

ПРИВАТНИЙ ЗАКЛАД ВИЩОЇ ОСВІТИ
ОДЕСЬКИЙ ТЕХНОЛОГІЧНИЙ УНІВЕРСИТЕТ «ШАГ»
Кафедра інформаційних технологій та фундаментальної підготовки

Випускна кваліфікаційна робота бакалавра

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

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

Виконавець проекту:
Подрушняк Павло Володимирович
Група: КН-П-212

Керівник	Доцент кафедри ІТ та ФП, к.т.н., доцент Новак С.М.
----------	---

Анотація

Метою цієї кваліфікаційної роботи стало проєктування та реалізація серверної частини (бекенду) для стримінгового вебсервісу, орієнтованого на користувачів, що бажають мати зручний, швидкий і безкоштовний доступ до фільмів та серіалів. У рамках розробки було створено функціональний бекенд, що забезпечує взаємодію між клієнтським інтерфейсом, базою даних і зовнішніми сервісами.

Система підтримує такі функції, як реєстрація та автентифікація користувачів, керування колекцією відео, сортування за категоріями, рейтингову систему, рекомендації, список улюбленого контенту, профілі користувачів та зв'язок із фронтом через REST API. Для зберігання даних використано MongoDB, бекенд реалізовано на Node.js (Express.js). Уся бізнес-логіка винесена в окремі модулі, що сприяє масштабованості та повторному використанню коду.

Основний акцент зроблено на безпеці (JWT, валідація запитів), стабільності API, структурованій архітектурі (MVC), а також на підтримці тестованості та читабельності коду.

1. Постановка задачі

Сформулювати серверну частину платформи, що має забезпечити повну інтеграцію з клієнтським інтерфейсом, обробку даних користувача, пошук контенту, доступ до метаданих відео, персональні рекомендації, а також захист приватних даних.

2. Архітектура системи

Було реалізовано багаторівневу архітектуру типу MVC:

- Model: схема контенту, користувача, списків, рейтингів (MongoDB, Mongoose)
- View: (відсутня на бекенді – замінено API)
- Controller: обробка бізнес-логіки
- Router: маршрутизація REST API з підтримкою CRUD

Додатково впроваджено middleware для логування запитів, автентифікацію через JWT, валидацію вхідних даних за допомогою express-validator.

3. Реалізація бекенду

Основні модулі:

- /auth: реєстрація, логін, logout, оновлення токена
- /movies: CRUD-контролери для роботи з фільмами
- /user: отримання/оновлення профілю, улюблені фільми
- /admin: панель для модерації (обмежений доступ)
- /search: повнотекстовий пошук

Усі контролери ретельно розділені на окремі файли, логіка ізольована, API відповідає принципам REST.

4. Функціональні можливості API

- Аутентифікація/авторизація з JWT
- Пошук фільмів по назві, жанру, рейтингу
- Збереження обраного
- Рекомендації за жанром
- CRUD-функціонал (тільки для адміністратора)
- Логування помилок

5. Безпека

- JWT (JSON Web Tokens): для захисту сесій користувачів
- Шифрування паролів через bcrypt
- CORS-захист для запитів із frontend
- Rate Limiting для запобігання спаму
- Валідація даних (email, паролі, ролі)

6. Тестування

Тестування проведено вручну через Postman та Swagger (документація API). Усі маршрути перевірено на стабільність та правильність відповіді. Також проведено просте модульне тестування на рівні контролерів.

7. Висновки та перспективи

Серверна частина платформи розроблена згідно з сучасними стандартами вебпрограмування. Вона є масштабованою, безпечною та легко адаптується під зміну бізнес-логіки. У майбутньому можливе:

- Підключення зовнішнього відеохостингу або CDN
- Додавання WebSocket для підтримки чату
- Інтеграція з платіжними сервісами
- Масштабування під мікросервіси

Проект створює міцну технічну основу для розвитку повноцінної стримінгової платформи.

Приклад реалізації контролера авторизації користувача:

```
exports.login = async (req, res) => {
  const { email, password } = req.body;
  try {
    const user = await User.findOne({ email });
    if (!user || !(await bcrypt.compare(password, user.password))) {
      return res.status(401).json({ message: 'Invalid credentials' });
    }
    const token = jwt.sign({ id: user._id }, process.env.JWT_SECRET, {
      expiresIn: '1h' });
    res.json({ token });
  } catch (error) {
```

```
    res.status(500).json({ message: 'Server error' });
  }
};
```

Фрагмент маршруту для додавання фільму (тільки для адміністратора):

```
router.post('/add', authMiddleware, adminMiddleware, async (req, res) =>
{
  const { title, description, genre } = req.body;
  try {
    const movie = new Movie({ title, description, genre });
    await movie.save();
    res.status(201).json(movie);
  } catch (error) {
    res.status(400).json({ error: 'Cannot add movie' });
  }
});
```

Налаштування JWT-автентифікації:

```
const jwt = require('jsonwebtoken');

module.exports = (req, res, next) => {
  const token = req.header('Authorization')?.split(' ')[1];
  if (!token) return res.status(401).json({ message: 'No token provided'
});
  try {
    const decoded = jwt.verify(token, process.env.JWT_SECRET);
    req.user = decoded;
    next();
  } catch (error) {
    res.status(403).json({ message: 'Invalid token' });
  }
};
```