= database, lazy=True)
    friends = db.relationship('User',
                              secondary = friends,
                              primaryjoin = (friends.c.user_id == id),
                              secondaryjoin = (friends.c.friend_id == id),
                              lazy = 'dynamic'
                             )


    def friend_add(self, user):
        if not self.is_friend(user):
            self.friends.append(user)

    def friend_remove(self, user):
        if self.is_friend(user):
            self.friends.remove(user)

    def is_friend(self, user):
        return self.friends.filter(friends.c.friend_id == user.id).count() > 0
```

// Представлення Компоненту авторизації

```
import React, { useState } from 'react';
import axios from 'axios';
import Register from './Register';
import { toast } from 'react-toastify';

function Login(props) {
  const [modalActive, setModalActive] = React.useState(false);
  const [loginForm, setloginForm] = useState({
    username: '',
    password: '',
  });

  function logMeIn(event) {
    axios({
      method: 'POST',
      url: '/login',
      data: {
        username: loginForm.username,
        password: loginForm.password,
      },
    })
      .then((response) => {
        props.setToken(response.data.access_token);
      })
      .then(() => {
        localStorage.setItem('username', loginForm.username);
        window.location.reload();
      })
      .catch((error) => {
        if (error.response) {
          console.log(error.response);
          console.log(error.response.status);
          console.log(error.response.headers);

          toast.error(`${error.response.statusText}`);
        }
      });

    setloginForm({
      username: '',
      password: '',
    });
    // window.location.reload(false);

    event.preventDefault();
  }

  function handleChange(event) {
    const { value, name } = event.target;
    setloginForm((prevNote) => ({
      ...prevNote,
      [name]: value,
    }));
  }

  const handleModal = (e) => {
    e.preventDefault();
```

```
    setModalActive(true);
  };

  return (
    <div className="section__login">
      <h1>Увійти в аккаунт</h1>
      <div className="login__wrapper">
        <form className="login">
          <input
            onChange={handleChange}
            className="login__input"
            type="username"
            text={loginForm.username}
            name="username"
            placeholder="Username"
            value={loginForm.username}
          />
          <input
            onChange={handleChange}
            className="login__input"
            type="password"
            text={loginForm.password}
            name="password"
            placeholder="Password"
            value={loginForm.password}
          />

          <button className="login__btn green-btn" onClick={logMeIn}>
            Войти
          </button>
          <div className="reg__label">
            Нет аккаунта? <button onClick={handleModal}>Реєстрація</button>
          </div>
          <Register active={modalActive} setActive={setModalActive} />
        </form>
      </div>
    </div>
  );
}

export default Login;
```

```
// Представлення головної сторінки
import React from 'react';
import { useDispatch, useSelector } from 'react-redux';
import { Link } from 'react-router-dom';
import { loadfilms, loadSearch } from '../redux/actions/AllfilmAction';
import { getSinglefilm } from '../redux/actions/UserfilmAction';
import Modal from './Modal';
import plus from '../images/plus.png';

function Home() {
  let dispatch = useDispatch();
  const films = useSelector((state) => state.AllfilmReducer.films);

  const [page, setPage] = React.useState(0);
  const [modalActive, setModalActive] = React.useState(false);
  const [filmTitle, setfilmTitle] = React.useState('');
  const [currentfilm, setCurrentfilm] = React.useState('');

  let filmId;

  React.useEffect(() => {
    dispatch(loadSearch);
    console.log(films);
  }, []);

  const handleSearch = (e) => {
    e.preventDefault();
    dispatch(loadSearch(currentfilm));
  };

  const handleAdd = () => {
    dispatch(getSinglefilm(filmId));
  };

  const handleModal = (title) => {
    setModalActive(true);
    setfilmTitle(title);
  };

  return (
    <section className="home">
      <h1 className="home__title">Знайти Фільм</h1>
      <form className="search__form" onSubmit={handleSearch}>
        <input
          value={currentfilm}
          onChange={(e) => setCurrentfilm(e.target.value)}
          type="search"
          placeholder="введіть правильну назву фільму"
          required
          className="film__search"
        />
        <input className="search__submit green-btn" value="Поиск"
type="submit" />
      </form>
      <div className="all__film__wrapper">
        {films.length == 0 ? (
          <div className="empty__wrapper">
            <h3 className="empty__msg">Будь ласка, введіть правильну назву
фільму</h3>
```

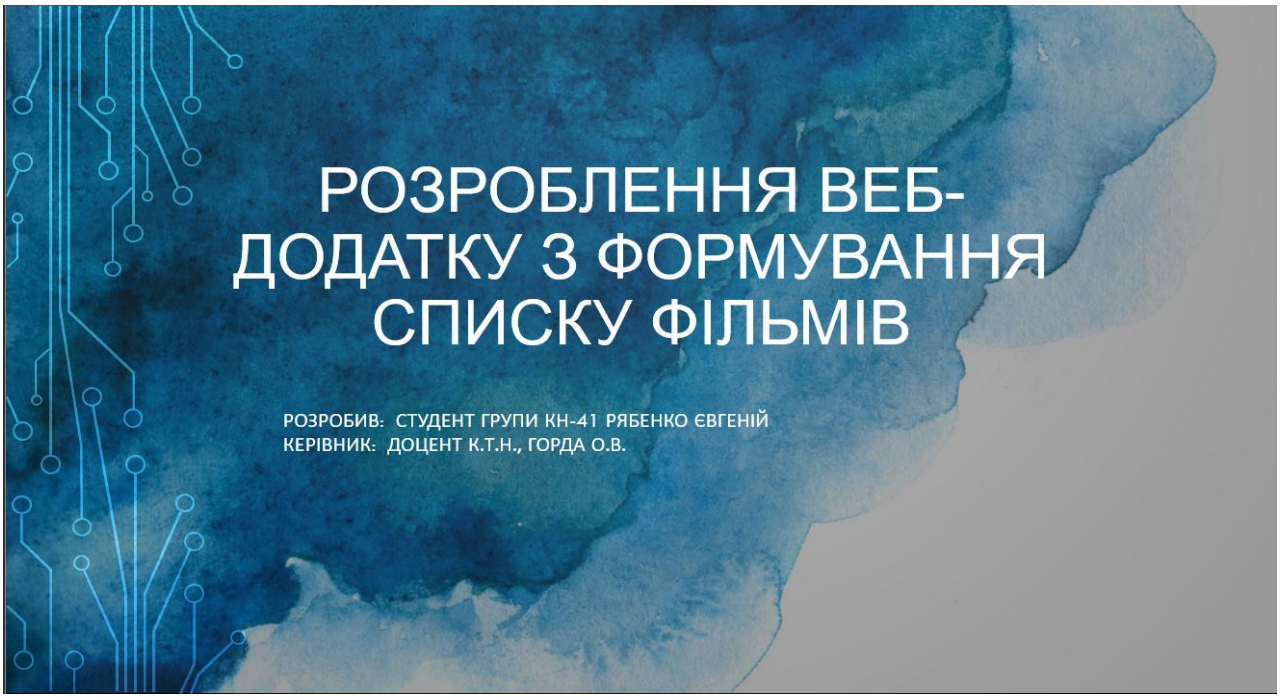
```
          </div>
        ) : (
          films.map((film, id) => (
            <div key={film.mal_id} className="all__film__col">
              <div className="all__film__body">
                <div className="all__film__image">
                  <img src={film.Poster} alt="film" />
                  <button onClick={() => handleModal(film.Name_Ru)}
className="add__film">
                    <img src={plus} alt="+" />
                  </button>
                  <div className="film__text__container">
                    <div className="all__film__title">{film.Name_Ru}</div>
                  </div>
                </div>
              </div>
            </div>
          ))
        )}
      </div>
      <Modal title={filmTitle} active={modalActive} setActive={setModalActive}
/>
    </section>
  );
}

export default Home;



//Метод авторазації на бекенді:
import json
from app import create_app
import bcrypt
from flask_bcrypt import Bcrypt
from flask_sqlalchemy import SQLAlchemy
from flask_jwt_extended import create_access_token, get_jwt, get_jwt_identity,
unset_jwt_cookies, jwt_required, JWTManager
from datetime import datetime, timedelta, timezone
from flask import request, jsonify
from models import User,User_List,db,friends, Animes
import flask_whooshalchemy3



app = create_app()
jwt = JWTManager(app)
bcrypt = Bcrypt(app)

app.config['JWT_SECRET_KEY'] =
"eyJhbGciOiJIUzI1NiJ9.eyJSb2xlIjoiQWRtaW4iLCJJc3N1ZXIiOiJJc3N1ZXIiLCJVc2VybmFt
ZSI6IkphdmFJblVzZSIsImV4cCI6MTY1NDE2MzYyNCwiaWF0IjoxNjU0MTYzNjI0fQ.9io4Bp719ms
GsYw8w_JNWhgLoE4d_pzx5_mJOvd3hrE"
app.config['JWT_ACCESS_TOKEN_EXPIRES'] = timedelta(hours=4)



@app.before_request
```

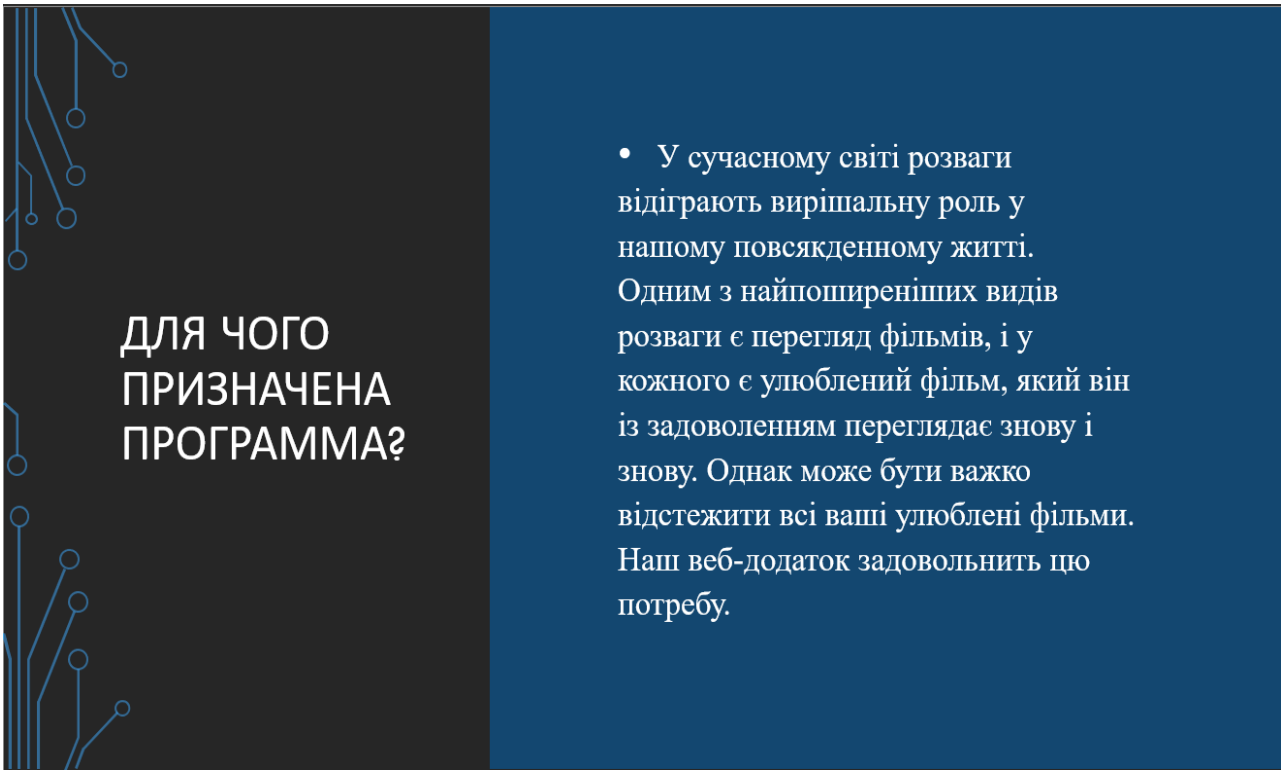
```
def indexing():
    flask_whooshalchemy3.search_index(app, User)
    flask_whooshalchemy3.search_index(app, Animes)



@app.after_request
def refresh_expiring_jwts(response):
    try:
        exp_timestamp = get_jwt()["exp"]
        now = datetime.now(timezone.utc)
        target_timestamp = datetime.timestamp(now + timedelta(hours=12))
        if target_timestamp > exp_timestamp:
            access_token = create_access_token(identity=get_jwt_identity())
            data = response.get_json()
            if type(data) is dict:
                data["access_token"] = access_token
                response.data = json.dumps(data)
        return response
    except (RuntimeError, KeyError):
        # No JWT == return original response
        return response

@app.route("/register", methods=["POST"])
@jwt_required(optional=True)
def register_user():
    jwt_token = get_jwt()
    if type(jwt_token) is not dict:
        return jsonify({"Error": "Can`t register new accout, while being
logged in"}), 409


    username = request.json["username"]
    password = request.json["password"]

    user_exists = User.query.filter_by(username=username).first() is not None

    if user_exists:
        return jsonify({"error": "User already exists."}), 409

    hashed_password = bcrypt.generate_password_hash(password)
    new_user = User(username=username,password=hashed_password)
    db.session.add(new_user)
    db.session.commit()


    return jsonify({
        "id": new_user.id,
        "username": new_user.username
    })

@app.route('/login', methods=["POST"])
def login_user():
    username = request.json.get("username")
    password = request.json.get('password')

    user = User.query.filter_by(username=username).first()

    access_token = create_access_token(identity=username)
```

```
    if user is None:
        return jsonify({"error": "Unathorized"}), 401

    if not bcrypt.check_password_hash(user.password, password):
        return jsonify({'error': "Unathorized"}), 401


    return jsonify({
        "id": user.id,
        "username": user.username,
        "access_token": access_token
    })


@app.route("/logout", methods=["POST"])
def logout():
    response = jsonify({"msg": "logout successful"})
    unset_jwt_cookies(response)
    return response
```

Додаток Б

Презентація:

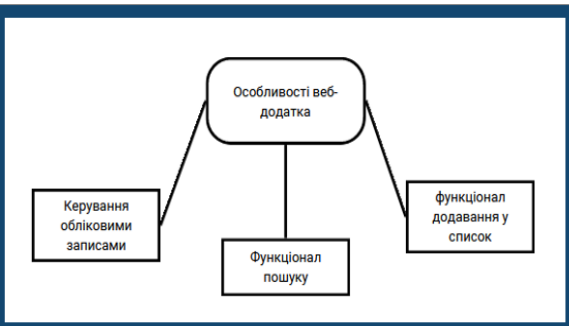




МЕТА ТА ЦІЛІ ДОСЛІДЖЕННЯ

- - **Мета** нашого веб-додатку — полегшити кіноманам складання та ведення власного списку улюблених фільмів. Користувачі можуть додавати та видаляти фільми зі свого списку, давати їм оцінки та писати стислі рецензії за допомогою нашої зручної платформи. Користувачі також можуть миттєво знаходити свої улюблені фільми, шукаючи їх за назвою, жанром, режисером або актором.
- - **Предмет дослідження:** Веб додаток фокусується на царині кіно та його впливі на людей. Він визнає важливість фільмів як форми розваги, мистецтва та культурного самовираження. Вивчаючи вподобання та вибір користувачів, Веб додатокпрагне отримати уявлення про різноманітні смаки та інтереси кінолюбителів. Ця інформація може бути використана для покращення користувацького досвіду, рекомендації відповідних фільмів та створення спільноти кіноманів.
- - **Об'єкт дослідження:** Об'єкт дослідження для веб-додатку, який допомагає користувачам створювати свій список улюблених фільмів, може бути вивчення користувацького поведінки та вподобань у створенні та управлінні списком фільмів. Основні аспекти дослідження можуть включати: Патерни створення списків, цінки та рецензій, пошук фільмів
- - **Методи:** Веб додаток використовує різні методи для досягнення своїх цілей. Ці методи включають: Реєстрація та аутентифікація користувачів; Інтеграція з базами даних фільмів; Створення та організація списківпорядку. Користувачі можуть класифікувати свої списки за темами, жанрами чи особистими вподобаннями.

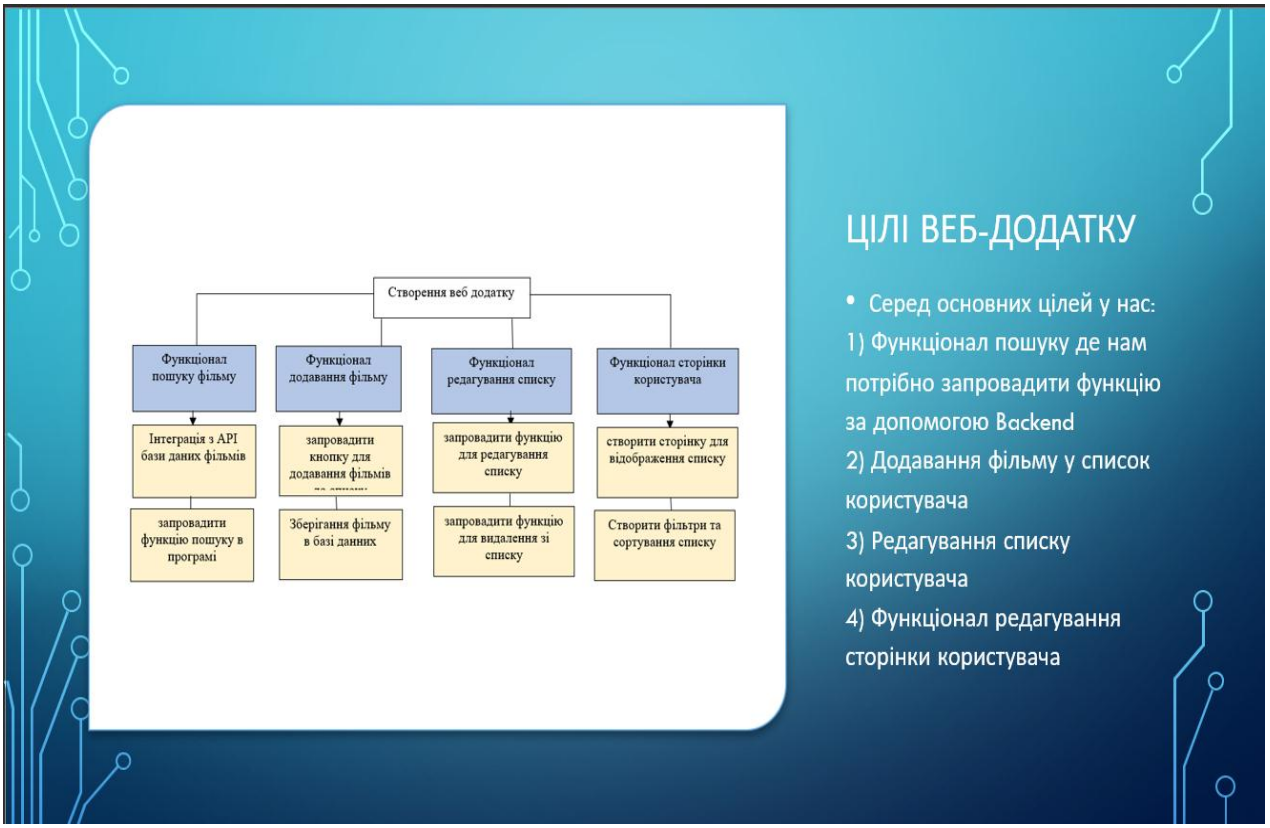
ОСОБЛИВОСТІ НАШОГО ВЕБ-ДОДАТКУ



1) Можливість керувати обліковими записами користувачів є ключовим компонентом кожної онлайн-програми. Створення облікових записів для користувачів повинно бути швидким і простим, в ідеалі з можливістю входу в соціальні мережі. Після створення облікового запису вони повинні мати можливість входити в систему та керувати даними свого облікового запису

2) Функція пошуку — ще одна важлива функція веб-програми для створення списку улюблених фільмів. Користувачі повинні мати можливість шукати фільми за назвою, жанром, режисером або актором. Результати пошуку мають відображати релевантну інформацію про кожен фільм, таку як назва, рейтинг, рік випуску та короткий опис.

3) Додавання системи оцінювання та перегляду є ще одним елементом, який може покращити взаємодію з користувачем. Користувачі повинні мати можливість ранжувати фільми, які вони переглянули. Користувачі повинні мати можливість оцінювати фільми за шкалою від 1 до 10.



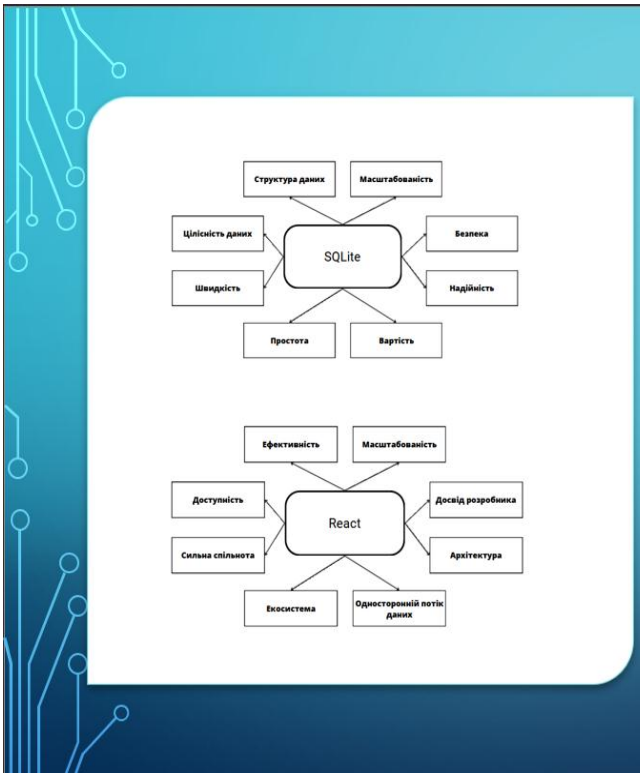
ЦІЛІ ВЕБ-ДОДАТКУ

- Серед основних цілей у нас:

1) Функціонал пошуку де нам потрібно запровадити функцію за допомогою Backend
2) Додавання фільму у список користувача
3) Редагування списку користувача
4) Функціонал редагування сторінки користувача

СУЧАСНІ ВИМОГИ ДО ВЕБ-ДОДАТКІВ

- Існує потреба в безпечних, зручних, доступних і високопродуктивних програмах, що відображається в різноманітності та складності поточних потреб веб-програм. Щоб виправдати очікування сучасних споживачів, розробники повинні враховувати ці фактори під час розробки та створення онлайн-програм.



ВИБІР СУБД ТА FRONT-END ФРЕЙМВОРКУ

- Ми обрали як СУБД SQLite по безліч переваг, але найголовніше це його простота та доступність що є дуже важливими якостями для нашого додатку
- Вибір прийшовся на React як фреймворку для нашого додатку тому що це на даний момент найсучасніша бібліотека JS з якою можна швидко розробляти веб-додатки

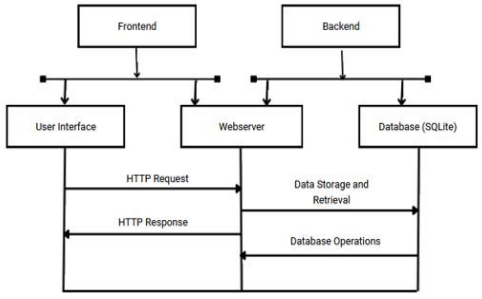
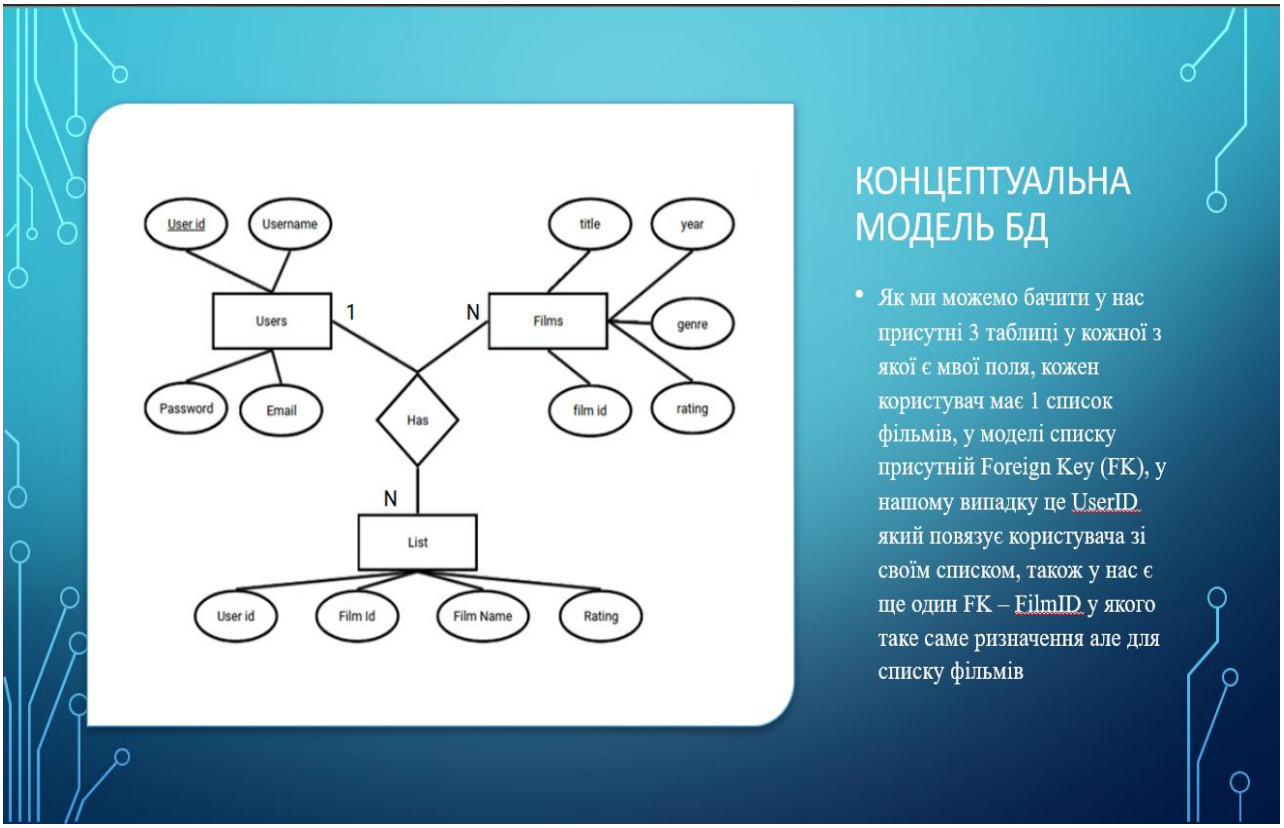
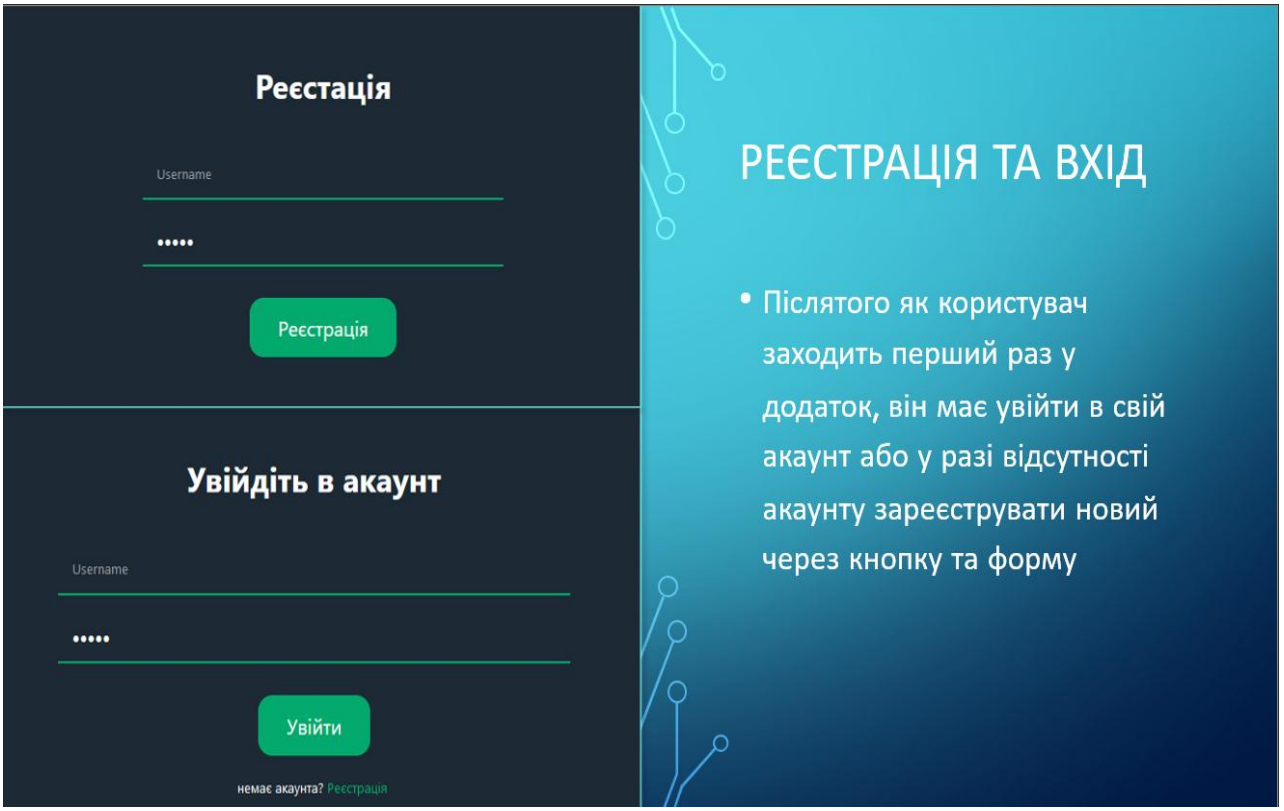


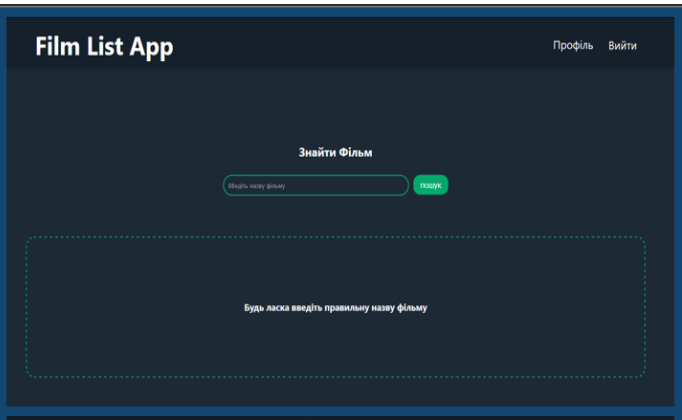
СХЕМА РОБОТИ ВЕБ-ДОДАТКА

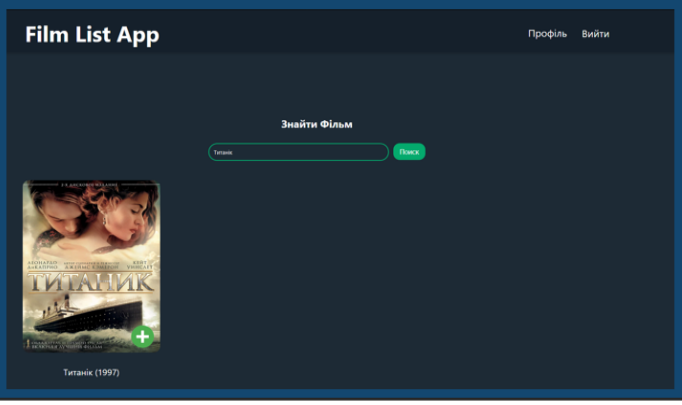
- Діаграма ілюструє взаємодію між зовнішнім (на стороні клієнта) і серверним (на стороні сервера) компонентами веб-додатку, а також взаємодію з веб-сервером і базою даних.


User id
Username
Users
1
N
Films
title
year
genre
Password
Email
Has
film id
rating
N
List
User id
Film Id
Film Name
Rating
КОНЦЕПТУАЛЬНА МОДЕЛЬ БД
• Як ми можемо бачити у нас присутні 3 таблиці у кожної з якої є мвої поля, кожен користувач має 1 список фільмів, у моделі списку присутній Foreign Key (FK), у нашому випадку це UserID який повязує користувача зі своїм списком, також у нас є ще один FK – FilmID у якого таке саме ризначення але для списку фільмів


Реєстація
Username
•••••
Реєстрація
Увійдіть в акаунт
Username
•••••
Увійти
немає акаунта? Реєстрація
РЕЄСТРАЦІЯ ТА ВХІД
• Післятого як користувач заходить перший раз у додаток, він має увійти в свій акаунт або у разі відсутності акаунту зареєструвати новий через кнопку та форму

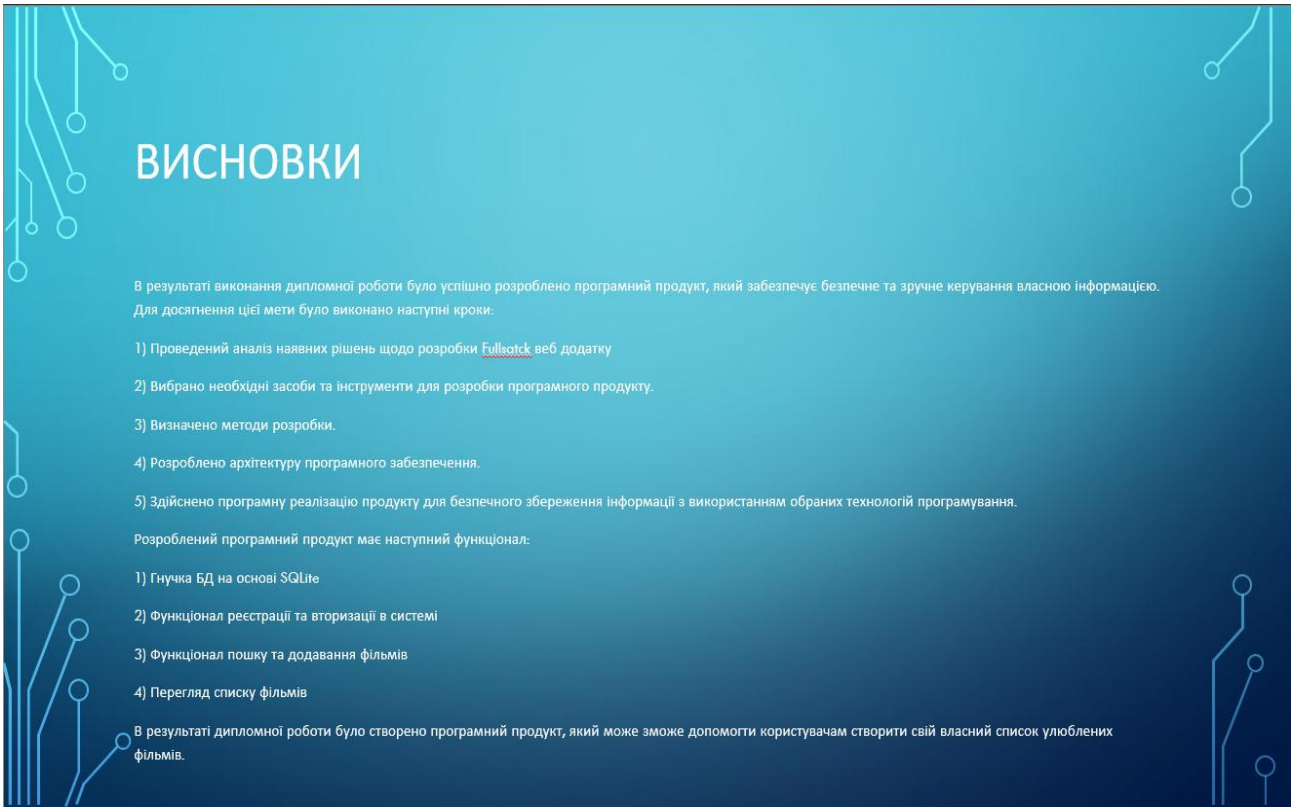
ГОЛОВНА СТОРІНКА ДОДАТКУ

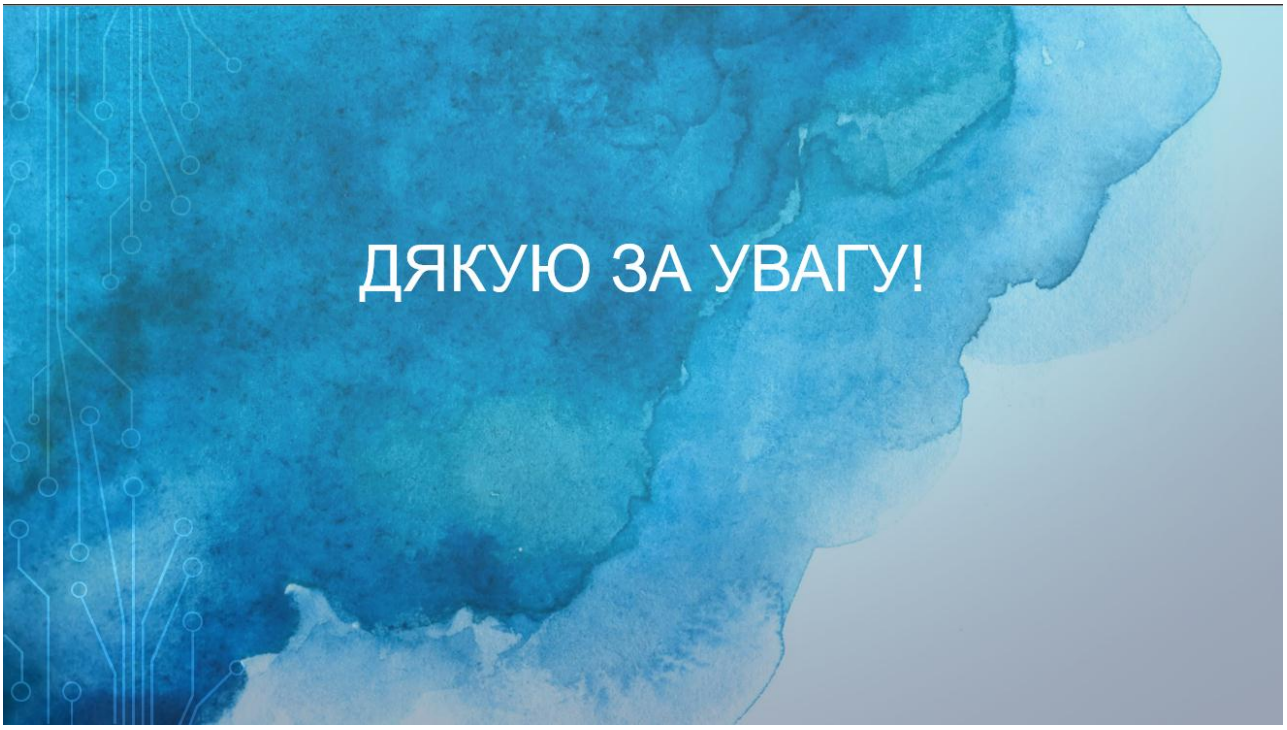




СТОРІНКА КОРИСТУВАЧА

- Після додавання фільму ми переходимо на сторінку користувача де можемо побачити всі додані фільми з можливістю редагувати оцінку та статус або видалити фільм зі списку у разі потреби


ВИСНОВКИ
В результаті виконання дипломної роботи було успішно розроблено програмний продукт, який забезпечує безпечне та зручне керування власною інформацією. Для досягнення цієї мети було виконано наступні кроки:
1) Проведений аналіз наявних рішень щодо розробки Fullsatck веб додатку
2) Вибрано необхідні засоби та інструменти для розробки програмного продукту.
3) Визначено методи розробки.
4) Розроблено архітектуру програмного забезпечення.
5) Здійснено програмну реалізацію продукту для безпечного збереження інформації з використанням обраних технологій програмування.
Розроблений програмний продукт має наступний функціонал:
1) Гнучка БД на основі SQLite
2) Функціонал реєстрації та вторизації в системі
3) Функціонал пошку та додавання фільмів
4) Перегляд списку фільмів
В результаті дипломної роботи було створено програмний продукт, який може зможе допомогти користувачам створити свій власний список улюблених фільмів.


ДЯКУЮ ЗА УВАГУ!