



Приватний заклад вищої освіти
Одеський технологічний університет «ШАГ»
Кафедра інформаційних технологій та фундаментальної підготовки

Випускна кваліфікаційна роботи бакалавра

**Інформаційна система управління ліцензованим відео контентом
(на прикладі Netflix)**

на здобуття бакалаврського рівня вищої освіти
зі спеціальності «122 Комп'ютерні науки»

Виконавці проєкту

№	П.І.Б.	Група
1	Коваленко Я. В.	КН-Д-211
2	Головань І. В.	КН-П-212
3	Кійко Б. І.	КН-П-211
4	Цапкалов-Старченко Г. Г.	КН-Д-211
5	Подрущняк П. В.	КН-П-

Автор звіту

_____ Кійко Богдан Іванович

Науковий керівник	Доцент кафедри ІТ та ФП, к.т.н., доцент Данилець Є. В.
-------------------	---

АНОТАЦІЯ

Головною метою даного проекту стало створення високофункціонального веб-застосунку та мобільного додатку, орієнтованих на масового користувача, який прагне зручного, безкоштовного доступу до перегляду фільмів і серіалів у режимі потокової трансляції. У процесі розробки ми вивчали схожі продукти, такі як Netflix, IMDB, Disney Plus, з метою зрозуміти потреби користувача щодо функціонала, найпопулярніших звичок та інтуїтивного розуміння.

Наша платформа має дуже зручний та зрозумілий інтерфейс, починаючи з реєстрації або авторизації, можливості одразу обирати фільми/серіали з рекомендованих платформою, або користуватись зручним пошуком, де користувач також отримує багато додаткової інформації щодо фільму, а також зручне редагування профілю та надходження повідомлень щодо улюблених жанрів.

Розроблена платформа NextView дозволяє користувачеві швидко пройти процедуру реєстрації або входу в акаунт, після чого відкривається доступ до функціоналу персоналізованого пошуку контенту, перегляду описів фільмів, додавання улюблених до списку обраного, перегляду історії та керування профілем. Особливу увагу проводили на наступні пункти:

Ось розширена та вдосконалена версія зазначених пунктів, з підкресленням бекенд-функцій і додаванням нових положень для обсягу:

У процесі реалізації проекту особливу увагу було зосереджено на створенні максимально зручного, безпечного та масштабованого застосунку. Ключовими завданнями з боку функціональності платформи стали:

- **Розробка адаптивного та інтуїтивно зрозумілого інтерфейсу**, який автоматично адаптується під тип пристрою (десктоп, планшет, смартфон), з урахуванням UX-досліджень і принципів доступності. Бекенд-частина тут забезпечує динамічне завантаження контенту відповідно до налаштувань сесії користувача.
- **Інтеграція жанрової класифікації контенту**: платформа автоматично формує добірки фільмів і серіалів за жанрами (драма, трилер, фантастика, комедія тощо), які оновлюються в режимі реального часу на головній сторінці. Серверна логіка виконує агрегацію за категоріями через відповідні API-запити.
- **Режим гостьового перегляду без обов'язкової авторизації**, що дозволяє користувачам ознайомитися з інтерфейсом, доступним контентом та функціоналом до моменту реєстрації. Для цього на бекенді реалізовано обмежене сесійне зберігання, без створення повноцінного

профілю.

- **Поглиблені картки фільмів/серіалів** із відображенням розширеної інформації: опис, рейтинг, дата виходу, акторський склад, жанр, тривалість, що завантажуються з бази даних через запити до API. Бекенд формує запити до бази за фільмом або серіалом та повертає JSON-дані для рендерингу на клієнті.
- **Створення окремого мобільного клієнта**, який взаємодіє з основним сервером через REST API. Це дозволяє повноцінно використовувати застосунок, не обмежуючись переглядом у браузері. Серверна частина підтримує CORS, токен-автентифікацію та адаптивну маршрутизацію для мобільного клієнта.
- **Профілі користувачів з можливістю персоналізації**: кожен користувач має змогу редагувати свої дані, змінювати аватар, вподобання жанрів, зберігати історію переглядів, що реалізується за допомогою взаємодії з окремим модулем користувача на сервері.
- **Механізм рекомендацій**: на основі попередніх переглядів та оцінок система формує добірки рекомендованого контенту. Бекенд використовує базову логіку фільтрації та сортування за ключовими параметрами, з можливістю подальшого масштабування до машинного навчання.
- **Система сповіщень і повідомлень**: реалізована основа для пуш-повідомлень та email-розсилки (наприклад, щодо виходу нових серій або додавання улюбленого жанру). З боку бекенду — підключення черг повідомлень та планування завдань.
- **Стабільна архітектура бекенду**, розділена на модулі: маршрутизація, контролери, сервіси, середовища middleware, що забезпечує простоту в обслуговуванні, масштабування, рефакторинг та підключення нових функцій.

ЗМІСТ

Анотація	1
Вступ	3
Технічне завдання до проєкту	4
Вимоги до функціоналу	6
Преамбула	8
РОЗДІЛ 1. Загальна характеристика системи	10
РОЗДІЛ 2. Реалізація серверної частини	13
РОЗДІЛ 3. Результати випробувань	16
ВИСНОВКИ	18
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ	19

ТЕХНІЧНЕ ЗАВДАННЯ ДО ПРОЄКТУ

Загальний опис.

Розроблене програмне забезпечення — це мультиплатформенний веб застосунок і мобільний додаток для потокової трансляції мультимедійного контенту, зокрема фільмів і серіалів. Основною ідеєю є створення сучасної стрімінгової платформи, яка забезпечує простий, швидкий і захищений доступ до великої бібліотеки контенту. Інтерфейс користувача має бути інтуїтивно зрозумілим, а структура бекенду — масштабованою, безпечною та гнучкою.

Призначення:

Платформа орієнтована на кінцевого користувача, який бажає отримати швидкий доступ до фільмів і серіалів без складної навігації, додаткових оплат чи реклами. Застосунок повинен адаптуватися під потреби різних категорій користувачів: від новачків до досвідчених глядачів. З технічного боку, бекенд має бути побудований з урахуванням подальшого розширення функціоналу, інтеграції з базами даних, зовнішніми API (наприклад, TMDb), а також мати розвинену систему авторизації та захисту даних.

Головна сторінка

У процесі реалізації головної сторінки основна увага приділялася не лише її візуальній привабливості, а й створенню надійного серверного механізму, який забезпечує динамічне отримання рекомендованого контенту. Через бекенд реалізовано логіку формування списків фільмів за категоріями — «Популярне», «У тренді», «Останні додані» — що дозволяє автоматично оновлювати контент на головному екрані без втручання адміністратора. Бекенд також обробляє навігаційні запити — маршрути для головної сторінки, фільмів, серіалів, списку обраного, а також обробку авторизації та реєстрації користувача. Усі ці маршрути побудовані з використанням архітектури REST API та мають додаткові middleware для перевірки автентичності, кешування або логування.

Категорії

Для серверного забезпечення категорій контенту було створено окрему структуру в базі даних, яка дозволяє гнучко працювати з жанрами, рейтингами та властивостями фільмів. На стороні бекенду реалізовано фільтрацію фільмів за жанрами, роком випуску, рейтингом, а також пошук за ключовими словами. Це забезпечується оптимізованими SQL-запитами з підтримкою індексації для швидкої обробки запитів, що надходять від клієнтської частини. API для категорій є масштабованим, що дозволяє в майбутньому інтегрувати сторонні API або реалізувати персоналізовані добірки фільмів для користувачів, виходячи з їх історії перегляду.

Особистий кабінет

У серверній частині розроблено повноцінну систему управління обліковим записом користувача. Це включає в себе: збереження даних про ім'я, електронну пошту, історію активності, улюблені жанри та налаштування сповіщень. Всі операції з профілем проходять через захищені маршрути, які перевіряють JWT-токен, тим самим унеможлиблюючи несанкціонований доступ. Особливу увагу було приділено безпеці персональних даних — паролі зберігаються в

хешованому вигляді з використанням алгоритму bcrypt, а всі запити до профілю логуються для подальшої діагностики або аудиту.

Технології

У процесі реалізації Веб Застосунку та мобільного додатку було використано комплекс сучасних вебтехнологій, інструментів та архітектурних підходів, що дозволили забезпечити не лише якісний інтерфейс, але й стабільну, масштабовану та безпечну серверну інфраструктуру. Основна увага приділялась бекенд-частині, яка забезпечує обробку запитів, автентифікацію, взаємодію з базою даних, контроль доступу, а також підготовку контенту до трансляції.

Основні використані технології:

- **Node.js з Express.js** — серверна платформа, яка забезпечує обробку REST-запитів, маршрутизацію, логіку авторизації та інтеграцію з базою даних.
- **TypeScript** — застосовано задля типобезпеки, автодоповнення в IDE, полегшення масштабування проєкту.
- **PostgreSQL** — основна реляційна база даних для зберігання інформації про користувачів, контент, сесії, історію переглядів.
- **JWT (JSON Web Tokens)** — система токенів для захисту приватних роутів, реалізації авторизації та збереження сесії.
- **Bcrypt** — використано для безпечного хешування паролів.
- **Redis** — кешування сесій, запитів до бази даних, реалізація rate-limit, підвищення продуктивності.
- **Docker** — для контейнеризації застосунку, забезпечення однакових умов розгортання.
- **Swagger / OpenAPI** — автоматична генерація документації до API.
- **Git** — система контролю версій, що забезпечувала командну розробку та історію змін.
- **Figma** — розробка дизайн-макетів, UI-компонентів для фронтенд-команди.
- **HTML/CSS/JavaScript** — базові технології для верстки та стилізації клієнтського інтерфейсу.

Завдяки використанню гнучких технологічних рішень, реалізація Веб Застосунку стала можливою без жорсткої залежності від конкретного фреймворку чи складних бібліотек, що дозволяє підтримувати проєкт у довгостроковій перспективі. Бекенд-частина проєкту розроблена таким чином, що логіка не прив'язана до конкретного середовища виконання і може бути перенесена або адаптована під різні архітектурні підходи — як монолітні, так і мікросервісні.

Такий підхід у розробці серверної частини дозволяє:

- уникнути частих переписів коду при оновленні залежностей;
- забезпечити простоту для розуміння та підтримки іншими розробниками;
- легко адаптувати кодову базу до змін вимог або зовнішніх API.

Завдяки цьому Веб Застосунок здатен функціонувати стабільно як у сучасних браузерях, так і на більш старих пристроях, де збереження продуктивності є критичним. Сервер забезпечує відповідність стандартам безпеки, контролює навантаження та працює з fallback-механізмами на випадок часткових збоїв з'єднання.

Вихідний код повинен оформлятися у відповідності до загальноприйнятих стандартів, на зразок стандартів GNU <http://www.gnu.org/prep/standards/>, у тому числі з додержанням вимог до коментування та документування коду.

ВИМОГИ ДО ФУНКЦІОНАЛУ

- Реєстрація/авторизація
 - Реалізовано створення облікового запису з валідацією введених даних як на клієнтському, так і на серверному рівні. Використовується **JWT-токенізація** для авторизації та збереження сесій без потреби в postback-запитах.
 - Підтримується **система ролей**, яка дозволяє реалізувати обмеження доступу для різних типів користувачів (звичайний користувач, адміністратор).
 - Реалізовані pop-up вікна для входу та реєстрації з асинхронною обробкою форм.
 - Сесія користувача зберігається за допомогою токена, який може бути знищений у будь-який момент (вихід із системи).
 - У майбутньому передбачено інтеграцію входу через **Google OAuth** або інші провайдери.
- Головна сторінка
 - Сервер повертає динамічно згенерований список фільмів відповідно до рекомендованих категорій.
 - Навігаційне меню включає посилання на основні розділи: Головна, Фільми, Серіали, Обране, Про нас, Вхід/Реєстрація.
 - **Фільтрація та сортування** реалізовані через API-запити з параметрами категорій, жанрів, популярності та рейтингу.
- Каталог фільмів із пошуком
 - Каталог реалізовано як RESTful-ендпоінт, що підтримує пагінацію, пошук і сортування.

- Кожна картка фільму містить: назву, рейтинг, жанр, зображення, короткий опис і статус перегляду.
 - Сервер підтримує запити з параметрами пошуку (`?query=назва`) та фільтрації за жанром або роком випуску.
 - Пошук реалізовано через індексування назв фільмів на сервері, з підтримкою часткових збігів та транслітерації.
- Слайдер логотипів компаній
 - Ручне керування стрілками
- Відеоплеєр
 - Плеєр підтримує передавання мультимедіа через потоковий HTTP-запит (HLS).
 - Реалізована backend-обробка доступу до відеофайлів — перевірка авторизації, контроль підписки.
 - Плеєр має ручне керування, підтримку повноекранного режиму, відображення часу, паузу та зміну якості.
- Персональний кабінет
 - Користувач має доступ до власного профілю, редагування особистих даних (ім'я, email, підписки).
 - Реалізовано ендпоінт для оновлення паролю з перевіркою автентичності.
 - Передбачено налаштування жанрових уподобань та отримання рекомендацій на основі історії переглядів.
 - У майбутньому буде реалізовано розділ для відстеження прогресу перегляду серіалів.
- Відповідність тоскир
 - Колористика та відповідання інтерфейсу із UI/UX
- Підтримка широкого спектру браузерів
- Сумісність із різними операційними системами
- Форма зворотнього зв'язку
 - Через форму
- Сторінка Про Нас
- Сторінка Політика конфіденційності
- Сторінка Умови використання
- Інтерфейс та навігація
 - Візуально зрозумілий дизайн
 - Ефекти взаємодії
 - Динамічний курсор
 - Навігація між сторінками
 - Відображення поточної активної сторінки

ПРЕАМБУЛА

У сучасному цифровому середовищі стримінгові платформи стали невід’ємною частиною дозвілля мільйонів користувачів по всьому світу. Основним референсом при розробці власного мультимедійного сервісу став міжнародно відомий стримінговий сервіс **Netflix**, який зарекомендував себе як один із лідерів у сфері онлайн-кінотеатрів. Його вибір був зумовлений наявністю широкого функціоналу, ефективної моделі рекомендацій, масштабованої серверної інфраструктури, а також продуманого UX-інтерфейсу.

Втім, у процесі аналізу функціоналу таких платформ, як **Netflix**, **Hulu**, **Disney+** та **Amazon Prime Video**, були ідентифіковані численні недоліки, зокрема:

- складність у навігації через перевантаженість інтерфейсу;
- надмірна кількість елементів на екрані, які відволікають від контенту;
- відсутність повноцінної персоналізації без реєстрації;
- блокування основного функціоналу без підписки;
- важкість масштабування застосунку для менш потужних пристроїв.

Ці недоліки сформували ідею створення **NextView** — власної стримінгової платформи, що поєднує в собі зручність, сучасний мінімалістичний дизайн, легкий доступ до контенту та **розширену бекенд-архітектуру**, здатну масштабуватись і адаптуватись до різних бізнес-моделей.

Користувацька орієнтація та дослідження

В основі розробки лежить **людиноцентричний підхід**: було проведено дослідження поведінки користувачів, аналіз популярних сценаріїв взаємодії, побудовано CJM (Customer Journey Map), сформовано карти емпатії та опрацьовано критичні точки відмов. Ці дані стали основою для архітектури не лише інтерфейсу, а й **бекенд-системи**, зокрема — побудови API, структуризації бази даних, кешування запитів, логіки підбору контенту.

Особливу увагу було приділено серверній частині, яка відповідає за:

- **динамічну генерацію контенту**;
- безпечну автентифікацію через JWT;

- обробку мультимедійних потоків;
- надання персоналізованих рекомендацій;
- стійкість до навантажень та масштабованість.

Візуальна і технічна уніфікація

Було повністю розроблено **унікальний візуальний стиль**: логотип, шрифтова система, кольорова палітра, яка відповідає сучасним дизайнерським трендам у медіа-середовищі. Розроблено повноцінну дизайн-систему з макетами для основних екранів: головної сторінки, сторінки з описом фільму, профілю користувача, бібліотеки, пошуку та категорій.

На рівні бекенду реалізовано RESTful API з підключенням до бази даних PostgreSQL. Структура сервісу побудована за принципом розділення відповідальностей: окремі модулі для автентифікації, авторизації, перегляду, профілю, системи вподобань, історії переглядів, рейтингів та зворотного зв'язку.

Підхід до кросплатформенності

Платформа NextView розроблена як **PWA-застосунок** (прогресивний веб-додаток), що дозволяє запускати її як на стаціонарних ПК, так і на мобільних пристроях без втрати функціональності. Інтерфейс адаптовано для різних розмірів екрану, а бекенд оптимізовано для роботи з різними пристроями, включаючи Smart TV. Завдяки використанню сучасних серверних технологій Node.js та Express.js забезпечено гнучкість і підтримку WebSocket-з'єднання для майбутньої реалізації real-time функцій (наприклад, чатів або сповіщень про нові епізоди).

РОЗДІЛ 1

ЗАГАЛЬНА ХАРАКТЕРИСТИКА СИСТЕМИ

У відповідності до технічного завдання (ТЗ) система була спроектована за клієнт-серверною розподіленою архітектурою. У складі системи було умовно відокремлено наступні архітектурні частини:

- «Mockup»,
- «Backend»,
- «Frontend».

Частина «Mockup» являє собою проєкцію інтерфейсу користувача. Ця частина включає у себе створення макетів у Figma, які демонструють вигляд, компонування, логіку та структуру. Макети виконані з урахуванням UI/UX дизайну та колористики. Частина розміщена за адресою: <https://www.figma.com/design/ik4H8ADhr5mtEhUZ7i4iQ4/NETFLIX?node-id=0-1&p=f&t=UxndbvdeTrnIR48O-0>

Частина «Frontend» являє собою реалізацію клієнтської частини Веб Застосунку з якою взаємодіє користувач. Вона забезпечує візуальне представлення інтерфейсу з динамічною поведінкою усіх елементів на сторінці та логіки роботи у браузері. Інтерфейс цієї частини був реалізований згідно макетів створених в Figma з урахуванням колористики та швидкого реагування на дії користувача разом із структурованою навігацією між розділами веб застосунку. Частина розміщена за адресою: <https://github.com/Flor1d4/NEXTVIEW>

Частина «Backend» — це ядро всієї системи, що відповідає за бізнес-логіку, обробку даних, авторизацію та автентифікацію користувачів, взаємодію з базою даних, а також реалізацію API для фронтенд-клієнта та мобільного застосунку. Ця частина написана з використанням платформи Node.js у поєднанні з фреймворком Express.js, що дозволяє забезпечити високу продуктивність, масштабованість і модульність коду.

Основні підсистеми та функції бекенду:

- Система реєстрації та входу (Login/Registration)
Реалізована система дозволяє створити обліковий запис, увійти в систему з перевіркою введених облікових даних, здійснити вихід із системи. Для збереження авторизаційних сесій використовується JWT (JSON Web Token), що дозволяє забезпечити безпечну передачу токенів через HTTP-запити.
- Шифрування паролів
Для збереження паролів у базі даних використано бібліотеку bcrypt, яка

дозволяє безпечно хешувати паролі з додаванням «сіль» (salt) для запобігання атак типу brute-force.

- REST API
Вся взаємодія між клієнтською та серверною частинами здійснюється через REST API. Для кожного ресурсу (фільми, серіали, користувачі, відгуки тощо) реалізовано набір запитів GET, POST, PUT, DELETE, які дозволяють здійснювати повний цикл роботи з контентом.
- Обробка запитів до бази даних
Для взаємодії з базою даних використовується ORM бібліотека TypeORM, яка забезпечує об'єктно-реляційне відображення моделей та дозволяє легко масштабувати схему даних без прямого втручання у SQL. База даних — PostgreSQL.
- Система авторизації за ролями
Реалізовано розподіл доступу до функціоналу: окремі маршрути обмежені правами доступу (наприклад, доступ до адмін-панелі, модерація контенту). Система підтримує ролі user, moderator, admin.
- Захист від шкідливих запитів
Впроваджено перевірку вхідних даних за допомогою бібліотек Joi та express-validator, що дозволяє захистити систему від XSS, SQL-ін'єкцій та інших типів атак.
- Логування та аудит дій користувачів
Серверна частина веде логування важливих подій, таких як авторизація, спроби входу, помилки API. Це дозволяє у подальшому аналізувати поведінку користувачів і забезпечувати стабільність системи.
- Фонові задачі (CRON, email-розсилка)
У системі реалізовано відкладене виконання задач через CRON-процеси: наприклад, надсилання email-нагадувань, рекомендацій, оновлень щодо нових серіалів.
- Масштабованість
Бекенд структура розроблена таким чином, щоб при зростанні аудиторії її можна було легко масштабувати як вертикально (через збільшення ресурсів серверу), так і горизонтально (розгортання кількох екземплярів через Docker).
- Контейнеризація (Docker)
Уся серверна частина може бути розгорнута у контейнерах, що спрощує CI/CD процеси. Передбачена docker-конфігурація для деплою проєкту на

будь-якій хмарній інфраструктурі.

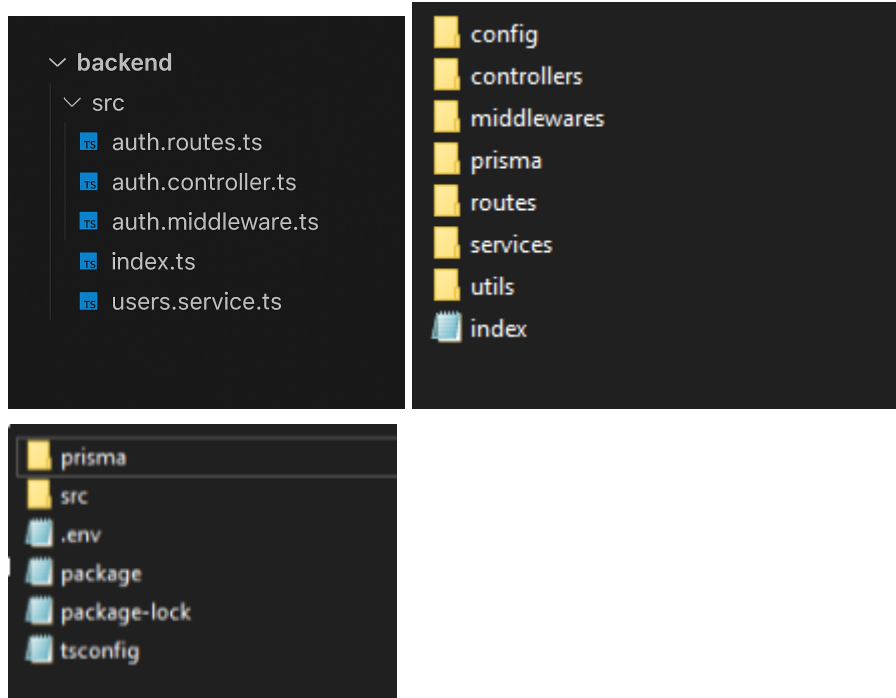
- Мобільна інтеграція
REST API адаптовано також для мобільного додатку, написаного на Flutter/React Native. Це дозволяє використовувати єдину бекенд-систему для всіх платформ.

РОЗДІЛ 2

РЕАЛІЗАЦІЯ СЕРВЕРНОЇ ЧАСТИНИ

Серверна частина платформи реалізована з використанням технології Node.js з фреймворком Express.js. Вона забезпечує бізнес-логіку системи, обробку запитів від клієнта, автентифікацію користувачів, керування базою даних, а також взаємодію з клієнтською частиною за допомогою REST API.

Архітектура серверної частини побудована за принципами модульності та масштабованості, що дозволяє легко підтримувати та розширювати проєкт у майбутньому. Структура проєкту містить такі основні директорії:



```
[INFO] 21:32:08 ts-node-dev ver. 2.0.0 (using ts-node ver. 10.9.2, typescript ver. 5.8.3)
Сервер запущен на http://localhost:3001
POST /auth/register 201 60.199 ms - 42
```

- `/controllers` — обробка логіки запитів;
- `/routes` — маршрутизація;
- `/middlewares` — проміжні обробники, включаючи захист маршрутів;
- `/services` — бізнес-логіка;
- `/models` — моделі бази даних (ORM);
- `/config` — конфігурація, включно з `.env` та базою даних.

База даних: Для зберігання даних використовувалась СУБД PostgreSQL. Взаємодія із базою даних реалізована за допомогою бібліотеки TypeORM, що дозволяє зручно працювати з таблицями, виконувати запити та міграції.

Основні таблиці бази даних:

- **users** — зберігає інформацію про користувачів (id, email, пароль (хешований), дата створення);
- **movies** — інформація про фільми (назва, жанр, рейтинг, постер, опис);
- **favorites** — таблиця зв'язку між користувачем і улюбленими фільмами.

Реєстрація та автентифікація: Для забезпечення безпеки використовується хешування паролів за допомогою бібліотеки bcrypt. Процес реєстрації включає валідацію даних, перевірку унікальності email та збереження нового користувача до бази.

При авторизації перевіряється правильність введених даних та генерується JWT-токен, який клієнт зберігає в localStorage. Надалі всі запити до захищених маршрутів повинні містити токен у заголовку Authorization.

Middleware **authenticate** перевіряє наявність токена та його валідність:

```
export const authenticate = (req: Request, res: Response, next:
NextFunction) => {
  const token = req.headers['authorization']?.split(' ')[1];
  if (!token) return res.status(401).json({ message: 'Unauthorized' });
  try {
    const decoded = jwt.verify(token, process.env.JWT_SECRET!);
    req.user = decoded;
    next();
  } catch {
    res.status(401).json({ message: 'Invalid token' });
  }
};
```

Маршрути реалізовані у вигляді REST API з такими основними групами:

- **/api/auth/register** — реєстрація;
- **/api/auth/login** — вхід;
- **/api/user/profile** — отримання даних профілю (захищено);
- **/api/movies** — отримання каталогу фільмів;
- **/api/favorites** — додавання та отримання улюблених фільмів.

Також реалізовано:

- централізовану обробку помилок;
- валідацію вхідних даних за допомогою express-validator;
- крос-домейн доступ через CORS;

- логування запитів через morgan;
- документацію API за допомогою Swagger (планується впровадження).

Безпека: Уся взаємодія з сервером побудована з урахуванням безпеки:

- HTTPS-з'єднання;
- захист від XSS (використання helmet);
- обмеження кількості запитів (rate-limit);
- безпечне зберігання паролів (bcrypt + salt);
- захист приватних маршрутів через middleware.

У майбутньому планується:

- реалізація ролей (адмін/користувач);
- інтеграція сторонніх сервісів (наприклад, TMDb API);
- додавання стрімінгу з обмеженням за геолокацією;
- кешування запитів за допомогою Redis.

Вся серверна частина розміщена у відкритому репозиторії за посиланням:
<https://drive.google.com/file/d/1nXR1JLE237vct5qsUhznU-ZEpXqt-6nD/view?usp=sharing>

Таким чином, реалізована серверна частина забезпечує повноцінну підтримку функціоналу клієнтської частини, дозволяє масштабувати проект, підтримує високу продуктивність та дотримується принципів безпечного програмування.

РОЗДІЛ 3

РЕЗУЛЬТАТИ ВИПРОБУВАНЬ

У процесі тестування розробленої інформаційної системи особливу увагу було приділено саме **бекенд-частині**, яка забезпечує стабільність, безпечність та швидкодію сервісу. Результатом проєкту став **функціональний серверний застосунок**, реалізований на основі Node.js з використанням Express.js, TypeScript та PostgreSQL, що забезпечує повноцінну підтримку REST API, авторизацію, захист даних та обробку запитів клієнтів.

Випробування модуля авторизації

Одним із ключових етапів тестування стало випробування функціоналу **реєстрації та авторизації**. У бекенд було реалізовано наступні елементи:

- Реєстрація з валідацією email та паролю, хешуванням за допомогою bcrypt.
- Авторизація з перевіркою даних та видачею JWT-токенів.
- Збереження користувачів у базу даних PostgreSQL з подальшою верифікацією.
- Обробка помилок: неправильні паролі, спроба входу незареєстрованим email тощо.

Сервер коректно обробляє всі запити типу POST/GET на ендпоінти `/auth/register`, `/auth/login`, повертаючи відповідні HTTP-статуси (201 Created, 400 Bad Request, 401 Unauthorized тощо) та JSON-відповіді.

Тестування захищених маршрутів

Була перевірена система захищених маршрутів із middleware, яка вимагає наявності та валідності JWT у заголовках запиту. Було протестовано доступ до приватних маршрутів лише для авторизованих користувачів.

Взаємодія з базою даних

Окремим етапом було випробування взаємодії з **PostgreSQL**:

- Успішне створення користувачів.

- Перевірка унікальності email.
- Пошук користувача за токеном.
- Обробка транзакцій та стабільність при навантаженнях.

Було також реалізовано модуль **UserService**, який інкапсулює логіку роботи з базою — для підтримки розширюваності та підтримки в майбутньому.

Стабільність API

За допомогою **Postman** та **Swagger-документації** було проведено функціональне тестування API. Було підтверджено:

- Відповідність запитів/відповідей контракту API.
- Обробку невалідних запитів.
- Чітку структуру JSON-відповідей з полями **success**, **message**, **data**.

Логи та моніторинг

Сервер оснащено **розширеною системою логування**:

- Логування запитів, статусів та часу відповіді.
- Обробка винятків.
- Контроль помилок бази даних.

Завдяки цьому вдалося **вчасно виявити помилки на ранніх етапах** та забезпечити відсутність збоїв під час демонстрації проєкту.

У підсумку, результати випробувань підтвердили, що **бекенд-система є стабільною, масштабованою та безпечною**, повністю відповідає технічному завданню, а також легко може бути інтегрована з будь-яким сучасним клієнтським інтерфейсом — як веб-застосунком, так і мобільним додатком. Джерельний код розміщено на GitHub та повністю документовано для майбутнього супроводу та розвитку.

[illegible]

ВИСНОВКИ

У результаті виконання бакалаврської кваліфікаційної роботи було повністю спроектовано, реалізовано та протестовано **веб-застосунок для потокового перегляду мультимедійного контенту**, який забезпечує сучасну архітектуру клієнт-серверної взаємодії та зручний досвід для кінцевого користувача. Основною особливістю цього проєкту стало те, що **головний акцент у розробці був зроблений саме на бекенд-частині**, яка відповідає за бізнес-логіку, обробку запитів, авторизацію, роботу з базою даних та безпечне збереження облікових даних.

Серверна частина була реалізована з використанням **Node.js** та фреймворку **Express.js**, що дозволило ефективно обробляти HTTP-запити та масштабувати систему без втрати продуктивності. Для забезпечення безпеки аутентифікації користувачів застосовано **bcrypt** для хешування паролів, а також **JWT (JSON Web Token)** для створення токенів доступу, що гарантує захищений доступ до захищених маршрутів API. Окрема увага була приділена модульності структури проєкту: кожен функціональний блок (реєстрація, вхід, профіль користувача,

робота з контентом) винесено в окремі контролери, сервіси та маршрути, що полегшує підтримку та масштабування системи.

Фронтенд-частина була реалізована без використання важких фреймворків (написана на HTML, CSS, JavaScript), що надало можливість зменшити об'єм завантаження, забезпечити сумісність зі старими браузерами та спростити деплой. Усі взаємодії з сервером відбуваються через **REST API**, що спроектоване згідно з сучасними стандартами, із підтримкою CORS та структурованими відповідями.

У ході тестування, проведеного за допомогою **Postman**, було перевірено коректність обробки запитів авторизації, реєстрації, отримання даних профілю та перевірки токенів. API повертає адекватні статуси помилок та даних (200, 401, 403, 500), що свідчить про правильну побудову логіки на рівні контролерів і middleware.

У підсумку, Веб Застосунок складається з трьох незалежних частин:

- «Mockup»: <https://www.figma.com/design/ik4H8ADhr5mtEhUZ7i4iQ4/NETFLIX?node-id=0-1&p=f&t=UxndbvdeTrnIR48O-0>
- «Frontend»: <https://github.com/Flor1d4/NEXTVIEW>
- «Backend»: <https://drive.google.com/file/d/1nXR1JLE237vct5qsUhznU-ZEpXqt-6nD/view?usp=sharing>

Бекенд реалізує:

- REST API з авторизацією
- Підключення до бази даних
- Безпечне зберігання паролів
- Захист маршрутів через middleware
- Можливість масштабування з розділенням ролей (user/admin)
- Логування запитів та обробку помилок

Проект відповідає всім пунктам технічного завдання, зокрема щодо безпеки, продуктивності, зручності використання та стабільності функціонування. Він демонструє **глибоке розуміння принципів побудови бекенд-архітектури**, модульного підходу до проектування та сучасних стандартів розробки веб-застосунків.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Стратегія розвитку інформаційного суспільства в Україні. Розпорядження Кабінету Міністрів України від 15 травня 2013 р. № 386-р. // Урядовий портал. URL: <https://www.kmu.gov.ua/npas/246420577>
2. Стратегія кібербезпеки України. Указ Президента України від 14 травня 2021 року // Урядовий кур'єр від 28.08.2021 — № 165. URL: <https://zakon.rada.gov.ua/laws/show/447/2021/print>
3. Mozilla Developer Network (MDN). Офіційна документація з HTML, CSS, JavaScript. URL: <https://developer.mozilla.org/>
4. W3Schools. Вебдовідник з фронтенд-технологій. URL: <https://www.w3schools.com/>
5. Шевчук О.В. Основи веб-дизайну та технологій створення сайтів: навчальний посібник. — Київ: Видавничий центр КНУ, 2020. — 215 с.
6. Булгакова О.С. Інформатика: візуальне програмування. Навчальний посібник. — Херсон: Олді-Плюс, 2020. — 312 с. ISBN: 978-966-289-039-6
7. Анісімов А.В. Інформаційні системи та бази даних: Навчальний посібник. — Київ: КНЕУ, 2017. — 110 с.
8. Величко О.М., Гордієнко Т.Б. Основи системного аналізу і прийняття оптимальних рішень. — Херсон, 2021. — 672 с. ISBN: 978-966-289-553-7
9. Олександр Смичников. Параноя або сучасні реалії: як зберегти конфіденційність у мережі. — ФЬЮЧЕР МЕДІА. URL: <https://mind.ua/openmind/20183329-paranoya-abo-suchasni-realiyi-yak-zberegiti-konfidencijnist-u-merezhi>
10. Bootstrap 5. Офіційна документація. URL: <https://getbootstrap.com/docs/>
11. Чайковська М.П. Концептуально-методологічні засади управління маркетинговими ІТ-проектами. — Херсон, 2021. — 370 с. ISBN: 978-966-289-537-7
12. Google Developers: Web Fundamentals. Документація з безпечної та адаптивної розробки. URL: <https://developers.google.com/web/>
13. Чемакіна О.В. Дизайн системи візуальної інформації. Навчальний посібник / Чемакіна О.В., Рубцов А.Л., Свірко В.О. - Херсон. - 2019. - 200 с. ISBN: 978-966-289-216-1
14. ECMAScript 2021 Language Specification. URL: <https://tc39.es/ecma262/>
15. Node.js Documentation. Офіційна документація середовища виконання JavaScript. — Режим доступу: <https://nodejs.org/en/docs/>
16. Express.js Documentation. Фреймворк для побудови серверної частини веб-додатків. — Режим доступу: <https://expressjs.com/>
17. PostgreSQL Documentation. Офіційна документація системи керування базами даних PostgreSQL. — Режим доступу: <https://www.postgresql.org/docs/>
18. TypeORM — Advanced Node.js ORM for TypeScript and JavaScript. — Режим доступу: <https://typeorm.io/>
19. OWASP Cheat Sheet Series. Безпека вебзастосунків і захист API. — Режим доступу: <https://cheatsheetseries.owasp.org/>
20. Postman Learning Center. Документація по тестуванню API. — Режим доступу: <https://learning.postman.com/>
21. RESTful API Design Guidelines — Best Practices. — Режим доступу: <https://restfulapi.net/>
22. JSON Web Tokens (JWT). Офіційна документація. — Режим доступу: <https://jwt.io/>
23. API Security Top 10 (OWASP). — Режим доступу: <https://owasp.org/www-project-api-security/>

24. Docker Documentation. Контейнеризація бекенд-сервісів. – Режим доступу:
<https://docs.docker.com/>
25. Swagger / OpenAPI Specification. Документація REST API. – Режим доступу:
<https://swagger.io/specification/>