Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерної інженерії та інформаційних систем

КВАЛІФІКАЦІЙНА РОБОТА

бакалавр
Освітній рівень

Інформаційна система для ознайомлення з розважальним контентом
Назва теми

КвРІСТ 200179.20.01.02 ПЗ
Шифр

Галузь знань *12 «Інформаційні технології»*
Шифр, назва

Спеціальність *126 «Інформаційні системи та технології»*
Шифр, назва

Освітня програма *«Інформаційні системи та технології»*
Назва

Виконав: студент IV курсу, група ІСТ-20-1 _____ В.А. Дударчук
Підпис Ініціали, прізвище

Керівник _____ П. Г. Регіда
Підпис, дата Ініціали, прізвище

Нормоконтролер _____ І.О. Засорнова
Підпис, дата Ініціали, прізвище

До захисту допускаю:
Зав. кафедри комп'ютерної інженерії та інформаційних систем _____ Т.О. Говорущенко
Підпис Ініціали, прізвище

«12» червня 2024 р.

Хмельницький 2024

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

Освітній рівень БАКАЛАВР

Галузь знань 12 ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

Спеціальність 126 ІНФОРМАЦІЙНІ СИСТЕМИ ТА ТЕХНОЛОГІЇ

Освітня програма «ІНФОРМАЦІЙНІ СИСТЕМИ ТА ТЕХНОЛОГІЇ»

ЗАТВЕРДЖУЮ

Зав. кафедри Т.О.Говорущенко

"16" 01 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

Дударчуку Валерію Анатолійовичу

Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Інформаційна система для ознайомлення з розважальним контентом

Керівник проекту (роботи) Регіда П.Г., ст.. викладач.

Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 15.02.2024 р. № 8

2. Строк подання студентом проекту (роботи) на кафедру 03.06.2024 р.

3. Вихідні дані до проекту (роботи) Завдання на кваліфікаційну роботу

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

Аналоги інформаційних систем для перегляду розважального відео-контенту

Проєктування інформаційної системи для ознайомлення з розважальним контентом

Програмна реалізація інформаційної системи

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)

Схема взаємодії компонентів інформаційної системи

Структура компонентів користувацького інтерфейсу

Взаємодія користувача з сайтом

6. Консультанти розділів дипломного проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Засорнова І.О., доцент кафедри КІС		
Антиплагіат	Нічепорук А.О., доцент кафедри КІС		

7. Дата видачі завдання «_10_»_____01_____2024 р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітка
1	Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи з керівником	10.01.2024	виконано
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження	01.02.2024	виконано
3	Робота над розділом 1 – дослідження предметної області та постановка задачі	01.03.2024	виконано
4	Робота над розділом 2 –вибір програмних засобів для реалізації інформаційної системи	01.04.2024	виконано
5	Робота над розділом 3 – проєктування інформаційної системи для ознайомлення з розважальним контентом	30.04.2024	виконано
6	Оформлення пояснювальної записки згідно вимог	31.05.2024	виконано
7	Попередній захист ВКР	30.05.2024	виконано
8	Захист ВКР на засіданні ЕК	Червень 2024 року	

Студент _____ В.А.Дударчук
Підпис Ініціали, прізвище

Керівник роботи _____ П. Г. Регіда
Підпис Ініціали, прізвище

№ рядка	формат	Позначення	Найменування	Кіл. листів	№ екз	Примітка
			Текстові документи			
1		КвРІСТ 200179.20.01.02 ПЗ	Пояснювальна записка	60		
			Графічні матеріали			
2		КвРІСТ 200179.20.01.02 Е8	Схема взаємодії компонентів інформаційної системи	1		
3		КвРІСТ 200179.20.01.02 Е8	Структура компонентів користувацького інтерфейсу	1		
4		КвРІСТ 200179.20.01.02 Е8	Схема взаємодії користувача з сайтом	1		

				КвРІСТ 200179.20.01.02 ВП	
Зм	Арк	№ докум	Підпис	Дата	
Розробив		Дударчук			Відомість проекту
Перевір.		Регіда			
Н. контр.		Засорнова			
Затв.		Говорущенко		12.06	

Літера	Аркуш	Аркушів
У	1	1

ХНУ, ІСТ-20-1

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Інформаційна система для ознайомлення із розважальним контентом».

Автор роботи: Дударчук Валерій Анатолійович.

Керівник роботи: Регіда Павло Генадійович.

Пояснювальна записка: 60 с., 39 рис., 3 табл., 4 дод., 60 джерел.

Графічна частина: 3 креслення.

ІНФОРМАЦІЙНА СИСТЕМА, ОБЛІКОВИЙ ЗАПИС, БАЗА ДАНИХ, КОНТЕНТ.

Метою дипломної роботи є забезпечення функціонування інформаційної системи для перегляду розважального контенту, яка реалізована у вигляді веб-сайту, що забезпечує зручний та ефективний доступ до різноманітних фільмів, серіалів та інших видів медіа контенту.

Об'єктом дослідження є процес функціонування інформаційної системи для перегляду розважального контенту.

Предметом дослідження є інформаційна система. для перегляду розважального контенту.

Під час проведення даного дослідження були використані методи веб-орієнтованого програмування, програмування серверних додатків що використовують документні бази даних.

Підпис студента

03.06.2024

Дата

ЗМІСТ

ВСТУП

Початок ери онлайн-прокату фільмів можна відслідкувати до кінця 1990-х років, коли з'явилися перші інтернет-платформи, що пропонували можливість замовлення фільмів на відеокасетах або DVD. Користувачі могли вибирати фільми з онлайн-каталогів, замовляти їх та отримувати поштовим шляхом. Це дозволяло глядачам переглядати фільми у власному домі, обходячи необхідність виходу до кінотеатрів або оренди відеокасет у магазинах.

Проте справжній прорив відбувся з появою стрімінгових платформ у 2000-х роках. Завдяки швидкому розвитку Інтернету та технологій стало можливим дивитися фільми та серіали безпосередньо через Інтернет, не завантажуючи їх на свій пристрій. Це відкрило нові можливості для глядачів, які тепер могли насолоджуватися переглядом відеоконтенту у будь-який зручний час та на будь-якому пристрої, забезпечивши велику гнучкість і доступність.

Компанія Netflix стала піонером у цьому напрямку. Вона була заснована у 1997 році як сервіс оренди DVD, але згодом перейшла до стрімінгового контенту. У 2007 році Netflix запустив свій стрімінговий сервіс, який надавав користувачам необмежений доступ до широкого каталогу фільмів та серіалів. Це було значним кроком у розвитку онлайн-прокату фільмів, оскільки воно дало можливість глядачам миттєво переглядати відеоконтент без необхідності чекати на доставку фізичних носіїв.

Пізніше до ринку вступили інші компанії, такі як Amazon Prime Video, Hulu, Disney+, HBO Max, які також пропонують великий вибір фільмів та серіалів для стрімінгового перегляду. Це призвело до зростання конкуренції на ринку та появи нових функцій та сервісів для привертання аудиторії.

Розвиток технологій та зміни в споживчому підході: З появою швидкого Інтернету та розширенням доступу до високошвидкісних мереж, споживачі отримали можливість споживати контент безпосередньо через Інтернет, без необхідності завантажувати або чекати на фізичні носії.

Вплив на кінематограф та розвиток кіноіндустрії: Зростання популярності стрімінгових платформ вплинуло на кінематограф та спосіб, яким фільми та серіали створюються, розповсюджуються та споживаються. Більше компаній і творчих особистостей вирішують випускати свій контент через ці платформи, що робить ринок більш різноманітним і конкурентоспроможним.

Відповідь на змінні потреби аудиторії: Стрімінгові платформи не лише забезпечують доступ до великого каталогу фільмів і серіалів, але й пропонують оригінальний контент, створений спеціально для платформи.

Міжнародний вплив та глобалізація: Стрімінгові платформи стають все більш глобальними, надаючи доступ до свого контенту на різних мовах та для різних культур.

Технологічні інновації та вирішення проблем доступності: Стрімінгові платформи сприяють технологічним інноваціям, які полегшують доступ до контенту для людей з обмеженими можливостями або тих, хто проживає у віддалених регіонах.

Метою дипломної роботи є забезпечення функціонування інформаційної системи для перегляду розважального контенту, яка реалізована у вигляді веб-сайту, що забезпечує зручний та ефективний доступ до різноманітних фільмів, серіалів та інших видів медіа контенту.

Об’єктом дослідження є процес функціонування інформаційної системи для перегляду розважального контенту.

Предметом дослідження є інформаційна система. для перегляду розважального контенту.

Під час проведення даного дослідження були використані методи веб-орієнтованого програмування, програмування серверних додатків що використовують документні бази даних.

1 АНАЛОГИ ІНФОРМАЦІЙНИХ СИСТЕМ ДЛЯ ПЕРЕГЛЯДУ РОЗВАЖАЛЬНОГО ВІДЕО-КОНТЕНТУ

1.1 Аналіз YouTube

YouTube є однією з найпопулярніших та найбільших відеоплатформ у світі. Заснований у 2005 році, цей сервіс став першим великим сайтом для обміну відео в Інтернеті. Починаючи зі свого заснування, YouTube став ключовим медіа-каналом, де користувачі можуть безкоштовно завантажувати, переглядати та ділитися відеоконтентом різного роду.

Різноманітність контенту: На YouTube можна знайти величезний різноманітний контент. Від музичних відео, трейлерів фільмів та відеоігор до навчальних відео, кулінарних шоу та відеоблогів. Велика частина контенту на YouTube створюється самими користувачами, що дозволяє будь-кому стати контент-креатором та поділитися своїми ідеями, творчістю та знаннями з усім світом.

Соціальні функції: Однією з головних особливостей YouTube є можливість взаємодії з контентом та іншими користувачами. Користувачі можуть підписуватися на канали, виражати вподобання та не вподобання, залишати коментарі, ділитися відео та надсилати особисті повідомлення. Це створює активну спільноту користувачів, яка обговорює відео, обмінюється думками та враженнями.

Можливості для контент-креаторів: YouTube є важливим ресурсом для творчих особистостей та контент-креаторів, які можуть заробляти гроші на своїх відео. Платформа пропонує різні способи монетизації контенту, включаючи партнерські програми, рекламу та спонсорські угоди. Це робить YouTube привабливим місцем для творчих людей, які шукають можливість заробляти на своїй творчості та будувати власну аудиторію.

Глобальний вплив: Завдяки своїй широкій доступності та глобальному впливу, YouTube став важливою платформою для розповсюдження культурних,

соціальних та політичних ідей. Він дозволяє користувачам з усього світу спілкуватися та обмінюватися інформацією, що сприяє розвитку глобального співтовариства та розширенню горизонтів розуміння та співпраці.

Очевидно, що існує великий обсяг відеоконтенту, яким йому доводиться щодня керувати. Для цього використовується MySQL та різні системи управління базами даних у різних місцях, щоб забезпечити безперебійну роботу YouTube. Більшість даних YouTube зберігається в модульних дата-центрах Google. Модульний дата-центр є мобільним і може бути розміщений будь-де, де потрібен простір для зберігання даних. Оскільки YouTube був придбаний компанією Google у 2006 році, цілком логічно, що дані YouTube зберігаються в модульних дата-центрах Google. Загалом використовується 5 дата-центрів Google, які YouTube використовує разом із власною мережею розповсюдження контенту (CDN), щоб забезпечити постійний доступ до даних для кінцевих користувачів. Найпопулярніші відео переміщуються до CDN, яка реплікує їх у різні місця. Це означає, що користувач може отримати до них доступ набагато швидше і з меншою кількістю переходів. З іншого боку, менш популярні відео зберігаються на серверах YouTube, де до них можна отримати доступ на вимогу. Крім того, не існує жорсткого правила, що відео зберігаються в дата-центрі, найближчому до географічного регіону, з якого вони вийшли. Наприклад: Якщо ви завантажуєте відео на YouTube з Індії, ваші дані можуть зберігатися в дата-центрі у Великобританії. На додаток до всіх цих методів, Youtube також використовує хмарні сховища. Спочатку MySQL в основному використовувалася в базах даних YouTube для зберігання більшості даних, починаючи від відео і закінчуючи метаданими, такими як користувачі, теги та описи. Для баз даних використовувався змінний тип даних, який дозволяв зберігати відео та зображення. Однак, недоліком MySQL є те, що вона має мало можливостей для масштабування, що є дуже важливим фактором для таких компаній, як YouTube, які постійно розширюються. Однак YouTube не може повністю відмовитися від MySQL, тому Vitess використовується в поєднанні з MySQL. Vitess - це система

кластеризації баз даних, яка поєднує в собі багато важливих функцій MySQL з масштабованістю, яка є торговою маркою баз даних NoSQL. Vitess допомагає консолідувати запити до YouTube у менші партії, які набагато легше обробляти та виконувати. Він також створює резервні копії та масштабує, якщо це необхідно.

Дані, зібрані YouTube, також аналізуються для персоналізованого показу реклами. Google використовує алгоритми для збору всієї інформації про користувача, такої як історія браузера та пошукових запитів, географічна інформація тощо. Ці алгоритми аналізують інформацію, щоб зрозуміти, в яких продуктах чи послугах можуть бути зацікавлені користувачі. Потім компанії платять за рекламу своїх продуктів на YouTube, використовуючи Adwords і Adsense, які відстежують кількість кліків на цю рекламу. Використовуючи цей алгоритм, користувачі отримують таргетовану рекламу відповідно до своїх вподобань, а рекламодавці можуть гарантувати, що їхні продукти потрапляють до тих, хто може бути зацікавлений у їх купівлі.

1.2 Огляд Amazon Prime Video

Amazon Prime Video створила високодоступну архітектуру потокового відео в реальному часі, об'єднавши надлишкові компоненти для досягнення п'яти-дев'яти рівнів доступності, які вони вимагають для своєї платформи. Компанія використовувала поглиблений моніторинг для пом'якшення проблем ще до того, як вони почали впливати на клієнтів. Вони також оптимізували топологію розгортання і кодування відео, щоб зменшити витрати, забезпечуючи при цьому оптимальну якість відео для користувачів.

Prime Video розпочала трансляцію відео в прямому ефірі у 2015 році і поступово розширила свою пропозицію до більш ніж 1 000 телеканалів, що охоплюють новини, спорт та розваги.

Архітектура платформи для прямих трансляцій [1] поєднує в собі компоненти AWS Media Services (рисунок 1.1). AWS Elemental MediaConnect - це

високоякісний транспортний сервіс для живого відео. AWS Elemental MediaLive - це сервіс обробки відео в реальному часі, який може кодувати відеоконтент для трансляції та потокового передавання. AWS Elemental MediaPackage готує і розподіляє відеоконтент на різні підключені пристрої в різних форматах.

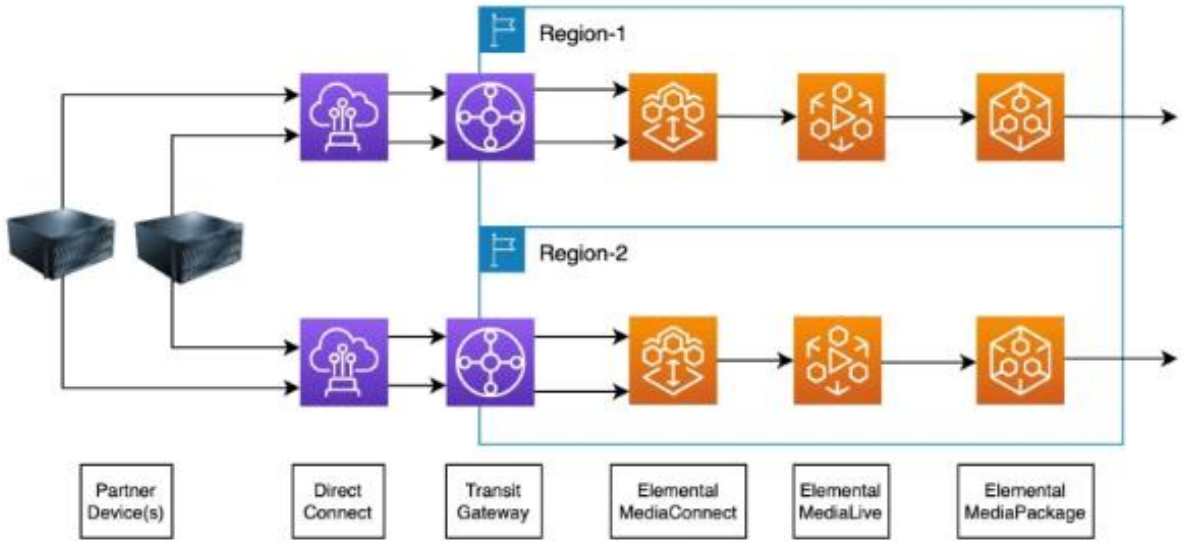


Рисунок 1.1 - Архітектура платформи для прямих трансляцій

Prime Video [2] розміщує стек обробки та розподілу відео в двох окремих регіонах для кожного зовнішнього джерела і використовує пряме підключення та транзитний шлюз для забезпечення мережевого зв'язку між пристроями партнерів і платформою. Кожен партнер подає два джерела сигналу (основний і резервний) у кожен регіон, що забезпечує відмовостійкість, якщо одне з них стає недоступним. Компоненти обробки медіа та мережеві компоненти розташовані послідовно в кожному регіоні.

Враховуючи загальну доступність компонентів у послідовному A = Ax * Ay та паралельному A = 1 - (1 - Ax) * (1 - Ay), а також той факт, що послуги AWS Elemental пропонують 99,9% SLA, можна розрахувати загальну доступність медіакомпонентів платформи [1] Prime Video: A = 1 - (1 - 99.9% * 99.9% * 99.9%) * (1 - 99.9% * 99.9% * 99.9%) = 99.999%. Завдяки розгортанню надлишкових

стеків обробки/розподілу в окремих регіонах, Prime Video може забезпечити п'ять-дев'ять ступенів доступності, використовуючи сервіси з нижчими рівнями SLA[1].

Component	SLA	Redundancy	Total SLA	Downtime per Month (s)	Downtime per Month (min)
MediaConnect	99.9000%	1	99.9000%	2,628	43.80
MediaLive	99.9000%	1	99.9000%	2,628	43.80
MediaPackage	99.9000%	1	99.9000%	2,628	43.80
Overall SLA of components in series			99.700%	**7,876**	**131.27**
Overall SLA of components in parallel	**99.700%**	**2**	**99.999%**	**24**	**0.39**

Рисунок – 1.2 Розрахунки загальної доступності

Навіть при наявності архітектури, яка може забезпечити високу доступність, важливо активно відстежувати стан і продуктивність всіх її компонентів, щоб швидко реагувати, якщо будь-які ключові показники не відповідають очікуваному рівню, що потенційно може призвести до погіршення якості обслуговування. Крім того, збір метрик дозволяє відстежувати загальні тенденції та складати звітність SLO (Service Level Objective).

Команда Prime Video прагне зменшити витрати на інфраструктуру, обираючи найбільш оптимальні регіони AWS для розміщення стеку обробки/розповсюдження медіа на основі розташування джерела сигналу партнера або використовуючи магістральну мережу AWS для надійної доставки даних або прийому сигналу в межах AWS, якщо це можливо. Вони також налаштовують інфраструктуру та конфігурації і оптимізують адаптивне кодування бітрейту на основі аналізу даних.

Пошукова оптимізація (SEO) Коли велика кількість користувачів шукає певний товар на Amazon у певний момент часу, він починає займати лідируючі позиції в рекламі, системах рекомендацій, а також в оголошеннях Facebook та Instagram, які вони показують своїм користувачам, у пошуковій системі Amazon

та інших пошукових системах, таких як Google, Bing тощо. Він використовує статистичний машинний переклад (SMT), який полягає в тому, що він аналізує мільйони документів на предмет відповідності пошуковим запитам. Він використовує методи SEO, щоб займати перші місця в результатах пошуку, щоб підвищити свою продуктивність

Alexa також є частиною Amazon, яка використовує машинне навчання, щоб передбачити, для чого користувач буде запитувати інформацію, а потім на основі цього надає багатий користувацький досвід, коли користувач задає питання. Alexa також використовує штучний інтелект, який покращує розпізнавання мови і допомагає збирати всі моделі голосових даних для цілей регресії та аналізу даних. Дані та машинне навчання, що використовуються Amazon Alexa, відіграють важливу роль у роботі її інтелектуальних функцій обробки природної мови (NLP) та генерації природної мови (NLG), які розуміють людську мову, а потім відповідають на неї відповідно до тону користувача та контексту, в якому він звертається до неї. Alexa Voice Services (AVS) - це, по суті, набір алгоритмів машинного навчання, які забезпечують роботу Amazon Alexa, присутніх по всьому світу, і вони регулярно оновлюються для надання найсвіжішої інформації. Крім того, Amazon Alexa здатна відповідати на запитання голосом відомих знаменитостей за допомогою технологій глибокого навчання.

Серверна частина: Java використовується обробляють запити користувачів, взаємодіють з базою даних, керують сеансами користувачів. Заст API розробка: які дозволяють різним компонентам та службам взаємодіяти між собою. Також реалізації бізнес-логіки платформи, такої як обробка підписок користувачів, управління контентом, рекомендаційні системи. Таким чином, Java використовується на різних рівнях платформи Amazon Prime Video. Також використовується Spring Framework який надає інструменти для управління залежностями та інверсії контролю, що дозволяє керувати залежностями між компонентами додатку, зберігати код додатку зрозумілим та легко змінюваним.

1.3 Розгляд Disney+

Disney+ - це стрімінгова платформа від компанії Disney, яка надає доступ до широкого асортименту відео контенту включаючи фільми, серіали, мультфільми, документальні фільми та багато іншого. Ключові аспекти платформи Disney+:

Disney+ пропонує широкий вибір відео контенту від компанії, також пропонує ексклюзивний відео контент, який доступний лише на цій платформі, підтримує різні пристрої, включаючи смартфони, планшети, комп'ютери, смарт-телевізори та ігрові консолі, що дозволяє користувачам переглядати вміст на будь-якому зручному пристрої, надає користувачам персоналізовані рекомендації, можуть завантажувати вміст на свої пристрої.

Disney+ на свої власні платформи та спростити процес тестування та контролю якості. ADK(Application Development Kit) доступний у вигляді програмного пакету, який містить усі інструменти, що можуть знадобитися дистриб'юторам для створення додатків на різних платформах пристроїв. Disney+ ADK використовують у випадках, коли наші партнери мають розгорнуту платформу і потребують низки бібліотек та певної мови програмування, або для включення передового обладнання, яке може взаємодіяти з відповідною вбудованою системою, щоб представити своїм користувачам досвід Disney+. ADK пропонує загальний спосіб для більш ніж тисячі різних комбінацій розгортання і дозволяє нашим командам Disney нормалізувати ринок і розширити можливості дистриб'юторів, зберігаючи при цьому зосередженість і гнучкість наших команд у роботі над наступним набором функцій і продуктів.

Технічні деталі ADK(Application Development Kit)

Тепер, є план як надавати Disney+ ADK дистриб'юторам, потрібно було заглибитися в технічні особливості реалізації ідей.

Основні компоненти, серед яких були такі:

Native Client Platform v2 Runtime: написана на C'99 і розгорнута у вигляді двійкового коду на телевізійних приставках та ігрових консолях.

Native Video Engine: написаний на C++'98, названий Disney Streaming Native Video Engine (DSS-NVE)

Рівень абстракції платформи: для апаратного портування під назвою Steamboat

Двійковий бінарник бігуна програми: написаний на C'99 під кодовою назвою Merlin

Клієнтська програма Disney+: Клієнтська програма Disney+. Написаний на Rust, скомпільований у WebAssembly, віддалено розміщений в AWS [3]

Різні компоненти були розроблені з самого початку, щоб дозволити нам підтримувати кілька дистриб'юторів і кілька клієнтських додатків без необхідності спеціальних реалізацій клієнтських додатків (рисунок 1.3) (звідси абстракція з рівнем виконання та апаратного перенесення).

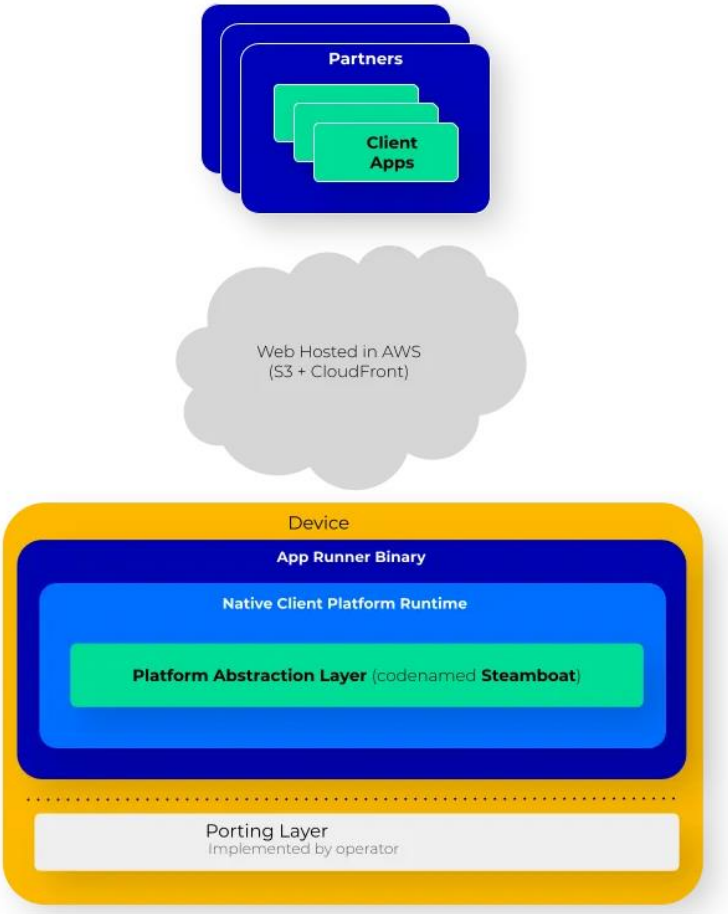


Рисунок 1.3 - Модель взаємої компонентів

Як це працює: Найпростіший спосіб зрозуміти рішення ADK - це порівняти його з традиційним рішенням на основі браузера [3]. У браузерному рішенні є два основні компоненти:

1. Рушій браузера: цей компонент містить підкомпонент відеоплеєра і написаний на скомпільованій мові, наприклад, на C. Рушій браузера забезпечує обробку та відтворення мультимедійного контенту, управління пам'яттю, роботу з мережею та інші низькорівневі функції.

2. Логіка програми: це HTML5+JavaScript, що відповідає за інтерактивність, відображення контенту, обробку подій користувача та інші високорівневі задачі. Логіка програми визначає, як користувач взаємодіє з додатком і яким чином відображається інформація.

У цій архітектурі рушій браузера надається постачальником платформи (наприклад, Google для Chrome, Mozilla для Firefox), а сам додаток створюється компанією, такою як Disney Streaming.

ADK (Application Development Kit), навпаки, використовує іншу архітектуру(рисунок 1.4):

1. Ядро на скомпільованій мові: у ADK основне ядро написане на мові C і містить інтерпретатор WebAssembly (WASM). Це ядро забезпечує основні функції програми та обробку ресурсів на низькому рівні. Крім того, воно відповідає за ефективне виконання та інтеграцію програмних компонентів, забезпечуючи стабільну і безпечну роботу системи.

2. WASM (WebAssembly): WebAssembly є відкритим стандартом, що представляє собою портативний формат двійкового коду. Він дозволяє завантажувати і виконувати високопродуктивний код через Інтернет. WASM працює подібно до того, як браузер завантажує HTML і JavaScript. Він дозволяє завантажувати будь-який логічний або користувацький контент, такий як зображення або інші ресурси, віддалено, забезпечуючи високу швидкість виконання та ефективність.

Переваги ADK над традиційною архітектурою

Оскільки основне ядро ADK написане на скомпільованій мові C, це дозволяє досягти вищої продуктивності порівняно з інтерпретацією JavaScript у браузері. Використання WebAssembly також забезпечує швидке виконання коду, близьке до рідного машинного коду, що дозволяє ефективно використовувати ресурси системи.

ADK дозволяє розробникам легше оновлювати та розширювати функціональність додатків. Завдяки WebAssembly, нові модулі та функції можуть бути завантажені та інтегровані в систему без необхідності перевипускати весь додаток. Це спрощує процес розробки та розгортання оновлень.

WebAssembly має ряд вбудованих механізмів безпеки, які забезпечують захист від зловмисних дій. Він працює в ізольованому середовищі, що обмежує можливості коду виконувати небажані дії на хості, забезпечуючи додатковий рівень безпеки порівняно з традиційним JavaScript.

WebAssembly є кросплатформенним і підтримується всіма основними браузерами, що дозволяє забезпечити високу сумісність додатків на різних пристроях та операційних системах. Це робить ADK універсальним інструментом для розробки високопродуктивних додатків, які можуть працювати на різних платформах.

ADK представляє сучасний підхід до розробки додатків для потокового мовлення, поєднуючи переваги високої продуктивності, гнучкості, безпеки та сумісності. Використання WebAssembly дозволяє створювати потужні та ефективні додатки, які можуть швидко реагувати на запити користувачів і забезпечувати високу якість обслуговування.

Таким чином, ADK є потужним інструментом, що забезпечує розробникам можливість створювати продуктивні, безпечні та гнучкі додатки з широкою сумісністю, що відповідають сучасним вимогам ринку.

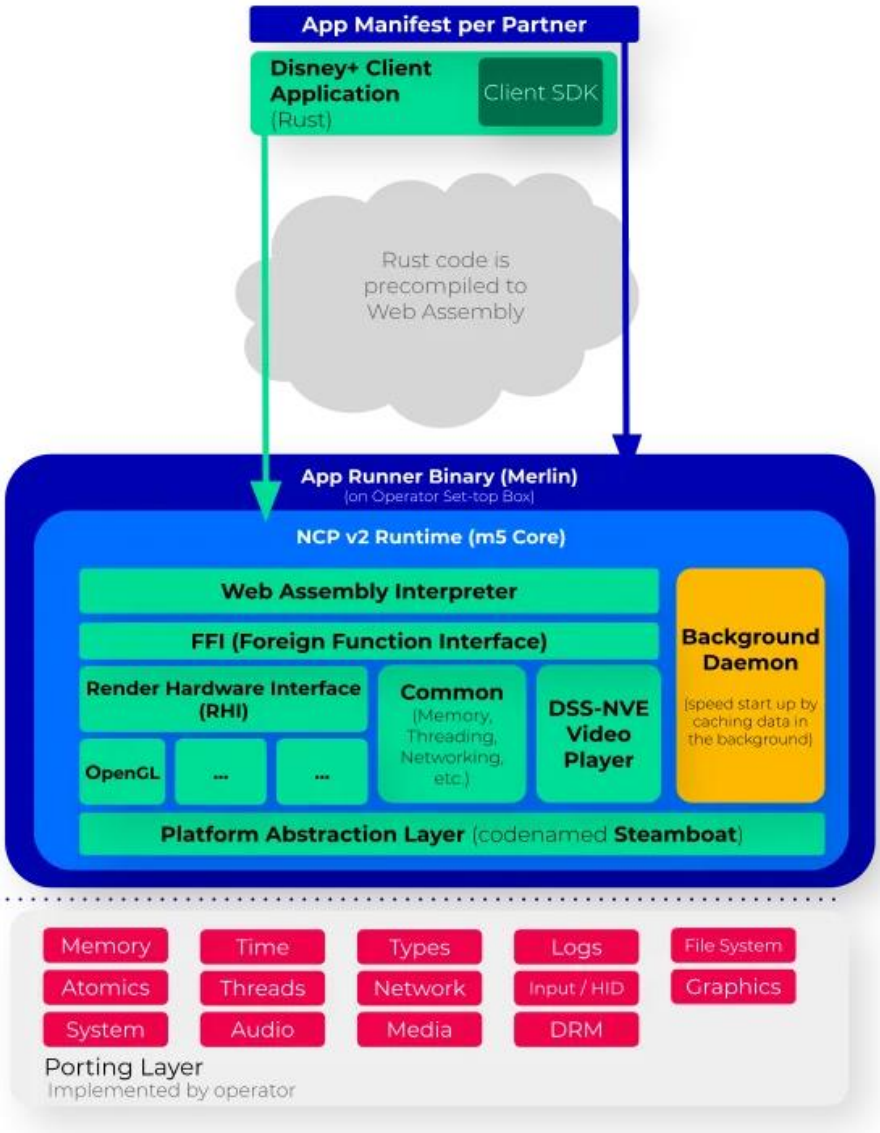


Рисунок 1.4 - Архітектура веб-додатку

React.js, Node.js та MongoDB використовуються на платформі Disney+. React.js використовується для створення динамічних та інтерактивних компонентів веб-інтерфейсу, Node.js використовується для створення серверної частини платформи Disney+, яка обробляє запити від клієнтів, взаємодіє з базою даних та надає функціональність веб-додатків, таких як автентифікація користувачів, обробка платежів, управління контентом. MongoDB використовується для зберігання різноманітних даних таких як інформація про користувачів, список фільмів та серіалів, налаштування категорій вмісту, історія

перегляду. Вони взаємодіють між собою для створення повноцінної платформи Disney+.

1.4 Аналіз HBO Max

HBO Max - це популярна стрімінгова платформа від компанії WarnerMedia, яка надає доступ до широкого асортименту відео контенту, включаючи фільми, серіали, анімацію та документальні програми. Ключові аспекти платформи HBO Max:

Пропонує широкий вибір відео контенту, ексклюзивний відео контент, який доступний лише на цій платформі, підтримує різні пристрої, включаючи смартфони, планшети, комп'ютери, смарт-телевізори, ігрові консолі та інші пристрої, що дозволяє користувачам переглядати вміст на будь-якому зручному пристрої, також надає користувачам персоналізовані рекомендації на основі їхніх вподобань та історії перегляду [4]. Користувачі можуть завантажувати вміст на свої пристрої (рисунок 1.5).

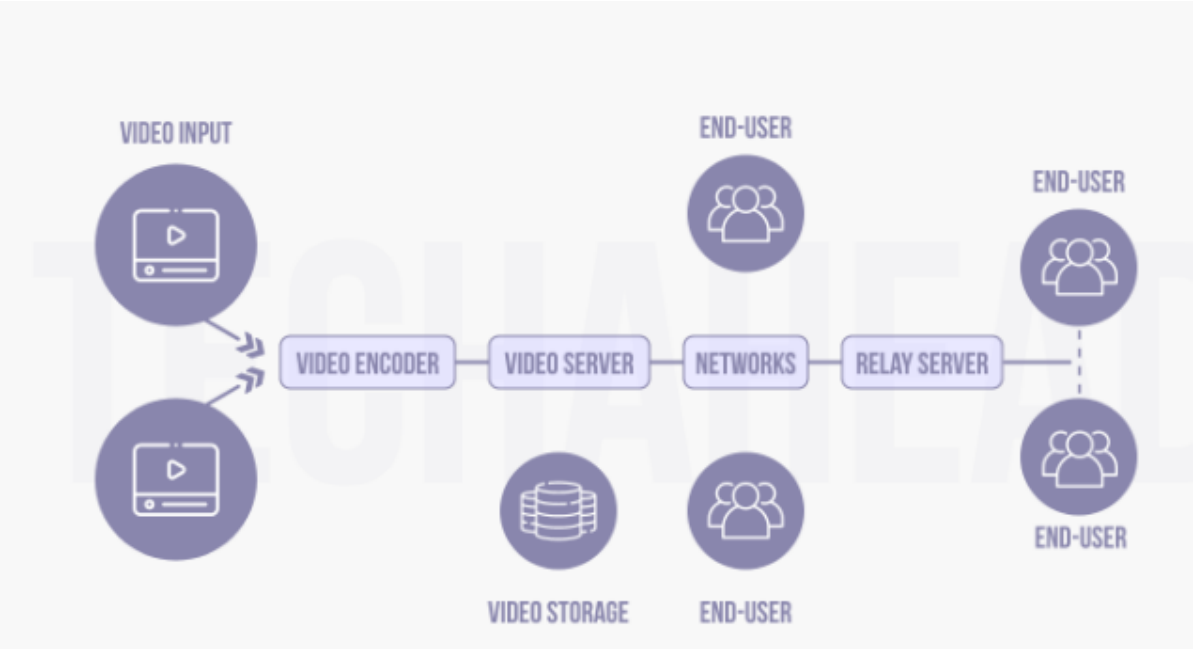


Рисунок 1.5 - Архітектура додатка відеострімінга

Фреймворк, який використовує HBO Max, створений компанією BAMTech, яку раніше придбала Disney. Таким чином, і HBO Max, і Disney використовують одну й ту саму платформу для доставки відеоконтенту своїм користувачам.

Згідно з Ottball, HBO Max використовує багато власного програмного забезпечення та систем, розроблених їхньою командою. Сюди також входить власна система трюків і власний відеоплеєр, також використовує мульти-DRM, комерційні та власні рішення для відеоаналітики (рисунок 1.6).

HBO Max використовував HLS v4 і v5 з 6-секундними фрагментами для відео- та аудіопрофілів, які дуже схожі на MPEG-DASH.

Водночас субтитри передаються через WebVTT, що гарантує безперешкодний доступ до субтитрів як для користувачів Android, так і для iPhone.

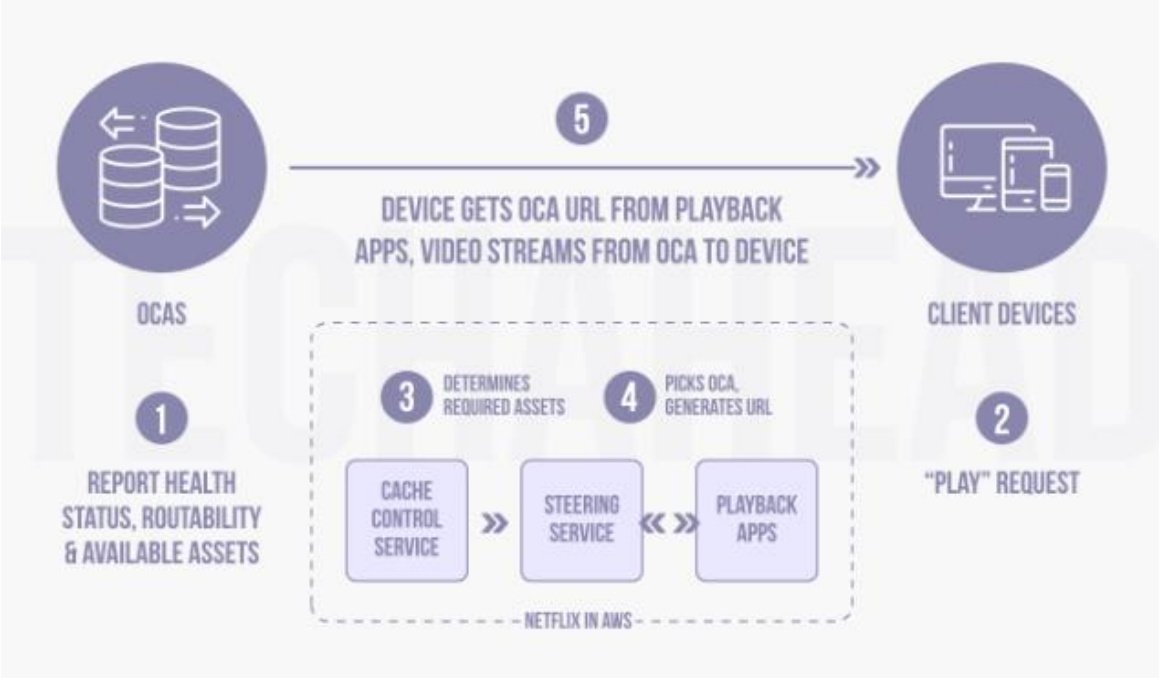


Рисунок 1.6 - Схема OCA для доставки контенту

HBO Max використовує Unified Streaming Platform v1.9.5, яка створює маніфести HLS. З останнім оновленням програми у 2021 році вони

використовують вдосконалену версію, оскільки Unified Streaming Platform v1.9.5 використовується з 2018 року.

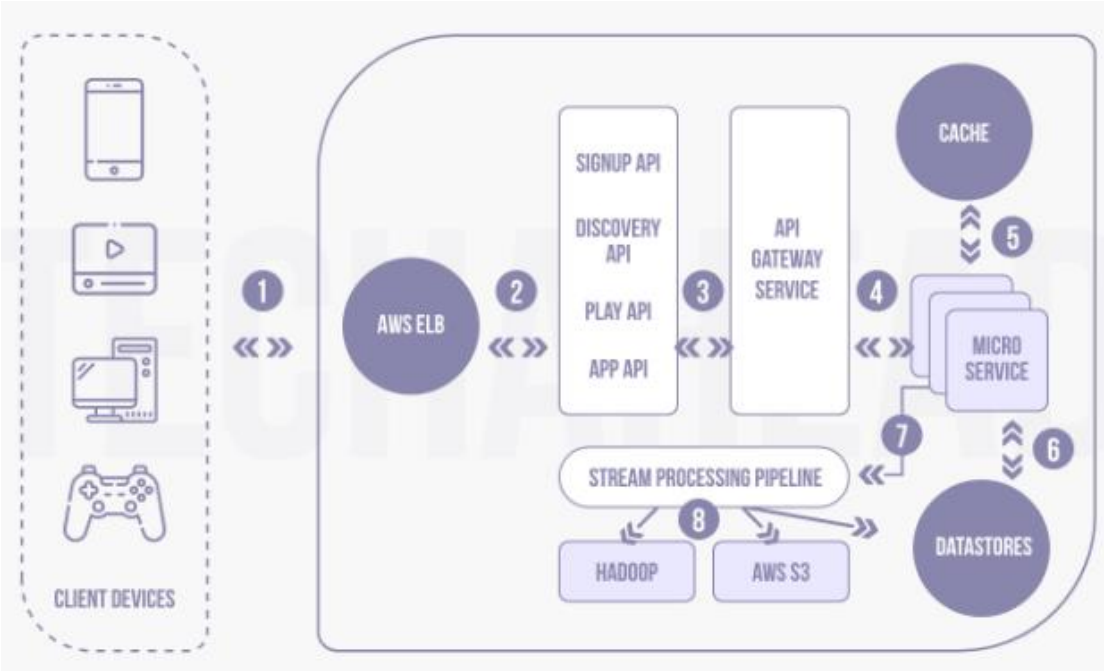


Рисунок 1.7 -Взаємодія з девайсами

Масштабованість має вирішальне значення. Здатність керувати такою величезною кількістю одночасних користувачів, не порушуючи їхнього загального досвіду та поведінки.

Застосовують деякі прийоми масштабування, які роблять потокове мовлення безперебійним і якісним для величезної аудиторії,

Однією з таких є перекодування відео, коли формат відео конвертується в інший формат, так що:

Всі користувачі, на всіх пристроях і в будь-якому місці, отримують однаковий, чудовий досвідЗменшується буферизація, що призводить до безперебійного перегляду (рисунок 1.7).

Низька затримка, що забезпечує високу якість потокового відео

Ось триступеневий процес перекодування відео:

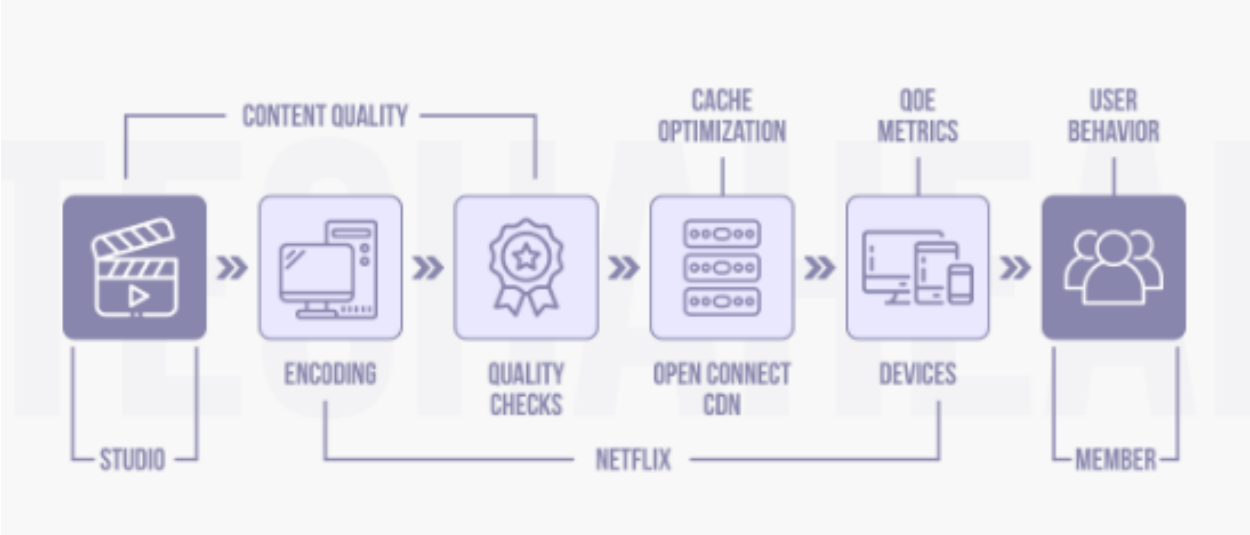


Рисунок 1.8 - Архітектура кодування відео

Перевірка: Під час конвертації відеоформатів деякі елементи можуть змінюватися, наприклад, кадри, кольори тощо. За допомогою процесу валідації система гарантує, що процес перекодування буде безперебійним (рисунок 1.8).

Паралельне кодування: Зазвичай відео в наші дні мають розмір і об'єм у кілька терабайт, і немає сенсу обробляти таку величезну кількість даних. Тому відео спочатку розбивають на менші фрагменти, а потім кодують їх паралельно. Для забезпечення такої паралельної обробки та кодування фрагментів потрібно кілька серверів. Після перевірки ці фрагменти знову об'єднуються назад.

Профіль кодування: Це секрет масштабованості, що реалізується за допомогою перекодування відео. Кожен пристрій має відеоформат, який забезпечує оптимальну продуктивність залежно від швидкості мережі та підключення. Такі програми, як HBO Max, перекодовують і обробляють відео на основі відеоформату, який підтримується цим пристроєм. Це забезпечує бездоганну масштабованість і продуктивність.

Серверна частина: Angular - це фреймворк для розробки веб-додатків, розроблений компанією Google. Він використовується для побудови клієнтської частини веб-додатків, використовується на платформі HBO Max для створення динамічного та інтерактивного інтерфейсу користувача. Він дозволяє розробникам швидко створювати складні компоненти, також забезпечує хорошу

швидкодію та масштабованість. ASP.NET використовується для створення надійної та масштабованої серверної частини. Він дозволяє розробникам ефективно керувати запитами користувачів, забезпечуючи швидку відповідь та високу доступність платформи, має багатий набір інструментів для розробки, тестування та управління веб-додатками. Microsoft Azure використовується для надання інфраструктури хмарних послуг. Azure, масштабує свої серверні потужності в залежності від потреб, забезпечуючи високу доступність та надійність платформи для користувачів. Крім того, Azure надає широкий набір інструментів для моніторингу, керування та захисту додатків, що дозволяє забезпечити високий рівень безпеки та ефективності. Таким чином, Microsoft Azure допомагає забезпечити успішну та безперебійну роботу у хмарному середовищі.

Angular, ASP.NET, Microsoft AzureІнтеграція та взаємодія: Angular, ASP.NET та Microsoft Azure взаємодіють між собою для створення повноцінної платформи HBO Max, яка надає користувачам зручний та ефективний спосіб отримання доступу до великого обсягу відео контенту.

Висновки

Під час виконання цієї роботи було вивчено і проаналізовано існуючі аналоги сайтів розважального контенту, такі як YouTube, Amazon Prime Video, HBO Max та Disney+. Цей аналіз дозволив виявити ключові аспекти їхньої функціональності, інтерфейсу та взаємодії з користувачем, що стало важливою основою для розробки нової системи. Детально розглянуто механізми навігації, методи рекомендаційного пошуку та системи управління користувачами, які застосовуються в цих платформах. Це дозволило зрозуміти, як створити інтуїтивно зрозумілий і зручний інтерфейс для майбутніх користувачів.

Крім того, проведено детальне дослідження базових принципів логіки та булевої алгебри, які лежать в основі створення ефективних інформаційних систем. Виявлено, що застосування цих принципів дозволяє значно підвищити продуктивність та надійність системи, що особливо важливо для обробки великих обсягів даних та забезпечення швидкої реакції на запити користувачів. На основі цього дослідження визначено оптимальні методи створення та оптимізації логічних та булевих функцій, які можуть бути застосовані у розробці інформаційної системи для розважального контенту.

Такий підхід дозволив не лише розширити теоретичні знання, а й отримати практичні навички в розробці ефективних алгоритмів та структур даних для покращення користувацького досвіду у таких системах. Застосування цих знань на практиці дозволило створити систему, яка не лише відповідає сучасним вимогам користувачів, але й перевершує їх очікування завдяки високій швидкості роботи, стабільності та зручності використання. Таким чином, відбулась інтеграція передових технологій та підходів в нову інформаційну систему, що робить її конкурентоспроможною на ринку розважального контенту.

2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ОЗНАЙОМЛЕННЯ З РОЗВАЖАЛЬНИМ КОНТЕНТОМ

2.1 Постановка задачі

Завдяки широкій популярності потокового відеоконтенту сучасні споживачі мають можливість переглядати широкий спектр фільмів, серіалів та іншого відеоконтенту в будь-який час і з будь-якого місця. Такі популярні веб-сайти, як Netflix, відомі своєю величезною бібліотекою контенту, а також інтуїтивно зрозумілим користувацьким інтерфейсом та індивідуальним обслуговуванням клієнтів. Зважаючи на зростаючий попит на платформи для потокового мовлення, необхідно розробляти нові сервіси зі схожими функціями. Створення платформи, яка б задовольняла сучасні потреби клієнтів і була життєздатним фінансовим проектом, є схожим на Netflix. Однак, щоб гарантувати зручність для користувачів і безперебійну роботу платформи, створення власної стрімінгової платформи потребує інтегрованої стратегії розробки програмного забезпечення на додаток до використання сучасних інструментів і технологій. Однією з головних проблем є розробка інтерактивного інтерфейсу, який би забезпечував легкий доступ до контенту, реєстрацію та дозвіл користувачів, а також персоналізацію контенту на основі вподобань користувачів. Крім того, необхідно вирішити питання зберігання та ефективного адміністрування бази даних користувачів і відеоматеріалів, а також гарантувати високий рівень безпеки і конфіденційності даних користувачів. Метою цього дослідження є розробка передової платформи для потокового мовлення, яка б відповідала сучасним тенденціям розвитку онлайн-відеосервісів і пропонувала споживачам широкий вибір відеоматеріалів, а також персоналізований досвід перегляду.

Основне завдання - створити корисний інструмент, який дозволить широкому колу користувачів легко отримати доступ до різноманітного відеоконтенту. Забезпечте зручний інтерфейс: Щоб користувачі могли легко знаходити та відображати потрібний контент, необхідно розробити інтуїтивно

зрозумілий та зручний інтерфейс. Впровадження процесу авторизації на практиці: Створіть систему авторизації, яка дозволить користувачам легко отримати доступ до своїх облікових записів через електронну пошту та інші відомі сайти, такі як Google і GitHub. Управління базою даних: Створити надійну систему управління базами даних на основі MongoDB для користувачів та відеоматеріалів, щоб гарантувати надійне зберігання даних та швидкий пошук інформації. Безпека даних: Забезпечити процедури для захисту конфіденційності та особистої інформації користувачів, гарантуючи високий ступінь захисту від зловживань. Розширена функціональність: Надавати користувачам інструменти, які спрощують додавання в обране та подальший перегляд відеоконтенту, надаючи їм легкий доступ до управління своїм контентом. Ефективність і стабільність: Переконайтеся, що платформа працює ефективно і стабільно, навіть коли нею користується багато користувачів.

2.2 Вимоги до інформаційної системи

Функціональні

Функціональні потреби для інформаційної системи включають кілька важливих аспектів, які зроблять перегляд відеоконтенту зручнішим і приємнішим для глядачів. По-перше, авторизація та реєстрація користувачів. Користувачі повинні мати можливість зареєструвати обліковий запис в інформаційній системі, вказавши свою електронну адресу та пароль. Користувачі також повинні мати можливість увійти за допомогою облікових записів Google або GitHub.

Після успішної авторизації користувач повинен мати доступ до особистого кабінету, де він може керувати своїм профілем і налаштуваннями. Однією з основних функцій платформи є управління контентом. Список усього доступного відеоконтенту повинен бути видимим для користувачів, які можуть фільтрувати його за різними критеріями, такими як рейтинг і жанр. Користувачі повинні мати можливість зберігати відео у списку обраного для подальшого перегляду.

Адміністратор також повинен мати можливість відстежувати інформацію і статистику щодо користувачів і того, як вони використовують платформу, а також додавати, редагувати і видаляти відеоконтент.

Персоналізація та пропозиції - одна з ключових особливостей платформи. Користувачі повинні мати можливість обирати відео на основі аналізу їхніх вподобань, проведеного системою. Клієнти повинні бачити рекомендації, які відповідають їхнім інтересам і точкам зору.

Нефункціональні

Щоб гарантувати ефективність і задоволення потреб клієнтів, платформа для потокового мовлення повинна вирішити кілька важливих питань, які не пов'язані з основними функціями. Серед них першочерговою необхідністю є виняткова продуктивність. Інформаційні система для перегляду розважального контенту повинна працювати швидко та ефективно, щоб скоротити час очікування користувачів та надавати швидкі відповіді на їхні запити.

Висока доступність також є ще однією важливою умовою. Платформа повинна завжди бути доступною для користувачів без будь-яких збоїв або перебоїв. Важливо забезпечити надійність і зміцнити довіру користувачів до стабільності платформи.

Безпека. Персональні дані користувачів повинні бути надійно захищені на платформі завдяки використанню найсучасніших методів шифрування та захисту від зловмисників.

Локалізація та інтернаціоналізація платформи також мають велике значення. Платформа повинна бути готова до різних мовних і культурних контекстів, щоб забезпечити зручність для глобальних користувачів. Це включає не тільки переклад інтерфейсу на різні мови, але й адаптацію контенту відповідно до культурних особливостей і уподобань користувачів у різних регіонах. Важливо також враховувати правові вимоги кожної країни, де працює платформа, і дотримуватися місцевих норм і правил.

2.3 Програмні технології для інформаційної системи

Для розробки клієнтської частини веб-додатку, яка відповідає за інтерактивний інтерфейс та відображення вмісту, використовуються наступні інструменти:

Фреймворк Next.js (рисунок 2.1) спрощує створення веб-додатків, особливо тих, що використовують React. Next.js [8, 14, 19, 21, 24] пропонує дуже цікавий набір функцій, саме тому ми вирішили використати його для створення фронтенду проекту. Використання рендерингу на стороні сервера (SSR) є однією з головних переваг Next.js. Це означає, що веб-сторінки можуть відображатися на сервері на додаток до клієнта. Це збільшує швидкість завантаження сторінок і підвищує їхню SEO-ефективність. Крім того, Next.js дозволяє статичну генерацію (SSG). Під час створення проекту ви можете заздалегідь створити статичні файли сторінок. Ця стратегія може значно полегшити навантаження на сервер і підвищити продуктивність програми. Зручні можливості маршрутизації в Next.js також спрощують роботу з різними сторінками та URL-адресами. Крім того, він поставляється з вбудованою підтримкою TypeScript, що підвищує зручність і безпеку команди розробників. Загалом, використання Next.js [48, 59, 60] допомагає нам створити потужний та ефективний фронтенд для онлайн-додатку. Можливо швидко створювати та оновлювати проект завдяки складним функціям та зручному інтерфейсу, що гарантує високу продуктивність та задоволення користувачів.

Рисунок 2.1 – Логотип Next.JS

Створювати динамічні та інтерактивні веб-додатки за допомогою фреймворку користувацького інтерфейсу React. (рисунок 2.2) Для створення інформаційної системи використовують React [59, 55, 44, 47, 33] для створення клієнтської частини веб-додатку, оскільки він надає низку корисних інструментів та функцій. Компонентний підхід до розробки інтерфейсу користувача - одна з ключових особливостей React [6, 7, 12, 22, 26]. Для створення складних користувацьких інтерфейсів потрібно почати з простих компонентів, які обробляють різні аспекти інтерфейсу та об'єднують їх разом. React має велику спільноту розробників і безліч корисних фреймворків та інструментів. Можна використовувати такі бібліотеки, як Material-UI для створення естетично привабливих користувацьких інтерфейсів, Redux для керування станом додатку та React Router для маршрутизації. Крім того, React має вбудовану підтримку JSX, розширення JavaScript [5, 23, 27] яке дозволяє описувати компоненти інтерфейсу користувача, використовуючи синтаксис, подібний до HTML. Це полегшує розробку та підтримку складних проектів, роблячи код легшим для читання та розуміння.

Рисунок 2.2 – Логотип React

У проекті TypeScript (рисунок 2.3) використовується для покращення підтримки коду та автозавершення, а також для гарантування безпеки та зручності розробки.

Можливість створювати типи для змінних, функцій та об'єктів є однією з основних можливостей TypeScript [29, 14, 10, 37, 43]. Оскільки компілятор TypeScript перевіряє типи даних протягом усього етапу розробки, це допомагає вам знаходити проблеми і допомагає в процесі.

Крім того, використовуючи типи, TypeScript робить код більш читабельним і зрозумілим. Завдяки цьому зменшується кількість помилок, а розробники-початківці можуть швидше засвоїти код. Використання найновіших функцій JavaScript, включаючи асинхронізацію/очікування та деструктуризацію об'єктів, також можливе за допомогою TypeScript, зберігаючи при цьому безпеку введення даних.

Рисунок 2.3 – Логотип TypeScript

Tailwind CSS (рисунок 2.4) у проекті, щоб надати сторінкам та компонентам шикарного та сучасного вигляду. Має кілька переваг, однією з яких є його атомарний підхід до стилів. Замість того, щоб створювати власний код CSS, можна використовувати готові класи, які пропонують різні стилі для різних частин [11, 41, 30, 50, 51]. Це допомагає нам розробляти інтерфейси швидко та ефективно, без потреби у великій кількості спеціального CSS коду. Він надає безліч утиліт-класів, які дозволяють вам робити багато різних речей, включаючи вирівнювання, розміщення, розмір елементів та багато іншого.

Це дає вам безліч можливостей створювати різні види дизайну без необхідності писати спеціальний код CSS [12]. Дозволяє легко змінювати стилі за допомогою конфігураційних файлів, що дозволяє нам розробляти оригінальні стилі або змінювати вже існуючі відповідно до вимог проекту.

Рисунок 2.4 – Логотип TailWind CSS

Axios – це бібліотека для здійснення HTTP-запитів з JavaScript, яка дозволяє нам взаємодіяти зі зовнішніми API та серверами. У проекті ми використовуємо Axios (рисунок 2.5) для виконання запитів до серверної частини додатку. Однією з основних переваг Axios є його простота та зручність використання. Він надає нам чистий та зрозумілий API для здійснення різних видів запитів, таких як GET, POST, PUT та DELETE, а також дозволяє нам легко передавати дані та обробляти відповіді. Крім того, Axios [31] автоматично обробляє різні аспекти HTTP-запитів, такі як обробка помилок, встановлення заголовків, обробка куків та інше. Це дозволяє нам зосередитися на логіці додатку, не турбуючись про деталі роботи з HTTP-запитами.Також Axios підтримує використання обіцянок (promises) та асинхронного програмування, що дозволяє нам легко управляти асинхронними операціями та здійснювати запити до сервера відповідним чином.

Рисунок 2.5 – Логотип Axios

MongoDB

Документно-орієнтована база даних з масштабованістю та гнучкістю для зберігання даних – це MongoDB (рисунок 2.6). Використовують MongoDB для

зберігання даних про користувачів, канти та фільми в проекті. Гнучкість структури даних MongoDB [32, 22, 40, 45, 57] є однією з її переваг. За допомогою цієї функції ви можете зберігати дані в документах з різними полями та форматами. Це звільняє нас від необхідності жорстко встановлювати структуру даних і дозволяє легко зберігати різноманітні типи даних, включаючи тексти, цілі числа, масиви та об'єкти.

MongoDB підтримує розподілене зберігання даних, що дозволяє розміщувати базу даних на декількох серверах і автоматично дублювати дані, гарантуючи їх доступність і надійність. Також доступні потужні функції маніпулювання даними, такі як пошук, сортування, фільтрація та агрегування. Можлива швидка взаємодіяти з базою даних і отримувати необхідну інформацію завдяки зручній мові запитів.

Рисунок 2.6 – Логотип MongoDB

Інструментом, який використовується для взаємодії з базою даних MongoDB, є Prisma ORM (рисунок 2.7). Моделі, поля та зв'язки між ними, які складають структуру даних бази даних, визначені в цьому файлі.

Визначаємо моделі даних у schema.prisma, даючи їм імена та поля. Наприклад, можна вказати такі поля, як "id", "name", "email" та інші для моделі "User". Кожне поле також може мати свій тип даних, наприклад, "String", "Int", "DateTime" тощо.

В schema.prisma ми можемо вказати зв'язки "один-до-багатьох" або "багато-до-багатьох" між моделями. В результаті можна створювати складні структури даних і встановлювати зв'язки між різними елементами бази даних.

Рисунок 2.7 – Логотип Prisma

Node.js (рисунок 2.8) є середовищем виконання JavaScript на стороні сервера, що дозволяє створювати швидкі та масштабовані додатки. Використання Node.js [22] дозволяє побудувати ефективний та продуктивний серверний шар.

Рисунок 2.8 – Логотип Node.js

Express.js (рисунок 2.9) - це популярний веб-фреймворк для Node.js, який дозволяє створювати маршрути, обробники запитів та керувати динамічним контентом. Express забезпечує швидку та просту реалізацію серверної логіки та API.

Рисунок 2.9 – Логотип Express.js

Mongoose (рисунок 2.10) - це об'єктно-документний абстракційний шар (ODM) для Node.js і MongoDB, який дозволяє працювати з даними MongoDB у зручний та продуктивний спосіб. Він надає засоби для моделювання даних та виконання операцій з базою даних.

Рисунок 2.10 – Логотип Mongoose

Passport.js (рисунок 2.11) - це модуль для аутентифікації користувачів у Node.js додатках. Він дозволяє реалізувати різні стратегії аутентифікації, такі як введення логіну та пароля, аутентифікація через сторонні сервіси (наприклад, Google або GitHub) та інші.

Рисунок 2.11 – Логопит Passport.js

2.4 Взаємодія між компонентами системи

Ініціалізація запиту на клієнтській стороні відбувається, коли користувач взаємодіє з веб-інтерфейсом вашого сайту. Це може бути натискання кнопок, введення тексту в поля, відправлення форм тощо. Коли ці дії відбуваються, JavaScript на клієнтській стороні ініціює відправку HTTP-запиту на сервер за допомогою вбудованих функцій або бібліотек, таких як fetch() або XMLHttpRequest. Після ініціалізації запиту на клієнтській стороні, цей запит надсилається на сервер через мережу Інтернет. На сервері веб-додатку запит приймається серверним програмним забезпеченням, яке постійно слухає на певному порту (наприклад, порті 80 для HTTP або порті 443 для HTTPS). Сервер перехоплює цей запит і починає його обробку. Цей процес є першим кроком взаємодії між клієнтом і сервером і відбувається при кожній взаємодії користувача з веб-сайтом. Після отримання запиту сервер починає процес обробки. Це включає аналіз запиту для розпізнавання інформації, переданої в запиті, такої як HTTP-метод (GET, POST, PUT, DELETE тощо), URL-адреса, на яку надійшов запит, та параметри запиту. Після аналізу сервер виконує відповідну логіку обробки. Це може включати перевірку даних, взаємодію з базою даних, виконання розрахунків тощо. Після обробки запиту сервер починає процес генерації відповіді. Це включає формування відповідних даних або результатів операцій, які будуть включені у відповідь клієнту. Наприклад, сервер може витягти дані з бази даних, обчислити необхідні значення або взяти інші дії в залежності від запиту. Після цього відповідь готується до відправлення клієнту. Після того, як сервер згенерував відповідь, він надсилає її назад клієнту через мережу Інтернет.

Це зазвичай відбувається за допомогою протоколу HTTP, де відповідь включає HTTP-статус відповіді, заголовки та вміст відповіді. Клієнт (браузер) приймає цю відповідь і готується до обробки. Після отримання відповіді клієнтська частина (фронтенд) оновлює інтерфейс відповідно до отриманих даних або результатів операції. Це може включати оновлення вмісту сторінки,

відображення повідомлень про статус операції або автоматичне перенаправлення на іншу сторінку. Наприклад, якщо користувач відправив форму, то може відобразитися повідомлення про успішну відправку або про помилку, а також може бути відображена нова інформація, отримана з сервера (рисунок 2.12).

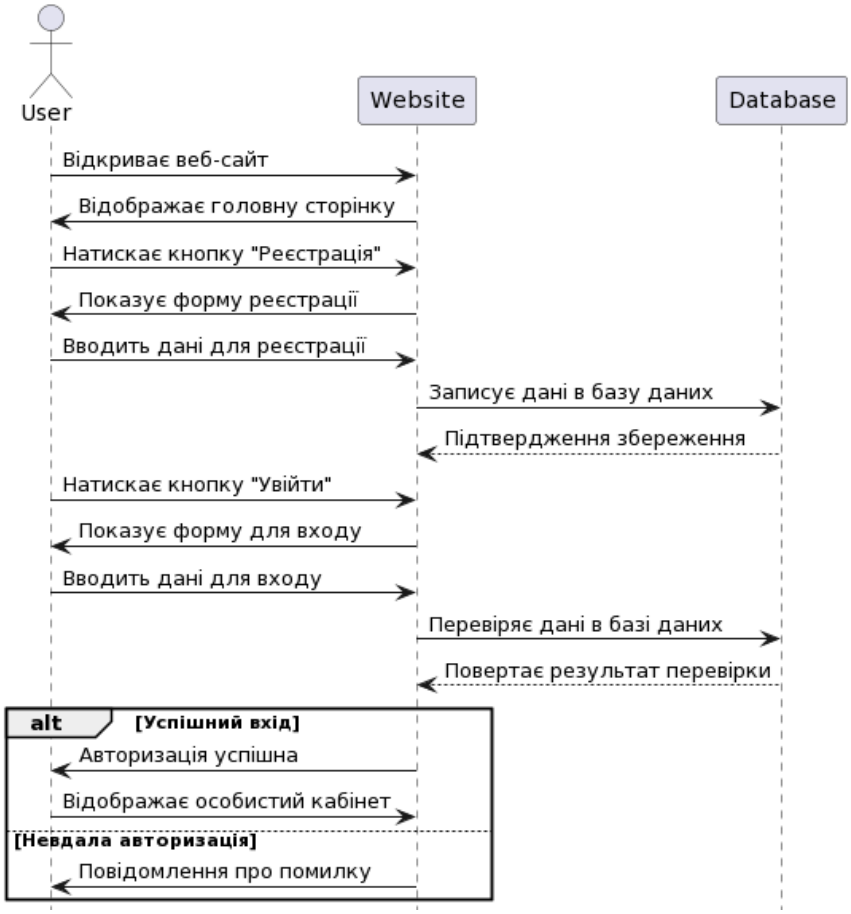


Рисунок 2.12 – Схема реєстрації користувача

2.5 Взаємодія користувача із системою

Коли користувач першість заходить на головну сторінку сайту, він знаходиться перед атмосферою рекомендацій фільмів та серіалів, яка відразу привертає його увагу. Це може бути важливим етапом, оскільки перше враження часто визначає подальшу взаємодію користувача з платформою. Важливою частиною цієї сторінки є кнопка "Увійти", яка відкриває доступ до різноманітних функцій та контенту сайту. (рисунок 2.13) Користувач може натиснути на цю

кнопку з метою отримати більше можливостей, таких як збереження улюблених фільмів або отримання персоналізованих рекомендацій.

При виборі кнопки "Увійти" відображається форма для реєстрації або входу на сайт. Це може бути першим кроком у взаємодії користувача з платформою, де він має можливість створити акаунт або увійти за допомогою вже наявного. Варіанти входу через GitHub або Google акаунт додають зручності та прискорюють процес авторизації. Це створює сприятливі умови для користувача, що може бути важливим фактором при залученні нових користувачів та збереженні існуючих.

Після успішного входу або реєстрації користувач має можливість досліджувати більше контенту на сторінці "Фільми". Каталог з фільмами і серіалами може бути джерелом нескінченного розваги для користувача, пропонуючи різні жанри та теми.

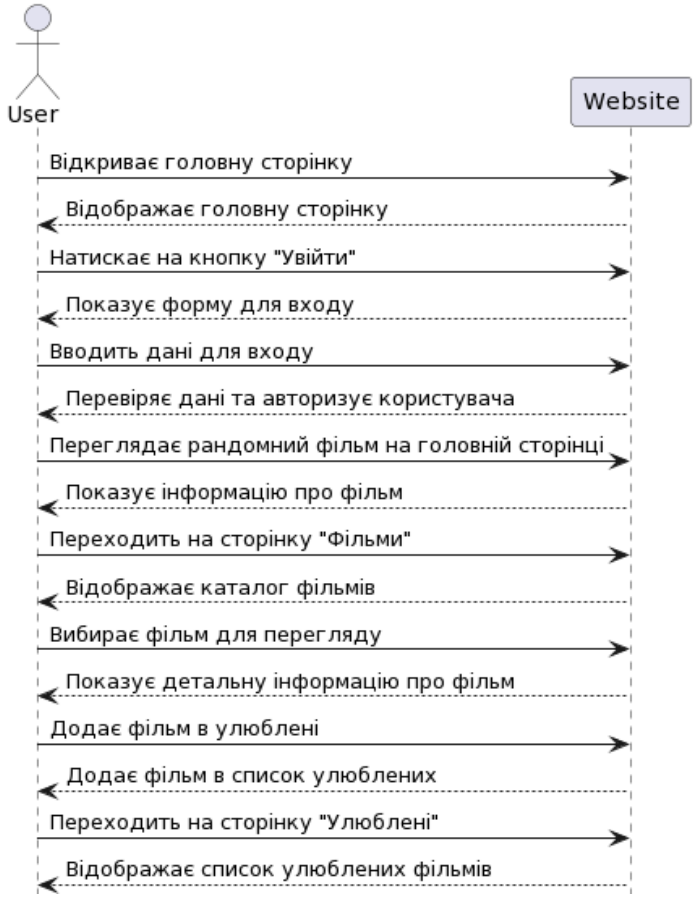


Рисунок 2.13 – Схема взаємодії користувача з сайтом

База даних MongoDB та її сутності:

1. Сутність "Account":

_id: Унікальний ідентифікатор облікового запису.

userId: Унікальний ідентифікатор користувача, пов'язаний з цим обліковим записом.

type: Тип облікового запису (наприклад, "local" для локальної аутентифікації або "external" для стороннього постачальника).

provider: Постачальник аутентифікації (наприклад, "github", "google" тощо).

providerAccountId: Ідентифікатор облікового запису в постачальника (наприклад, ідентифікатор користувача в GitHub).

access_token: Токен доступу, використовується для авторизації користувача.

expires_at: Дата та час закінчення терміну дії токена.

token_type: Тип токена (наприклад, "Bearer").

scope: Перелік дозволених операцій для токена.

id_token: Токен, що містить інформацію про користувача (наприклад, JWT токен).

2. Сутність "Movies":

_id: Унікальний ідентифікатор фільму.

title: Назва фільму.

description: Опис фільму.

videoUrl: Посилання на відеофайл фільму.

thumbnailUrl: Посилання на зображення (постер) фільму.

genre: Жанр фільму.

duration: Тривалість фільму.

3. Сутність "User":

_id: Унікальний ідентифікатор користувача.

name: Ім'я користувача.

email: Електронна адреса користувача.

image: Посилання на зображення профілю користувача.

emailVerified: Прапорець, що позначає, чи підтверджена електронна адреса користувача.

createdAt: Дата створення облікового запису користувача.

updatedAt: Дата та час останнього оновлення інформації про користувача.

favoriteIds: Масив ідентифікаторів улюблених фільмів.

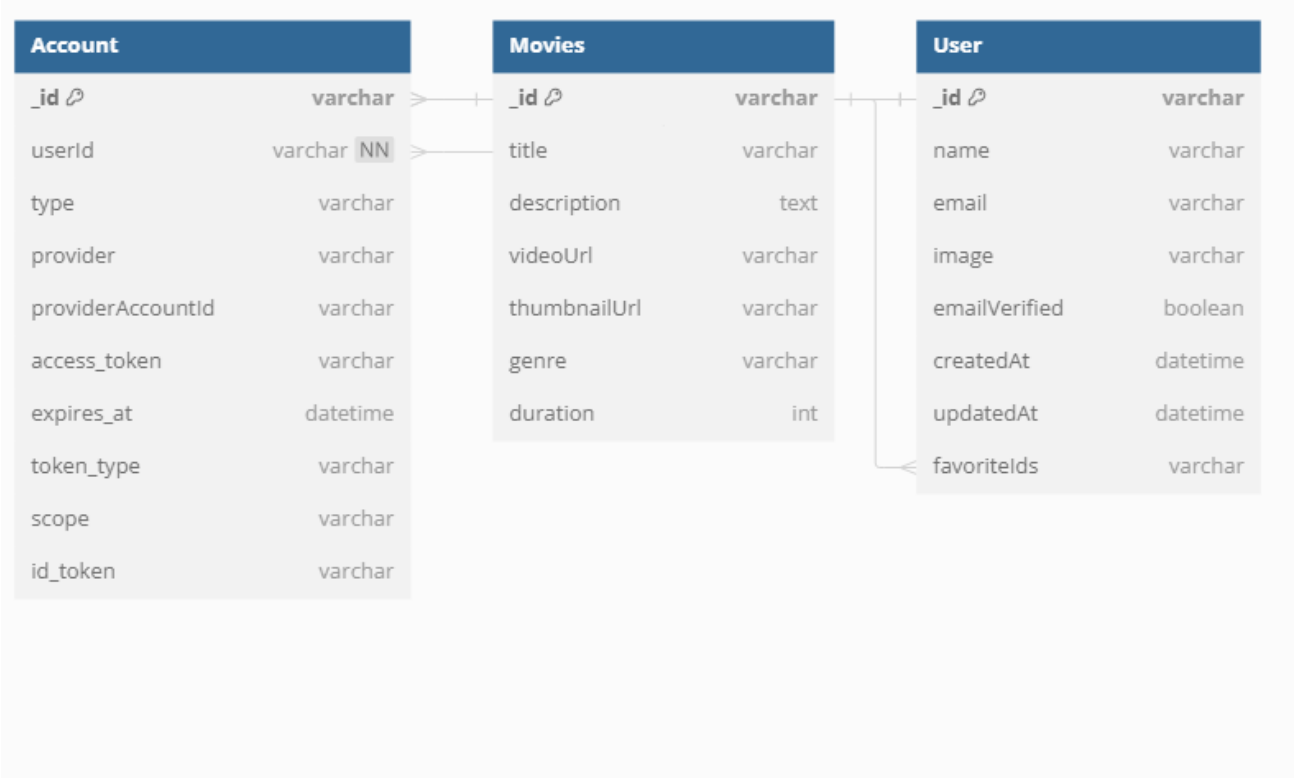


Рисунок 2.14 - Взаємодія MongoDB

Висновки

Мета цієї інформаційної системи - надати споживачам доступ до розважального контенту через Інтернет, включаючи телесеріали та фільми. Завдяки цьому користувачі можуть шукати та переглядати відеоконтент у зручний для них час, створюючи персоналізований і комфортний досвід.

Користувачі мають можливість змінювати свій профіль та переглядати інформацію про свої вподобання, створивши особистий обліковий запис в інформаційній системі. Назви фільмів і серіалів, жанри та рейтинги - це лише деякі з параметрів, за якими користувачі можуть шукати та переглядати контент. Для зручності вони можуть додавати певні матеріали до списку обраного, щоб переглянути їх пізніше.

Завдяки зручному інтерфейсу користувачі можуть легко орієнтуватися в системі та швидко знаходити потрібний матеріал. Система гарантує стабільне та ефективне функціонування завдяки використанню сучасних інструментів і технологій, що дозволяє глядачам насолоджуватися переглядом фільмів без зайвих перерв та збоїв.

Ця інформаційна система забезпечує користувачам можливість насолоджуватися улюбленими серіалами та фільмами в будь-який зручний для них час і в будь-якому місці. Це створює приємну атмосферу для перегляду розважальних матеріалів, підвищуючи загальне задоволення від користування платформою. Вона також пропонує високий рівень персоналізації та адаптації до індивідуальних потреб і вподобань користувачів, що робить її конкурентоспроможною на ринку розважального контенту.

Таким чином, система поєднує в собі передові технології, зручність використання та високу якість обслуговування, що дозволяє користувачам насолоджуватися контентом без обмежень, сприяючи максимальному задоволенню їхніх потреб і очікувань.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Компоненти інтерфейсу користувача інформаційної системи

Для інформаційної системи з перегляду розважального контенту використовуються різноманітні компоненти, які спрямовані на забезпечення необхідної функціональності та інтерфейсу. Ці компоненти включають у себе різноманітні елементи, такі як кнопки, форми, навігаційні панелі та багато іншого. Кожен з них має свою унікальну роль та призначений для виконання певних завдань у контексті інформаційної системи. Деякі з них спрямовані на навігацію між сторінками, інші - на відображення вмісту чи взаємодію з користувачем. Разом вони створюють зручне та ефективне середовище для користувачів і забезпечують потрібну функціональність для інформаційної системи з перегляду розважального контенту.

Компонент BgProvider

Забезпечує зручне додавання стилізованого фону для будь-якого вмісту всередині нього. Його основна функція полягає в створенні контейнера з фоновим зображенням, яке займає всю доступну площу екрану і має адаптивну прозорість. Це дозволяє створювати привабливий інтерфейс для різних частин веб-додатку, таких як головна сторінка або сторінка входу.

Компонент приймає `children`, який представляє будь-який вміст, що буде вкладений і відображений на фоні.

Компонент Billboard

Відповідає за відображенні відеобанера на головній сторінці. Він автоматично відтворює відео з інформаційним супроводом, створюючи візуально привабливий та інтерактивний елемент для користувачів. Billboard додає динамічний та інтерактивний контент на головну сторінку нашого додатку, підвищуючи залученість користувачів (Рисунок 3.1).

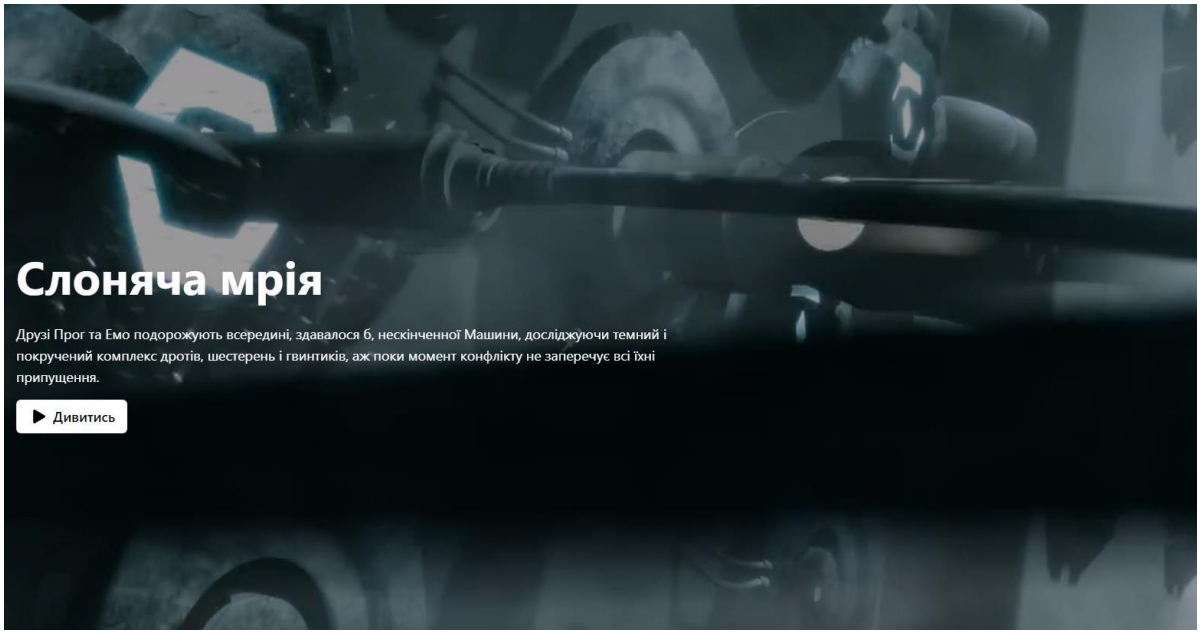


Рисунок 3.1 – Фонове відео Billboard

Компонент FavoriteButton

Його головні функції полягають у відстеженні чи фільм добавлений до улюблених, взаємодії з API для зміни цього стану та оновленні відповідного інтерфейсу

Компонент використовує внутрішній стан, щоб визначити, чи фільм вже є у списку улюблених чи ні.

При кліку на кнопку, він взаємодіє з сервером за допомогою Axios, надсилаючи відповідний запит на додавання або видалення фільму зі списку улюблених.

Після успішної взаємодії стан компонента оновлюється, а кнопка відображає відповідну іконку: "плюс", якщо фільм додається до списку улюблених, або "перевірено" (рисунок 3.2), якщо фільм видаляється.

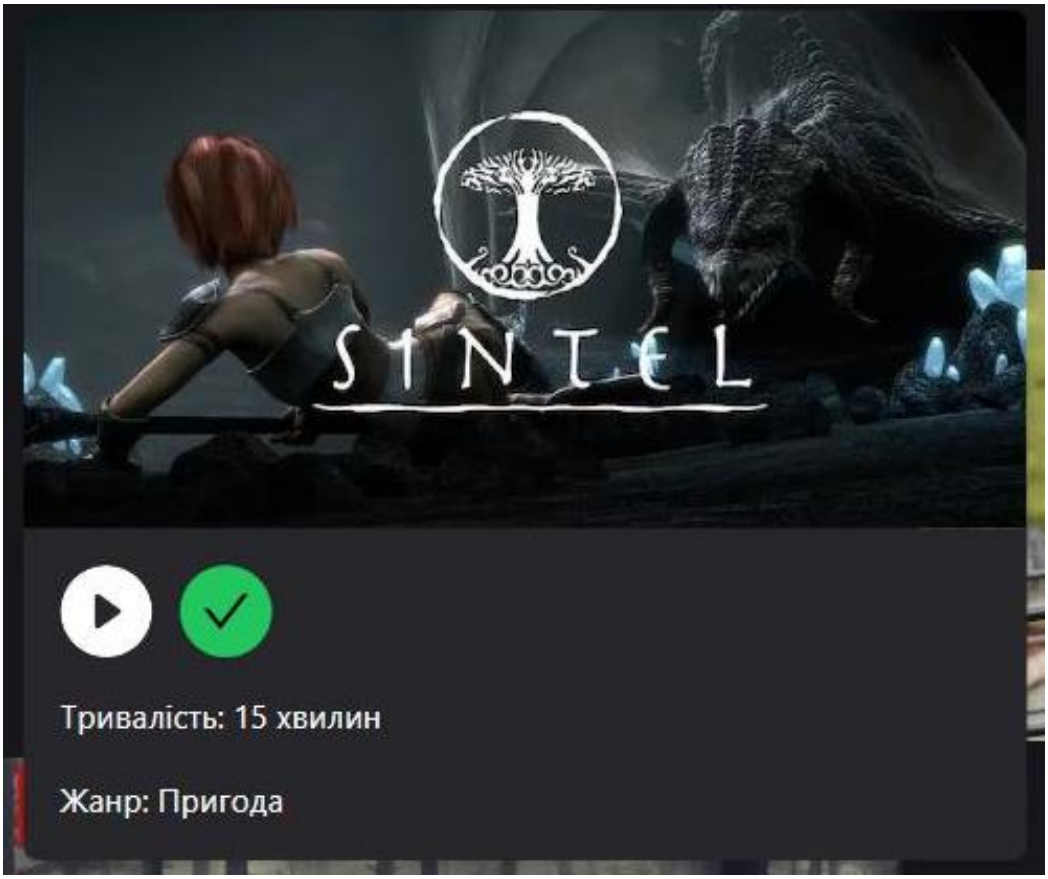


Рисунок 3.2 – Можливість додавати улюблені фільми FavoriteButton

Компонент GoBack

Використовується для повернення користувача на попередню сторінку у додатку, дозволяючи зручно навігуватися назад по історії перегляду. Основна функція цього компоненту полягає у взаємодії з маршрутизатором Next.js для переходу на попередню сторінку.

Компонент Input

Його основна роль полягає в забезпеченні можливості введення даних користувачем з подальшою взаємодією з інформацією. Він приймає параметри, такі як ідентифікатор, функція зміни значення, поточне значення, мітка та тип поля введення. Забезпечує зручний та естетичний спосіб для введення даних користувачем, допомагаючи полегшити їх взаємодію з додатком.

Компонент MovieCard

Відповідає за відображення інформації про окремий фільм. Основна мета цього компонента - формування інтерфейсу, який дозволяє користувачам зручно

переглядати деталі фільму та здійснювати взаємодію з ним. Функції компонента: Відображення зображення фільму, інтерактивність, кнопки дій, відображення інформації про фільм. (рисунок 3.3.)

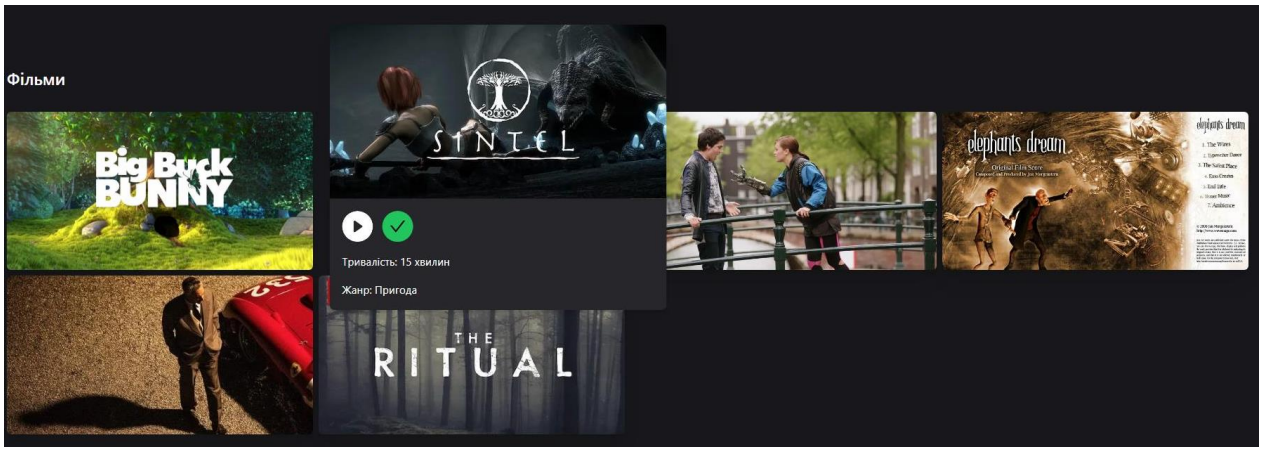


Рисунок 3.3 – Карта з фільмами MovieCard

Компонент MoviesList

Відображає список фільмів у додатку. Він використовує функції щоб отримати дані про поточного користувача, і визначає, чи належить фільм до списку улюблених користувача (рисунок 3.4). Потім компонент відображає заголовок списку та перераховує кожен фільм за допомогою компонента MovieCard.

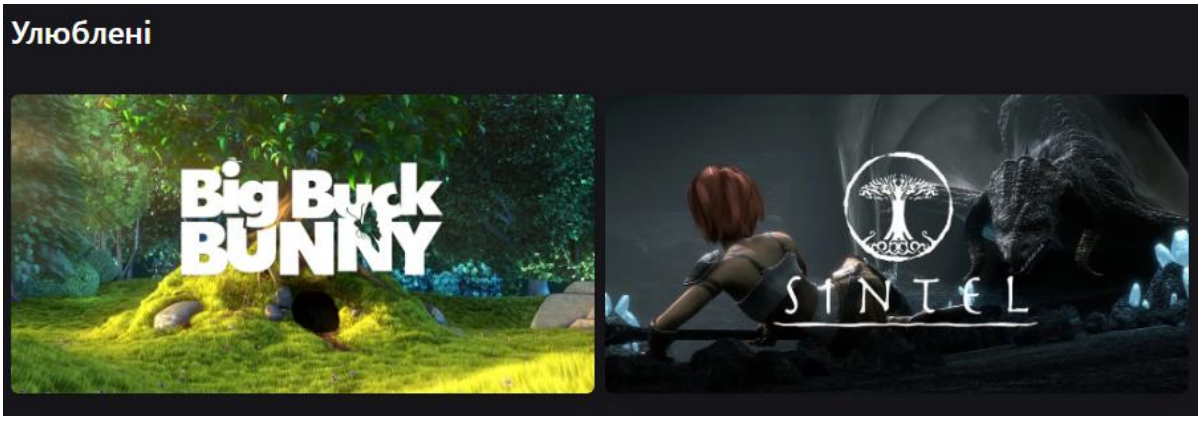


Рисунок 3.4 – Відображення улюблених фільмів

Компонент Navbar

Відображає верхнє навігаційне меню додатку. Він містить логотип додатку, який при кліці перенаправляє користувача на головну сторінку. У верхній частині меню розташовані елементи навігації, такі як "Головна", "Фільми" та "Улюблені", які допомагають користувачам швидко переміщатися між сторінками (рисунок 3.5). Якщо користувач увійшов у систему, у правій частині меню відображається його аватар та ім'я, а також кнопка "Вийти", яка дозволяє вийти з облікового запису. У разі, якщо користувач не увійшов у систему, замість імені відображається кнопка "Увійти", яка перенаправляє користувача на сторінку входу.

Компонент NavbarItem

Призначений для відображення окремого пункту навігаційного меню у верхній частині додатку. У випадку активності, текст відображається білим кольором з підкресленням, в іншому випадку - сірим кольором, змінюючись при наведенні курсора (рисунок 3.5).

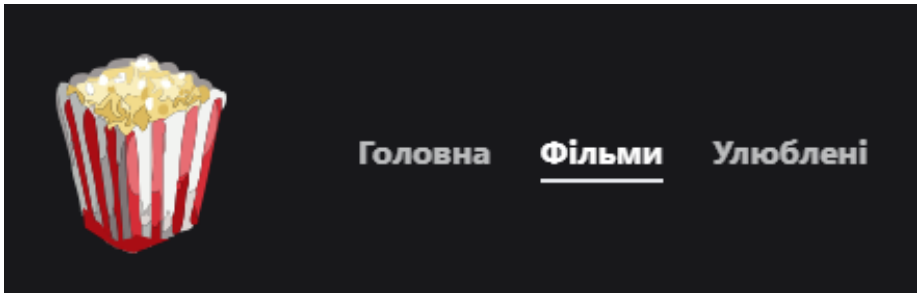


Рисунок 3.5 – Navbar, виділення білим та сірим NavbarItem

Компонент PlayButton

Відповідає за відображення кнопки для відтворення відео. Цей компонент має єдиний параметр movieId який використовується для побудови URL адреси відтворення відео (рисунок 3.6).

Кнопка представлена у вигляді текстового блоку з іконкою відтворення. При кліці на кнопку, використовуючи роутер Next.js, користувач буде перенаправлений на сторінку відтворення відео з відповідним ідентифікатором.

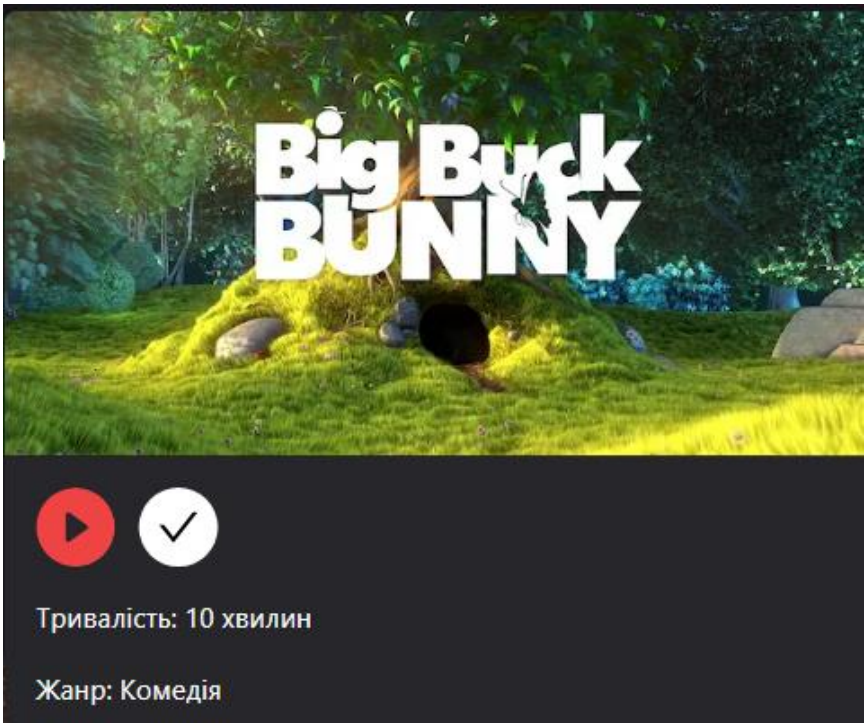


Рисунок 3.6 - PlayButton

Ці компоненти взаємодіють між собою (рисунок 3.7), а також взаємодіють з зовнішніми сервісами, такими як отримання даних про користувача або відео для реклами, за допомогою HTTP-запитів. Разом вони створюють повноцінний веб-додаток зі зручним інтерфейсом та різноманітною функціональністю для користувача.

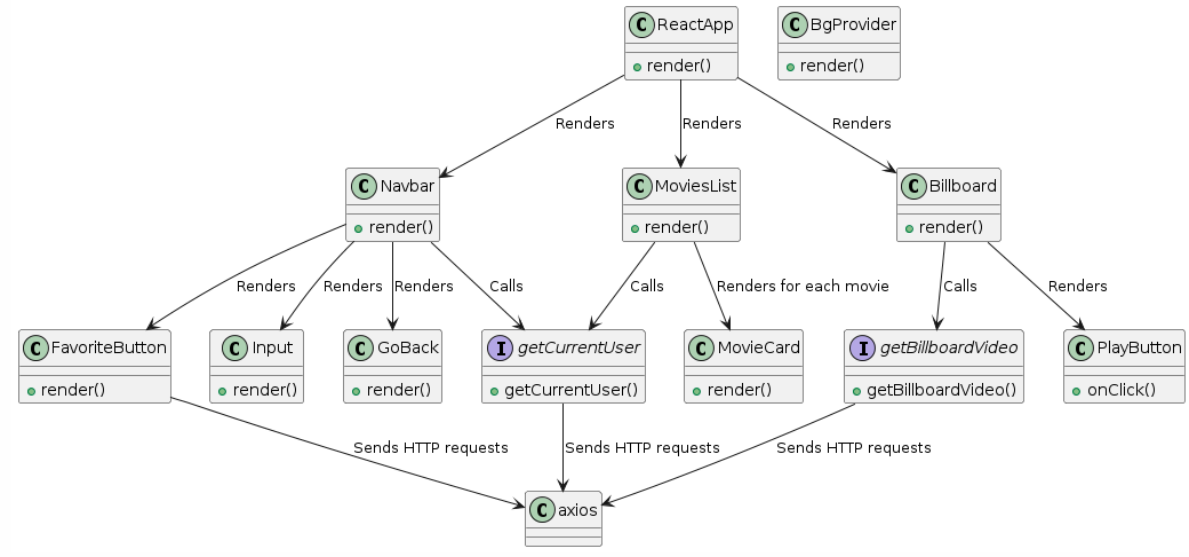


Рисунок 3.7 Взаємодія компонентів користувацького інтерфейсу.

3.2. Особливості функціонування серверної частини системи

API (Application Programming Interface) є ключовою технологією, що забезпечує обмін даними між клієнтською частиною веб-сайту та сервером, на якому розміщено додаток (рисунок 3.8). Це дозволяє здійснювати різноманітні операції, такі як створення користувачів, автентифікація, збереження та управління даними користувачів, включаючи улюблені фільми. API забезпечує комунікацію між компонентами системи, що дає можливість створювати інтерактивні та динамічні веб-додатки.

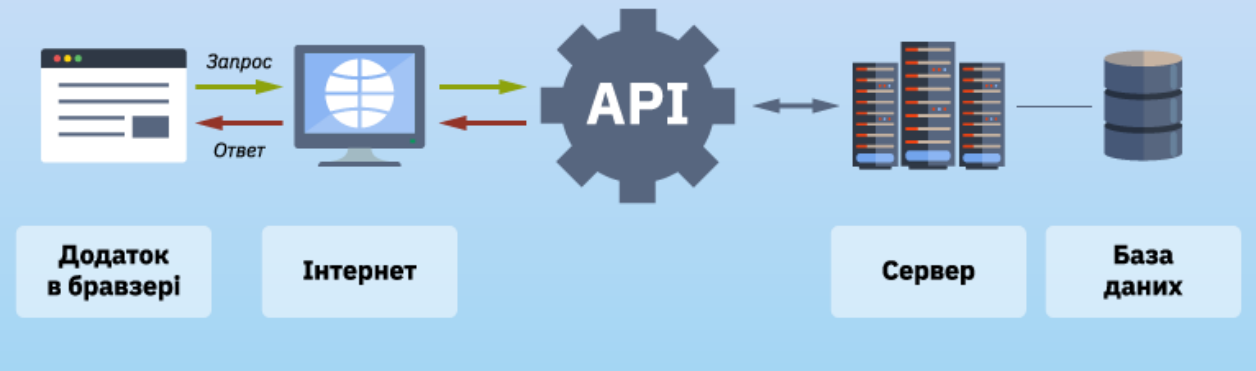


Рисунок 3.8 – Технологія API

Основна функціональність API:

1. Створення користувачів: клієнт може відправити POST-запит з даними користувача, такими як ім'я, електронна адреса та пароль. Сервер отримує ці дані, хешує пароль за допомогою bcrypt, а потім зберігає інформацію про користувача в базі даних MongoDB.

Auth - У інформаційній системі для перегляду розважального контенту з автентифікацією користувачів використовується бібліотека NextAuth для надання різних способів входу в систему. Для цього використовуються провайдери автентифікації, такі як Google, GitHub.

Провайдери Google та GitHub дозволяють користувачам використовувати свої облікові записи на цих платформах для входу в систему. Після успішної аутентифікації дані користувача автоматично надходять в систему.

Крім того, є можливість входу за допомогою електронної пошти та пароля за допомогою провайдера Credentials. Користувач може ввести свою електронну адресу та пароль, і після перевірки цих даних у базі даних йому надається доступ до системи.

Для забезпечення безпеки паролі користувачів зберігаються у хешованому вигляді за допомогою бібліотеки bcrypt, що гарантує захист від перехоплення та викрадення паролів.

2. Вхід користувача: користувач може ввійти до свого облікового запису, надавши свою електронну адресу та пароль. Сервер перевіряє, чи існує користувач з такою електронною адресою, та порівнює хешований пароль збережений у базі даних з тим, який був наданий. Якщо дані вірні, користувач автентифікується і отримує доступ до свого облікового запису.

Reg – тут використовуються функції `POST` відповідає за обробку POST-запитів для створення нового користувача у системі. При отриманні запиту, функція активує запит, де очікується наявність параметрів `email`, `name` та `password`, які представляють собою електронну адресу користувача, його ім'я та пароль.

Перевіряється, чи всі обов'язкові поля заповнені. Якщо немає хоча б одного з обов'язкових параметрів, повертається відповідь з кодом помилки.

Для забезпечення захисту паролю використовується`bcrypt`, щоб його хешувати та забезпечити безпеку. Після чого створюється новий користувач у базі даних, використовуючи дані, отримані з тіла запиту, а також хешований пароль та поточну дату як підтвердження електронної пошти.

Після успішного створення користувача повертається відповідь у форматі JSON з даними нового користувача. У випадку помилки, також повертається відповідь з відповідним кодом помилки.

3. Збереження улюблених фільмів: користувач може додати фільм до свого списку улюблених. При цьому на сервері створюється запис про вподобання користувача, який зв'язується з його обліковим записом.

Favorit – тут використовуються функції `POST` та `DELETE` відповідають за обробку POST- та DELETE-запитів відповідно для додавання та видалення фільмів із списку улюблених користувачів. Після отримання тіла запиту, де передається параметр `movieId`, який ідентифікує фільм, надходиться поточний користувач. Якщо користувач авторизований, перевіряється наявність фільму у базі даних. Якщо фільм знайдено, він додається до списку улюблених фільмів користувача, а потім оновлені дані користувача зберігаються у базі даних.

Функція `DELETE` виконує аналогічні дії, але замість додавання фільму видаляє його із списку улюблених фільмів користувача.

Функції гарантують, що тільки авторизовані користувачі можуть змінювати свій список улюблених фільмів та забезпечують надійне збереження даних у базі даних. У випадку помилки оброблюється виняток, і надсилається відповідь з відповідним кодом помилки.

Робота з базами даних є критично важливою для багатьох систем, оскільки забезпечує зберігання та доступ до даних, необхідних для функціонування різноманітних додатків та сервісів. На сервері баз даних зберігається велика кількість інформації, включаючи конфіденційні дані, такі як паролі користувачів. Збереження таких даних у безпечному вигляді є одним з ключових завдань розробників та системних адміністраторів.

Одним із ефективних способів забезпечення безпеки паролів є використання алгоритмів хешування. Хешування паролів передбачає перетворення вихідного паролю у хеш, який зберігається у базі даних замість самого паролю. Цей підхід гарантує, що навіть у разі компрометації бази даних, отримання справжніх паролів користувачів буде вкрай складним завданням.

Для захисту паролів користувачів у базі даних MongoDB використовує алгоритм хешування bcrypt. Цей алгоритм відомий своєю високою безпекою та стійкістю до різних видів атак.

Хешування паролів - це процес перетворення паролю у вигляд, який неможливо відновити у вихідний пароль. При хешуванні пароля генерується

унікальний хеш, який потім зберігається у базі даних замість самого пароля. Однак, важливо враховувати, що хеш може бути зламаний за допомогою перебору, тому важливо використовувати надійні алгоритми хешування та додаткові заходи безпеки.

Bcrypt є одним з найбільш надійних алгоритмів хешування паролів, особливо через його використання солі. Сіль - це випадкове значення, яке додається до вихідного паролю перед хешуванням. Використання солі ускладнює атаки з використанням попередніх атак хешування та робить процес зламу паролів більш часо- та ресурсозатратним.

Крім того, bcrypt дозволяє налаштовувати "робочий фактор" (work factor), який визначає кількість обчислень, які виконуються для створення хешу. Чим вище робочий фактор, тим більше ресурсів потрібно для обчислення хешу, що робить атаки brute force ще більш непрактичними.

Prisma - це інструмент, який дозволяє нам легко і зручно взаємодіяти з базою даних, не занурюючись у деталі роботи з MongoDB. Один із ключових аспектів Prisma - це можливість генерації класів на основі моделей даних, які ми описуємо у файлі `schema.prisma`. Ці класи представляють собою спрощений спосіб роботи з даними.

У проєкті визначені такі класи, як `User`, `Movie` та `Favorite`. Кожен з цих класів може мати різноманітні методи, які допомагають нам взаємодіяти з базою даних.

Наприклад, метод `findUnique()` дозволяє нам знаходити конкретний запис у базі даних за унікальним ідентифікатором, тоді як метод `create()` дозволяє створювати нові записи.

Ці методи надають зручний інтерфейс для роботи з даним. Вони допомагають нам зберігати, отримувати, оновлювати та видаляти дані у базі даних MongoDB без необхідності писати складні SQL-запити.

3.3 Перевірка роботи реалізованої системи

Для забезпечення високої якості та ефективності роботи веб-системи необхідно провести комплексну перевірку її функціональності. Це включає тестування основних можливостей, які повинні бути доступні користувачам під час взаємодії з сайтом.

Розглянуто етапи перевірки трьох ключових функцій системи: реєстрації користувача через Google акаунт, додавання фільмів до списку улюблених та відображення відео на головній сторінці сайту з можливістю їх перегляду у вбудованому плеєрі.

1. Реєстрації користувача через Google акаунт

Користувач бажає зареєструватися на веб-сайті за допомогою свого Google акаунту. Сайт повинен підтримувати функціонал OAuth для авторизації через Google (рисунок 3.9).

Користувач вже має активний Google акаунт і бажає використовувати його для реєстрації на сайті. Веб-сайт повинен бути інтегрований з Google API для можливості авторизації.

Таблиця 3.1 – Тест кейс реєстрації через Google

Дія	Очікуваний результат
Користувач натискає на кнопку "Реєстрація через Google"	Відкривається вікно авторизації Google
Користувач виконує авторизацію в Google.	Google успішно авторизує користувача і запитує дозвіл на доступ до даних.
Користувач погоджується на доступ до даних та підтверджує реєстрацію.	Користувач успішно зареєстрований на сайті та автоматично увійшов в систему.

Рисунок 3.9 – Відображення акаунту після успішного входу в базі

2. Коректне додавання фільму в улюблене

Користувач зареєстрований і увійшов в систему на веб-сайті. У користувача є доступ до бібліотеки фільмів, і він бажає додати певний фільм до списку улюблених(рисунок 3.10) .

Веб-сайт повинен підтримувати функціонал додавання фільмів до списку улюблених і зберігати цей список для кожного користувача.

Таблиця 3.2 - Тест кейс додавання фільму в улюблене

Дія	Очікуваний результат
Користувач вибирає фільм.	Відображаються деталі фільму з опцією додавання в улюблене.
Користувач натискає на кнопку "Додати в улюблене".	Фільм додається до списку улюблених користувача.
Користувач відкриває свій список улюблених.	Доданий фільм відображається у списку улюблених.

Рисунок 3.10 – Відображення улюблених в базі після додавання

3. Коректне відображення відео на головній сторінці сайту та можливість його перегляду у плеєрі

Веб-сайт має головну сторінку, на якій відображаються відео для перегляду. Користувач відвідує головну сторінку веб-сайту і хоче переглянути відео безпосередньо з цієї сторінки.

Веб-сайт повинен підтримувати відображення відео у вбудованому плеєрі з можливістю відтворення відео (рисунок 3.11).

Таблиця 3.3 - Тест кейс додавання фільму в улюблене

Дія	Очікуваний результат
Користувач відкриває головну сторінку.	На головній сторінці відображаються відео.
Користувач вибирає відео та натискає на кнопку відтворення.	Відео відкривається у вбудованому плеєрі.
Користувач натискає на кнопку відтворення у плеєрі.	Відео починає відтворюватися без помилок і з правильною візуалізацією.

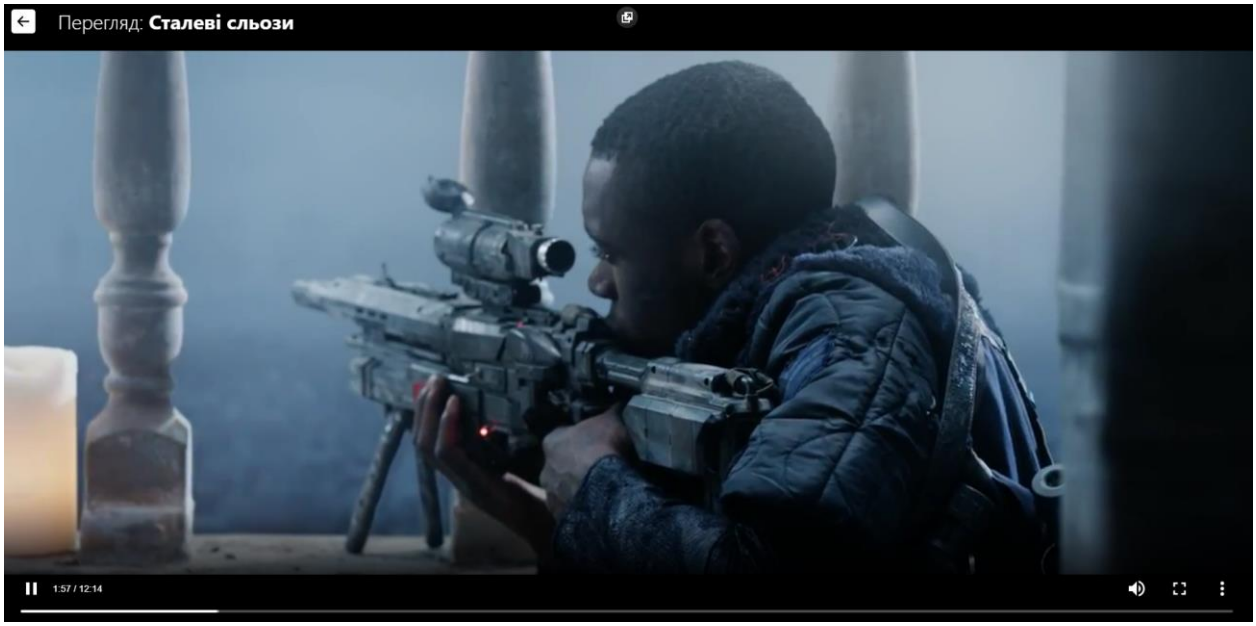


Рисунок 3.11 – Відтворення відео.

3.4 Демонстрація функціоналу компонентів інтерфейсу

Для забезпечення зручної та ефективної взаємодії користувачів з платформою розважального контенту було розроблено комплексний інтерфейс, що включає в себе кілька ключових елементів. Основна мета цього інтерфейсу - забезпечити інтуїтивно зрозумілий та приємний досвід для користувачів, надаючи їм легкий доступ до всіх необхідних функцій та розділів платформи.

Розробка інтерфейсу враховувала різноманітні сценарії використання, щоб задовольнити потреби як нових користувачів, так і досвідчених. Було проведено детальний аналіз поведінки користувачів та їх потреб, що дозволило створити структуру, яка є логічною та зрозумілою. Інтерфейс включає такі елементи, як головна сторінка, вікно реєстрації, домашня сторінка, сторінка для перегляду, сторінка з фільмами та сторінка з улюбленими.

Демонстрація функціоналу компонентів інтерфейсу детально описує процес взаємодії користувача з платформою, починаючи від моменту входу до системи і закінчуючи переглядом улюбленого контенту. Кожна сторінка виконує свою

специфічну функцію і пропонує певні можливості для користувача, забезпечуючи цілісний та зручний досвід..

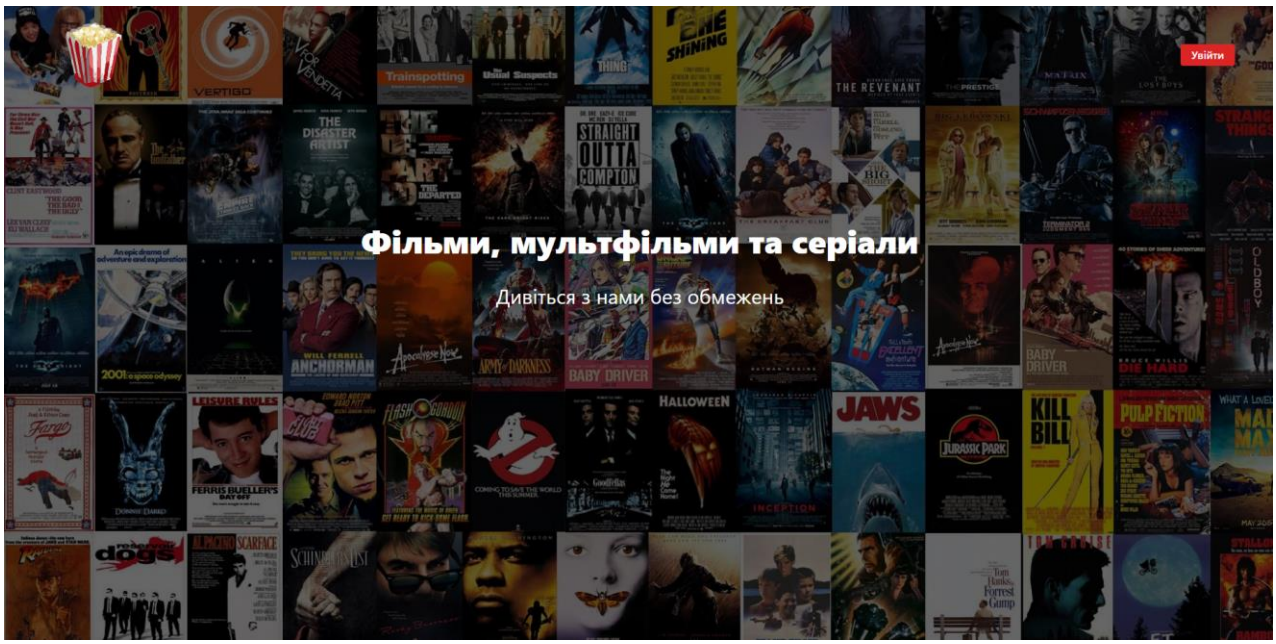


Рисунок 3.12 – Головна сторінка

Головна сторінка слугує точкою входу до сервісу. Вона дозволяє користувачам отримати доступ до своїх облікових записів та розпочати роботу з сервісом, надаючи інтуїтивно зрозумілий інтерфейс (рисунок 3.12).

На головній сторінці представлено привабливий фоновий колаж з популярними фільмами, що одразу привертає увагу користувачів та створює атмосферу різноманіття і багатства вибору контенту. У верхньому лівому куті розташований логотип сервісу, який слугує не лише елементом брендування, а й кнопкою повернення на головну сторінку з будь-якого розділу сайту.

Центральне місце займає великий напис "Фільми, мультфільми та серіали", який інформує користувачів про основний контент платформи. Під цим написом знаходиться слоган "Дивіться з нами без обмежень", що підкреслює доступність та широкий вибір контенту для перегляду без жодних лімітів.

У правому верхньому куті розміщена кнопка "Увійти", яка дозволяє користувачам швидко перейти до сторінки входу чи реєстрації. Ця кнопка відразу виділяється завдяки своєму яскраво-червоному кольору, що полегшує її знаходження навіть для нових користувачів.

Головна сторінка забезпечує користувачам перше враження про сервіс, її дизайн та функціональність спрямовані на те, щоб максимально спростити доступ до контенту та залучити користувачів до подальшої взаємодії з платформою. Користувачі можуть легко розпочати навігацію по сайту, отримуючи доступ до багатого асортименту фільмів, мультфільмів та серіалів, які вони можуть переглядати у будь-який час та без обмежень.

Користувач може:

- ввести натиснути на кнопку «Увійти» до свого облікового запису;
- натиснути на логотип, щоб повернутися до початкової сторінки.

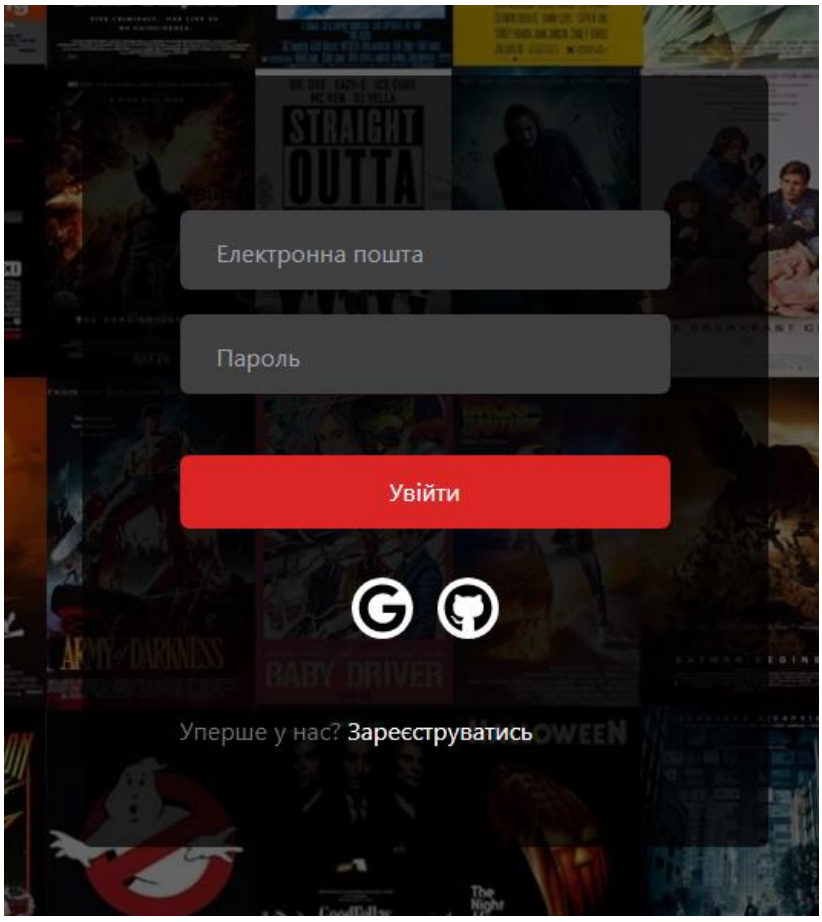
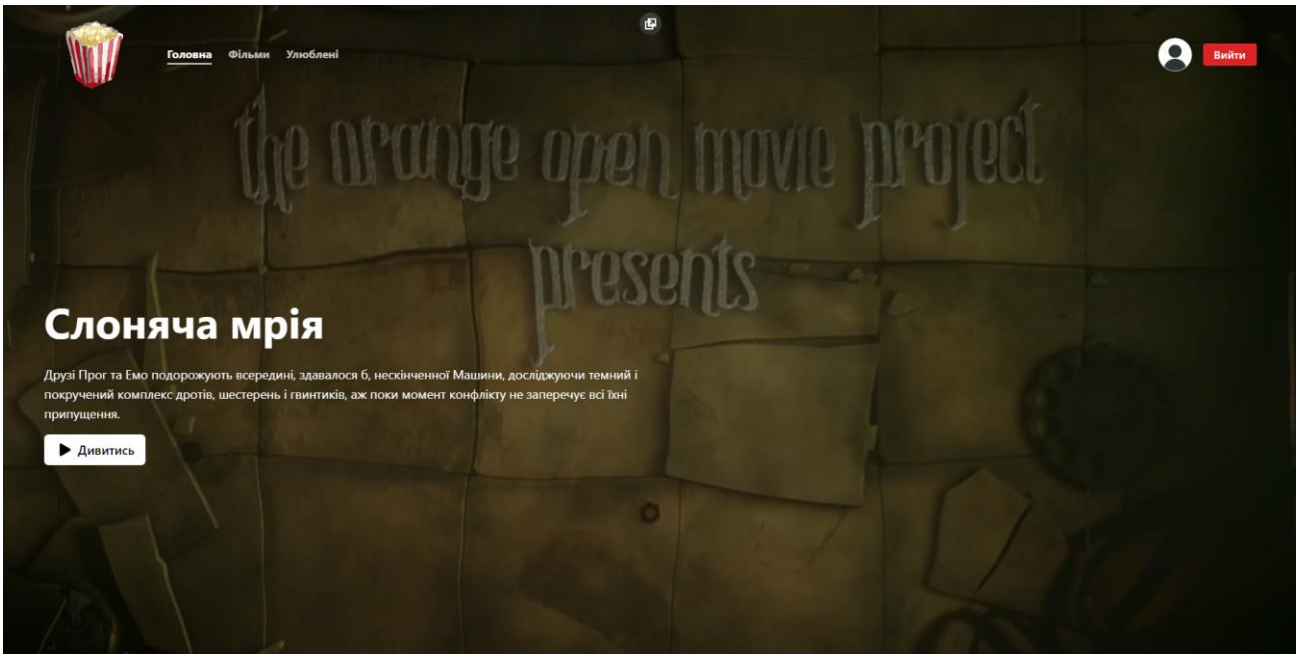


Рисунок 3.13 – Вікно реєстрації

У вікні реєстрації користувач бачить форму для авторизації на платформі. Тут користувач може: Ввести свою електронну пошту і пароль для входу в систему, використовувати альтернативні методи входу за допомогою Google або GitHub, зареєструвати новий обліковий запис, якщо у користувача його ще немає, натиснувши на посилання "Зареєструватись"(рисунок 3.13).

Тут користувач може:

- ввести свою електронну пошту і пароль для входу в систему;
- використовувати альтернативні методи входу за допомогою Google або GitHub;
- зареєструвати новий обліковий запис, якщо у користувача його ще немає, натиснувши на посилання "Зареєструватись".



Рискунок 3.14 – Домашня сторінка

На домашній сторінці користувач бачить промо-матеріал певного фільму чи серіалу. У цьому випадку, на екрані відображено анімаційний фільм "Слоняча мрія" з невеликим описом цього матеріалу. Тут користувач може: Переглянути доступні фільми та серіали, перегортаючи стрічку, вибрати фільм для перегляду,

натиснувши кнопку "Дивитись", перейти до інших розділів платформи, таких як "Фільми", "Улюблені", використовуючи навігаційне меню вгорі (рисунок 3.14).

Тут користувач може:

- переглянути доступні фільми та серіали, перегортаючи стрічку;
- вибрати фільм для перегляду, натиснувши кнопку "Дивитись";
- перейти до інших розділів платформи, таких як "Фільми", "Улюблені" тощо, використовуючи навігаційне меню вгорі.

Рисунок 3.15 – Сторінка для перегляду

На сторінці перегляду фільму користувач бачить кадр з фільму "Слоняча мрія" в процесі відтворення (рисунок 3.15).

Тут користувач може:

- переглядати фільм в повноекранному режимі;
- керувати відтворенням відео ставити на паузу, відтворювання та перемотування;
- вийти на попередню сторінку або в меню платформу натиснувши відповідні кнопки.

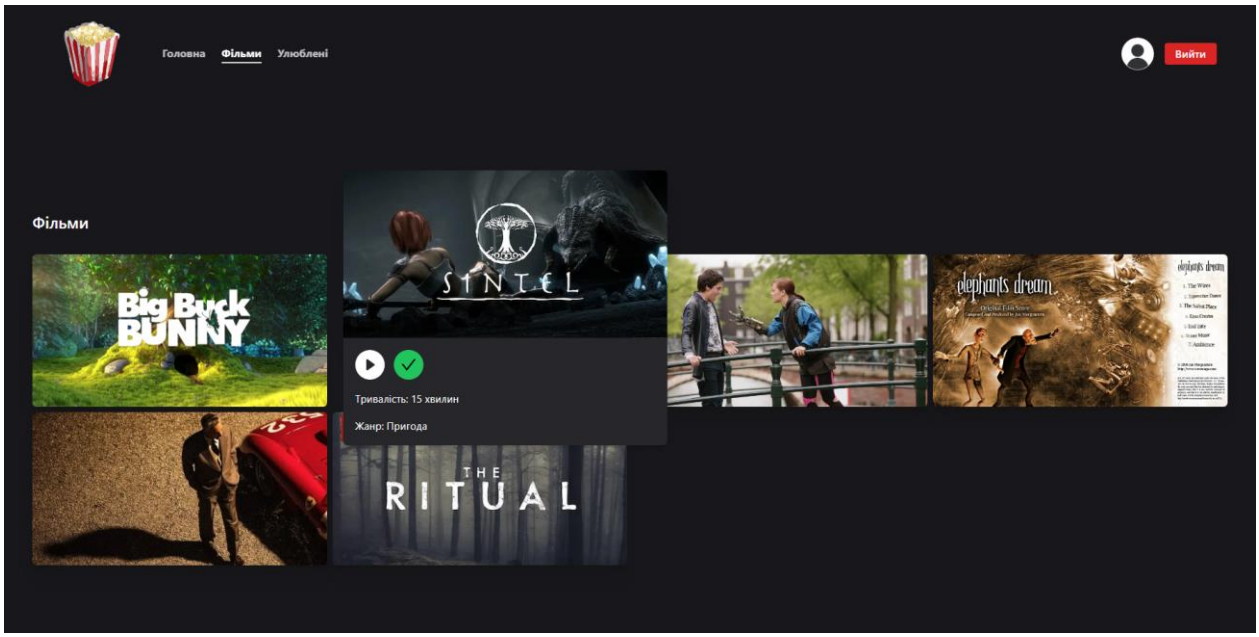


Рисунок 3.16 – Сторінка з фільмами

Сторінка з фільмами слугує основним навігаційним центром. Вона дозволяє користувачам знаходити та переглядати фільми та серіали, а також досліджувати різні розділи сервісу (рисунок 3.16).

Користувач може:

- взаємодіяти з логотипом, щоб повернутися на головну сторінку;
- використовувати панель навігації, щоб перейти до різних розділів сервісу, таких як "Фільми", "Серіали", "Улюблені" та "Вийти";
- переглядати список рекомендованих фільмів та серіалів, які підібрані на основі переглядів та уподобань користувача;
- переходити до інших розділів сервісу;
- натиснути на плитку з фільмом або серіалом, щоб розпочати його перегляд.

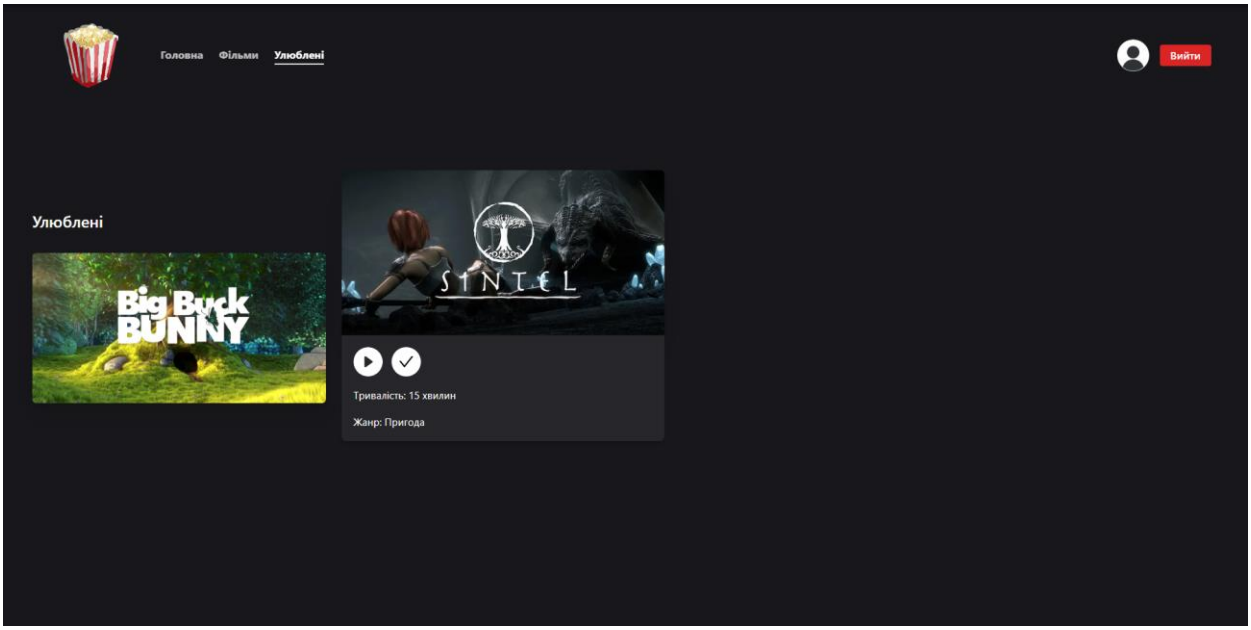


Рисунок 3.17 – Сторінка з улюбленими

Ця сторінка дозволяє користувачам легко знаходити та переглядати свої улюблені фільми та серіали, на цій сторінці відображаються всі фільми та серіали, які користувач додав до своїх "Улюблених". Кожен фільм або серіал представлений плиткою з постером, назвою та іншими даними (рисунок 3.17).

Користувач може:

- переглядати список своїх улюблених фільмів та серіалів;
- натиснути на плитку з фільмом або серіалом, щоб розпочати його перегляд;
- навести курсор на плитку з фільмом або серіалом, щоб побачити додаткову інформацію, таку як опис, рейтинг та рік випуску;
- видалити фільм або серіал зі своїх "Улюблених", натиснувши на кнопку видалення.

Висновки

У цій частині було детально розглянуто компоненти інформаційної системи, їх взаємодію, а також процеси, пов'язані зі створенням та аутентифікацією користувачів, збереженням улюблених фільмів та роботою з базою даних.

Взаємодія компонентів відбувається через API, який забезпечує програмний інтерфейс для взаємодії між клієнтською частиною та серверною частиною. Клієнтська частина відправляє запити на серверну частину, яка обробляє ці запити, взаємодіє з базою даних і повертає результати клієнтській частині. API реалізує функції створення користувачів, входу в систему, додавання фільмів у список улюблених та інші.

Процес створення користувача включає верифікацію даних, збереження інформації в базі даних та шифрування паролів. Вхід користувача передбачає перевірку введених даних з інформацією, що зберігається в базі даних, і авторизацію. Користувач може додавати фільми в свій список улюблених через відповідний API-запит. Інформація про улюблені фільми зберігається в базі даних і доступна для перегляду в особистому кабінеті користувача.

Робота інформаційної системи з базою даних є важливим аспектом. MongoDB зберігає всі дані про користувачів та фільми. Паролі користувачів захищені за допомогою хешування з використанням безпечних алгоритмів, таких як bcrypt, що забезпечує високу ступінь захисту. Prisma виступає як ORM (Object-Relational Mapping) інструмент, що спрощує взаємодію з MongoDB, відправляючи дані та запити до бази даних.

Демонстрація функціоналу компонентів інтерфейсу проекту охоплює критичні функціональні можливості системи, включаючи реєстрацію через Google акаунт, додавання фільмів в улюблене та відображення і перегляд відео на головній сторінці. Тестування дозволяє забезпечити якість і стабільність роботи інформаційної системи, виявляючи можливі помилки та недоліки.

ВИСНОВОК

У процесі роботи було вивчено та проаналізовано існуючі аналоги сайтів розважального контенту, такі як YouTube, Amazon Prime Video, HBO Max та Disney+. Цей аналіз допоміг визначити ключові аспекти їхньої функціональності, інтерфейсу та взаємодії з користувачем. Зокрема, були досліджені базові принципи логіки та булевої алгебри, що дозволило оптимізувати логічні та булеві функції для покращення користувацького досвіду у розробці інформаційної системи.

З метою створення конкурентоспроможної платформи було зосереджено увагу на кількох основних напрямках: зручність користування, швидкість доступу до контенту, безпека даних користувачів, а також масштабованість і надійність системи. Вивчення досвіду провідних платформ дозволило визначити найбільш ефективні практики та впровадити їх у новій розробці.

Основною метою розробленої інформаційної системи є надання користувачам зручного доступу до розважального контенту через Інтернет, включаючи телесеріали та фільми. Користувачі можуть створювати особисті облікові записи, змінювати профілі, переглядати інформацію про свої вподобання, а також додавати матеріали до списку обраного. Система дозволяє шукати контент за різними параметрами, такими як назва, жанр та рейтинг. Завдяки сучасним інструментам і технологіям, забезпечується стабільне та ефективне функціонування системи, що дозволяє користувачам насолоджуватися переглядом без перерв.

Фронтенд системи відповідає за взаємодію з користувачем і відображення даних. Для його розробки було використано сучасні фреймворки та бібліотеки, що забезпечують швидку роботу та зручний інтерфейс. Основними завданнями фронтенду є відображення списків фільмів, надання можливості пошуку та фільтрації контенту, управління профілем користувача та іншими інтерактивними елементами.

Бекенд обробляє дані та взаємодіє з базою даних. Він забезпечує виконання всіх операцій, пов'язаних з обробкою запитів від користувача, верифікацією даних, управлінням обліковими записами та забезпеченням безпеки.

База даних MongoDB зберігає інформацію про користувачів, фільми та їх взаємодії. Вона була обрана за її здатність до масштабування, гнучкість у роботі з даними та високу продуктивність. Всі дані про користувачів, включаючи паролі, зберігаються у зашифрованому вигляді, що забезпечує додатковий рівень безпеки.

Взаємодія компонентів системи відбувається через API, який забезпечує програмний інтерфейс між фронтендом і бекендом. API реалізує всі необхідні функції, включаючи створення користувачів, вхід в систему, додавання фільмів до списку улюблених та інші. Завдяки чіткій структурі API, забезпечується легка інтеграція нових функцій та модулів, що робить систему гнучкою та адаптивною до змін.

Процес створення користувача включає кілька етапів: верифікацію даних, збереження інформації в базі даних та шифрування паролів. Паролі користувачів захищені за допомогою алгоритму хешування bcrypt, який забезпечує високий рівень безпеки. Використання bcrypt ускладнює можливі атаки на паролі, забезпечуючи надійний захист даних користувачів.

Prisma виступає як ORM інструмент, що спрощує взаємодію з MongoDB. Вона дозволяє зручно працювати з даними, створювати запити до бази даних та забезпечує високу продуктивність. Використання Prisma значно полегшує процес розробки та підтримки системи, знижуючи кількість помилок та підвищуючи ефективність роботи з базою даних.

Тестування інформаційної системи охоплює критичні функціональні можливості, включаючи реєстрацію через Google акаунт, додавання фільмів до улюбленого та відображення відео на головній сторінці. Це дозволяє забезпечити якість і стабільність роботи системи, виявляючи можливі помилки та недоліки.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. How Amazon Prime Video Delivers 99.999% Availability While Reducing Costs. URL: https://www.infoq.com/news/2023/10/prime-video-availability-costs/. (дата звернення: 01.02.2024).

2. How Amazon Uses Machine Learning? URL: https://www.geeksforgeeks.org/how-amazon-uses-machine-learning/?ref=gcse (дата звернення: 02.02.2024).

3. Introducing the Disney+ Application Development Kit (ADK). URL: https://medium.com/disney-streaming/introducing-the-disney-application-development-kit-adk-ad85ca139073. (дата звернення: 10.02.2024).

4. Understanding the legacy of HBO Max & its powerful, scalable technical stack. URL: https://www.techaheadcorp.com/blog/understanding-the-legacy-of-hbo-max-its-powerful-scalable-technical-stack/. (дата звернення: 15.02.2024).

5. Nixon R. Learning PHP, MySQL & JavaScript. A Step-by-Step Guide to Creating Dynamic Websites., 2023. 826 p.

6. Wieruch R. The Road to Learn React: Your journey to master plain yet pragmatic React.js. 2023. 256 p.

7. Haverbeke M. Eloquent JavaScript: A Modern Introduction to Programming. 3rd ed. No Starch Press, 2018. 472 p.

8. Paige A., & Geers, B. Frontend Web Development with React: Essentials of Redux and Next.js. Apress, 2020. 300 p.

9. Freeman E., & Robson, E. Head First JavaScript Programming: A Brain-Friendly Guide. O'Reilly Media, 2014. 704 p.

10. Smith N. Learning TypeScript 2.x: Develop and maintain captivating web applications with ease. Packt Publishing, 2018. 512 p.

11. Tompkins A. Mastering Tailwind CSS: A Beginner's Guide. Independently Published, 2021. 120 p.

12. MacKenzie D. JavaScript Everywhere: Building Cross-Platform Applications with GraphQL, React, React Native, and Electron. O'Reilly Media, 2019. 344 p.

13. Banks A. Learning React: Modern Patterns for Developing React Apps. 2nd ed. O'Reilly Media, 2020. 350 p.

14. Rieder J. Full-Stack React, TypeScript, and Node: Build cloud-ready web applications using React 17 with Hooks and GraphQL. Packt Publishing, 2020. 520 p.

15. MongoDB Inc. MongoDB: The Definitive Guide: Powerful and Scalable Data Storage. 3rd ed. O'Reilly Media, 2019. 624 p.

16. Grinberg M. Flask Web Development: Developing Web Applications with Python. 2nd ed. O'Reilly Media, 2018. 256 p.

17. Ford N. Functional JavaScript: Introducing Functional Programming with Underscore.js. O'Reilly Media, 2014. 270 p.

18. Bostock M. Interactive Data Visualization for the Web: An Introduction to Designing with D3. 2nd ed. O'Reilly Media, 2016. 475 p.

19. Grover S. Mastering MEAN Stack: Master ExpressJS, AngularJS, and NodeJS. BPB Publications, 2018. 500 p.

20. Yang E. Fullstack D3 and Data Visualization: Build beautiful data visualizations with D3. Newline, 2020. 544 p.

21. Holm M. Next.js Quick Start Guide: Server-side rendering done right. Packt Publishing, 2019. 178 p.

22. Ross L. Full-Stack React Projects: Modern web development using React 16, Node, Express, and MongoDB. Packt Publishing, 2018. 490 p.

23. Duckett J. JavaScript and jQuery: Interactive Front-End Web Development. Wiley, 2014. 640 p.

24. Razavi S. Mastering Next.js: Build Powerful Server-Side Rendering Applications with Next.js. Independently Published, 2021. 200 p.

25. Chan R. Tailwind CSS: From Zero to Production. Independently Published, 2020. 150 p.

26. React Documentation. URL: https://reactjs.org/docs/getting-started.html (дата звернення: 20.04.2024).

27. JavaScript Documentation. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript (дата звернення: 20.05.2024).

28. Next.js Documentation. URL: https://nextjs.org/docs/getting-started (дата звернення: 20.05.2024).

29. TypeScript Handbook. URL: https://www.typescriptlang.org/docs/handbook/intro.html (дата звернення: 21.05.2024).

30. Tailwind CSS Documentation. URL: https://tailwindcss.com/docs (дата звернення: 22.04.2024).

31. Axios Documentation. URL: https://axios-http.com/docs/intro (дата звернення: 22.05.2024).

32. MongoDB Manual. URL: https://docs.mongodb.com/manual/ (дата звернення: 23.04.2024).

33. React Router Documentation. URL: https://reactrouter.com/docs/en/v6 (дата звернення: 23.04.2024).

34. JavaScript Info. URL: https://javascript.info/ (дата звернення: 23.05.2024).

35. Eloquent JavaScript. URL: https://eloquentjavascript.net/ (дата звернення: 23.04.2024).

36. JavaScript Tutorial. URL: https://www.w3schools.com/js/ (дата звернення: 23.05.2024).

37. Understanding TypeScript. URL: https://academind.com/learn/typescript/ (дата звернення: 24.05.2024).

38. TypeScript Deep Dive. URL: https://basarat.gitbook.io/typescript/ (дата звернення: 25.04.2024).

39. The Road to Learn React. URL: https://www.robinwieruch.de/the-road-to-learn-react/ (дата звернення: 25.04.2024).

40. Practical MongoDB Aggregations. URL: https://university.mongodb.com/courses/M121/about (дата звернення: 01.06.2023).

41. Learn Tailwind CSS. URL: https://scrimba.com/learn/tailwind (дата звернення: 25.04.2024).

42. Next.js Crash Course. URL: https://www.traversymedia.com/courses/nextjs (дата звернення: 26.04.2024).

43. Mastering TypeScript. URL: https://docs.microsoft.com/en-us/learn/modules/typescript-get-started/ (дата звернення: 26.04.2024).

44. Full-Stack React with GraphQL and Apollo. URL: https://www.udemy.com/course/full-stack-react-graphql-apollo/ (дата звернення: 26.04.2024).

45. Learn MongoDB Basics. URL: https://www.codecademy.com/learn/learn-mongodb (дата звернення: 26.04.2024).

46. JavaScript Algorithms and Data Structures. URL: https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/ (дата звернення: 28.04.2024).

47. Build a Full-Stack React Application. URL: https://www.pluralsight.com/courses/build-full-stack-react-application (дата звернення: 28.04.2024).

48. Next.js Guide. URL: https://vercel.com/guides/nextjs (дата звернення: 28.04.2023).

49. Pro MERN Stack. URL: https://github.com/vasansr/pro-mern-stack-2 (дата звернення: 29.04.2024).

50. TypeScript for JavaScript Programmers. URL: https://www.coursera.org/learn/typescript (дата звернення: 01.05.2024).

51. Learn React with TypeScript. URL: https://www.youtube.com/watch?v=Z5iWr6Srsj8 (дата звернення: 01.05.2024).

52. Next.js for Beginners. URL: https://egghead.io/courses/next-js-for-beginners (дата звернення: 01.05.2024).

53. Understanding Tailwind CSS. URL: https://dev.to/tailwindcss/understanding-tailwind-css-1mg6 (дата звернення: 10.05.2024).

54. JavaScript Essentials. URL: https://www.coursera.org/learn/javascript (дата звернення: 01.05.2024).

55. React Context API Guide. URL: https://kentcdodds.com/blog/how-to-use-react-context-effectively (дата звернення: 12.05.2024).

56. Mastering Redux. URL: https://app.pluralsight.com/library/courses/redux-fundamentals (дата звернення: 13.05.2024).

57. MongoDB for JavaScript Developers. URL: https://university.mongodb.com/courses/M001/about (дата звернення: 13.05.2024).

58. Introduction to Tailwind CSS. URL: https://www.youtube.com/watch?v=UBOj6rqRUME (дата звернення: 14.05.2024)

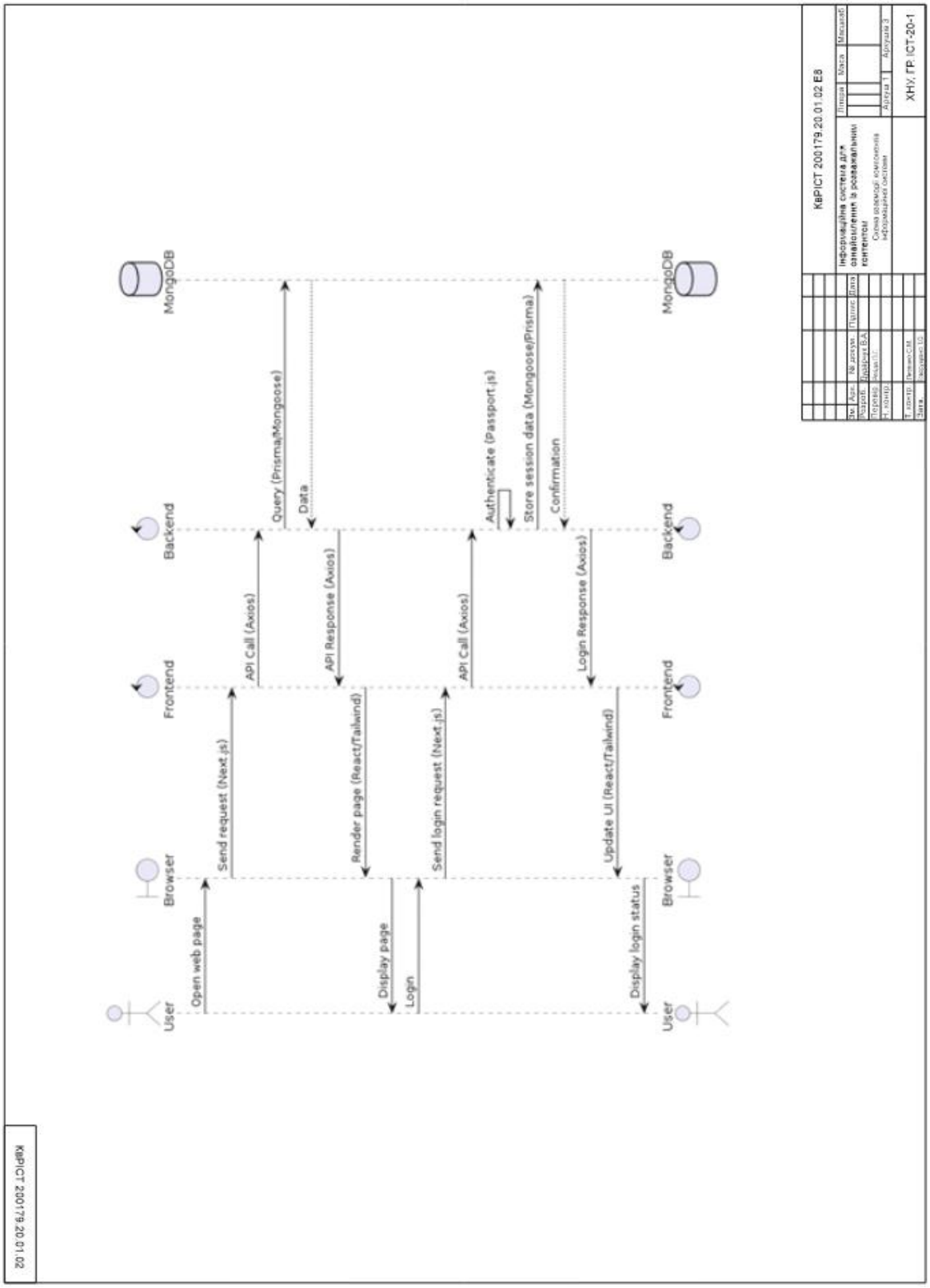
59. React and TypeScript Handbook. URL: https://react-typescript-cheatsheet.netlify.app/docs/basic/setup/ (дата звернення: 16.05.2024)

60. Next.js 14 Full Course 2024 | Build and Deploy a Full Stack App Using the Official React Framework. URL: https://www.youtube.com/watch?v=wm5gMKuwSYk&list=PL6QREj8te1P7gixBDSU8JLvQndTEEX3c3&ab_channel=JavaScriptMastery (дата звернення: 30.05.2024)

ДОДАТОК А

(обов’язковий)

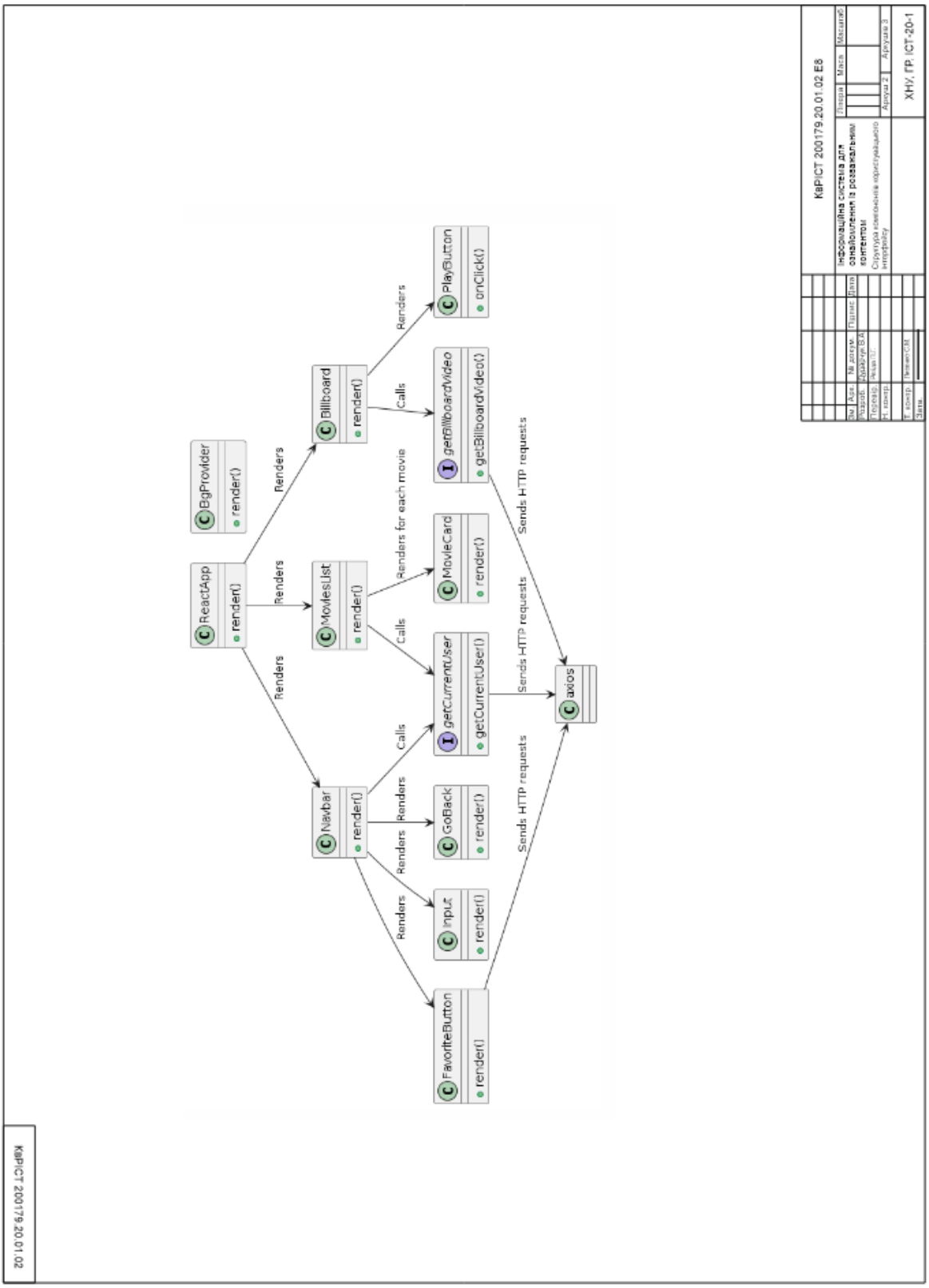
Копія креслення «Схема взаємодії компонентів інформаційної системи»



ДОДАТОК Б

(обов’язковий)

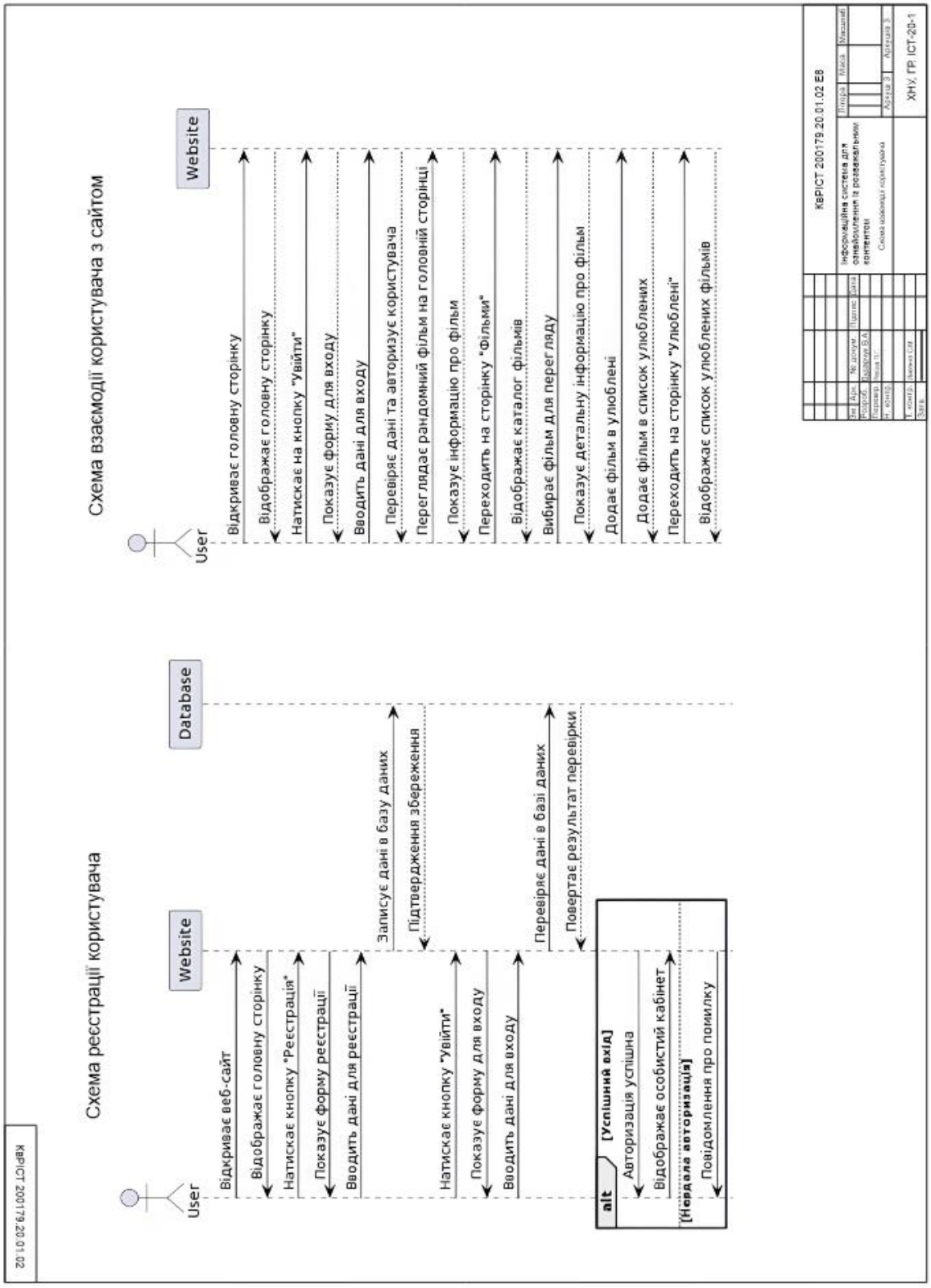
Копія креслення «Структура компонентів користувацького інтерфейсу»



ДОДАТОК В

(обов’язковий)

Копія креслення «Схема взаємодії користувача з сайтом»



ДОДАТОК Г

Код для авторизації:

```
"use client"

import React, { useState } from "react"
import { useRouter } from "next/navigation"
import { signIn } from "next-auth/react"

import BgProvider from "@/app/components/BgProvider"
import Image from "next/image"

import { FaGoogle, FaGithub } from "react-icons/fa"
import Input from "@/app/components/Input"
import axios from "axios"

type VariantType = "login" | "register"

const AuthPage: React.FC = () => {
  const router = useRouter()
  const [variant, setVariant] = useState<VariantType>("login")

  const [email, setEmail] = useState("")
  const [name, setName] = useState("")
  const [password, setPassword] = useState("")

  const handleVariantChange = () => {
    setVariant((prevVariant) =>
      prevVariant === "login" ? "register" : "login"
    )
  }

  const login = async () => {
    try {
      await signIn("credentials", {
        email,
        password,
        redirect: true,
        callbackUrl: "/home",
      })
    } catch (error) {
```

```
    console.log("login error", error)
   }
  }

  const register = async () => {
   try {
    await axios.post("/api/register", {
     name,
     email,
     password,
    })

    login()
   } catch (error) {
    console.log("login error", error)
   }
  }

  return (
   <BgProvider removeBgOnMobile>
    <nav className="px-4 md:px-16 py-6">
     <Image
      onClick={() => router.push("/")}
      width={100}
      height={100}
      src="/images/logo.svg"
      className="h-24 w-36 cursor-pointer"
      alt="logo"
     />
    </nav>

    <div className="flex justify-center">
        <div  className="bg-black/70 p-16 self-center mt-2 lg:w-2/5 lg:max-w-md
rounded-md w-full">
      <h2>{variant === "login" ? "Увійти" : "Зареєструватись"}</h2>
      <div className="flex flex-col gap-4">
       {variant === "register" && (
        <Input
         id="name"
         label="Ім'я"
         type="text"
         value={name}
         onChange={(e: React.ChangeEvent<HTMLInputElement>) =>
          setName(e.target.value)
```

```
            }
          />
        )}
        <Input
          id="email"
          type="email"
          label="Електронна пошта"
          value={email}
          onChange={(e: React.ChangeEvent<HTMLInputElement>) =>
            setEmail(e.target.value)
          }
        />
        <Input
          id="password"
          type="password"
          label="Пароль"
          value={password}
          onChange={(e: React.ChangeEvent<HTMLInputElement>) =>
            setPassword(e.target.value)
          }
        />
      </div>

      <button
        onClick={variant === "login" ? login : register}
        className="bg-red-600 py-3 text-white rounded-md w-full mt-10 hover:bg-red-
700 transition"
      >
        {variant === "login" ? "Увійти" : "Зареєструватись"}
      </button>

      <div className="flex flex-row items-center gap-4 mt-8 justify-center">
        <div
          className="
            w-10
            h-10
            bg-white
            rounded-full
            flex
            items-center
            justify-center
            cursor-pointer
            hover:opacity-80
            transition"
```

```
            onClick={() => signIn("google", { callbackUrl: "/home" })}
          >
            <FaGoogle size={32} />
          </div>
          <div
            className="
              w-10
              h-10
              bg-white
              rounded-full
              flex
              items-center
              justify-center
              cursor-pointer
              hover:opacity-80
              transition"
            onClick={() => signIn("github", { callbackUrl: "/home" })}
          >
            <FaGithub size={32} />
          </div>
        </div>

        <p className="text-neutral-500 mt-12">
          {variant === "login" ? "Уперше у нас?" : "Уже маєш акаунт?"}
          <span
            onClick={handleVariantChange}
            className="text-white ml-1 cursor-pointer hover:underline transition"
          >
            {variant === "login" ? "Зареєструватись" : "Увійти"}
          </span>
        </p>
      </div>
    </div>
  </BgProvider>
 )
}

export default AuthPage
```

Код API авторизації

```
import bcrypt from "bcrypt"
import NextAuth, { AuthOptions } from "next-auth"
```

```
import CredentialsProvider from "next-auth/providers/credentials"
import GithubProvider from "next-auth/providers/github"
import GoogleProvider from "next-auth/providers/google"
import { PrismaAdapter } from "@next-auth/prisma-adapter"

import prisma from "@/app/utils/prismadb"

export const authOptions: AuthOptions = {
  adapter: PrismaAdapter(prisma),
  providers: [
    GithubProvider({
      clientId: process.env.NEXT_AUTH_GITHUB_CLIENT_ID as string,
      clientSecret: process.env.NEXT_AUTH_GITHUB_CLIENT_SECRET as string,
    }),
    GoogleProvider({
      clientId: process.env.NEXT_AUTH_GOOGLE_CLIENT_ID as string,
      clientSecret: process.env.NEXT_AUTH_GOOGLE_CLIENT_SECRET as string,
    }),
    CredentialsProvider({
      name: "credentials",
      credentials: {
        email: { label: "email", type: "email" },
        password: { label: "password", type: "password" },
      },
      async authorize(credentials) {
        if (!credentials?.email || !credentials?.password) {
          throw new Error("Invalid credentials")
        }

        const user = await prisma.user.findUnique({
          where: {
            email: credentials.email,
          },
        })

        if (!user || !user?.hashedPassword) {
          throw new Error("Invalid credentials")
        }

        const isCorrectPassword = await bcrypt.compare(
          credentials.password,
          user.hashedPassword
        )
```

```
        if (!isCorrectPassword) {
          throw new Error("Invalid credentials")
        }

        return user
      },
    }),
  ],

  pages: {
    signIn: "/auth",
  },

  session: { strategy: "jwt" },
  jwt: {
    secret: process.env.NEXT_AUTH_JWT_SECRET,
  },

  secret: process.env.NEXT_AUTH_SECRET,
}

const handler = NextAuth(authOptions)

export { handler as GET, handler as POST }
```

Код API реєстрації

```
import bcrypt from "bcrypt"
import { NextRequest, NextResponse } from "next/server"

import prisma from "@/app/utils/prismadb"

export async function POST(request: NextRequest) {
  try {
    const body = await request.json()
    const { email, name, password } = body

    if (!email || !name || !password) {
      return NextResponse.error()
    }

    const hashedPassword = await bcrypt.hash(password, 12)
```

```
    const user = await prisma.user.create({
      data: {
        email,
        name,
        hashedPassword,
        emailVerified: new Date(),
      },
    })

    return NextResponse.json(user)
  } catch (error) {
    return NextResponse.error()
  }
}
```

Код Schema.prisma

```
generator client {
  provider = "prisma-client-js"
}

datasource db {
  provider = "mongodb"
  url      = env("DATABASE_URL")
}

model User {
  id            String    @id @default(auto()) @map("_id") @db.ObjectId
  name          String
  email         String?   @unique
  image         String?   @unique
  emailVerified  DateTime?
  hashedPassword String?
```

```
  createdAt      DateTime  @default(now())
  updatedAt      DateTime  @updatedAt
  accounts       Account[]
  favoriteIds    String[]  @db.ObjectId
}

model Account {
  id               String  @id @default(auto()) @map("_id") @db.ObjectId
  userId           String  @db.ObjectId
  type             String
  provider         String
  providerAccountId String
  refresh_token     String? @db.String
  access_token      String? @db.String
  expires_at        Int?
  token_type        String?
  scope             String?
  id_token          String? @db.String
  session_state     String?

  user User @relation(fields: [userId], references: [id], onDelete: Cascade)

  @@unique([provider, providerAccountId])
}

model Movie {
  id          String @id @default(auto()) @map("_id") @db.ObjectId
  title       String
  description  String
```

```
  videoUrl     String
  thumbnailUrl String
  genre        String
  duration     String
}
```

UNICHECK
by Turnitin

Ім'я користувача:
Кафедра КІ

ID перевірки:
1016337494

Дата перевірки:
09.06.2024 11:36:16 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
09.06.2024 15:42:08 EEST

ID користувача:
100005591

Назва документа: Дударчук_Інформаційна система для ознайомлення з розважальним контентом

Кількість сторінок: 66 **Кількість слів:** 11028 **Кількість символів:** 86382 **Розмір файлу:** 12.36 MB **ID файлу:** 1016138476

5.37%
Схожість

Найбільша схожість: 0.54% з джерелом з Бібліотеки (ID файлу: 1015083752)

5.08% Джерела з Інтернету	750	Сторінка 68
1.32% Джерела з Бібліотеки	93	Сторінка 72

0.15% Цитат

Цитати	1	Сторінка 73

Не знайдено жодних посилань

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	17

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 1.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 11%**

	Документ		Сумарний збіг по Базі Даних	
ID: 129188 Назва: БКР Інформаційна система для ознайомлення з розважальним контентом Додано в БД: 2024-06-09 Автора: В.А. Дударчук Керівники: П. Г. Регіда Консультанти: Опоненти:	Символи	Лексеми	Символи	Лексеми
	77111	640	1206 (2%)	17 (3%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Інформаційна система для ознайомлення із розважаним контентом

Автор: Дударчук Валерій Анатолійович

Спеціальність: 126– Інформаційні системи та технології

Освітня програма: освітньо-професійна

Науковий керівник: Регіда Павло Геннадійович

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) запозичення розміщені в розділах аналізу існуючих аналогів та прототипів, які не описують безпосередньо авторське дослідження і не стосуються результатів роботи;
2) усі запозичення фрагментарні, або мають належним чином оформленні посилання;
3) окремі виявлені збіги є загальновживаними фразами або виразами;

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості, складає 5.37% і адресується до 843 першоджерела, що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ П. Г. Регіда

Гарант ОП _____ Є.Г. Гнатчук

Завідувач кафедри КІІС _____ Т. О. Говорущенко

Завідувачу кафедри КІІС
д-р.техн.наук, проф. Говорущенко Т. О.

Дударчука Валерія Анатолійовича
ПІБ здобувача вищої освіти

ФІТ, 4 курсу, групи ІСТ-20-1

ЗАЯВА

З правилами чинного Положення «Про систему забезпечення академічної доброчесності у Хмельницькому національному університеті» від 01.07.2022, згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений (а). Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіат оповіщений(а) та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

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

3 червня 2024 року

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Інформаційна система для ознайомлення із розважаним контентом

Автор: Дударчук Валерій Анатолійович

Спеціальність: 126– Інформаційні системи та технології

Освітня програма: освітньо-професійна

Науковий керівник: Регіда Павло Геннадійович

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) запозичення розміщені в розділах аналізу існуючих аналогів та прототипів, які не описують безпосередньо авторське дослідження і не стосуються результатів роботи;
2) усі запозичення фрагментарні, або мають належним чином оформленні посилання;
3) окремі виявлені збіги є загальновживаними фразами або виразами;

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості, складає 5.37% і адресується до 843 першоджерела, що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ П. Г. Регіда

Гарант ОП _____ Є.Г. Гнатчук

Завідувач кафедри КІІС _____ Т. О. Говорущенко

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дипломник: Дударчук Валерій Анатолійович

Тема: Інформаційна система для ознайомлення з розважальним контентом

Спеціальність**:** 126 «Інформаційні системи та технології»

Обсяг кваліфікаційної роботи:

Кількість листів креслень ___3____ Кількість сторінок записки ___60___

1. Короткий зміст роботи та прийнятих рішень: Метою кваліфікаційної роботи є інформаційна система для ознайомлення з розважальним контентом.

2. Висновок про відповідність роботи дипломному завданню: Робота повністю відповідає поставленому завданню.

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи: У першому розділі кваліфікаційної роботи проведено дослідження предметної області, зокрема, аналіз існуючих аналогів сайтів розважального контенту, таких як YouTube, Amazon Prime Video, HBO Max та Disney+. Цей аналіз допоміг визначити ключові аспекти їхньої функціональності, інтерфейсу та взаємодії з користувачем. У другому розділі кваліфікаційної роботи виконано постановку задачі та представлено вимогу до інформаційної системи. Також визначено основні компоненти системи, їх призначення в системі, а також графічно представлено взаємодію цих компонентів у системі. Розглянуто деталі розробки функціоналу надання користувачам зручного доступу до телесеріалів та фільмів, з можливістю створення особистих облікових записів, перегляду інформації про свої вподобання, додавання матеріалів до списку обраного. У третьому розділі кваліфікаційної роботи представлено програмну реалізацію інформаційної системи. Фронтенд системи був розроблений за допомогою сучасних фреймворків та бібліотек для забезпечення швидкої роботи та зручного інтерфейсу. Основними завданнями фронтенду є відображення списків фільмів, пошук та фільтрація контенту, управління профілем користувача та іншими інтерактивними елементами. Бекенд системи забезпечує обробку даних та взаємодію з базою даних MongoDB, яка зберігає інформацію про користувачів, фільми та їх

взаємодії, використовуючи алгоритм хешування bcrypt для захисту паролів. Взаємодія компонентів системи відбувається через API, що забезпечує легку інтеграцію нових функцій та модулів. Використання ORM інструменту Prisma спрощує роботу з базою даних, підвищуючи ефективність розробки та підтримки системи.

4. Позитивні сторони роботи: Реалізовано зручний та приємний інтерфейс, можливість перегляду відео, додавання його в список улюблених, а також можливість створення особистого кабінету.

5. Негативні сторони роботи: відсутність Не реалізовані функції пошуку відео та можливість користувача залишати відгук.

6. Оцінка графічного оформлення та пояснювальної записки роботи: Пояснювальна записка оформлена коректно, згідно діючих стандартів оформлення документації.

7. Відгук про роботу в цілому: Робота виконана на належному технічному рівні.

8. Інші зауваження: _____

9. Оцінка дипломної роботи: Розглянувши позитивні та негативні сторони представленої дипломної роботи вважаю, що робота заслуговує оцінки «добре», 4.00 (С).

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) Гецко М.В
ст. викладач ф.ф. фіскомпрії кібербезпеки

"12" 06 2024 р. _____(підпис)