

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное
образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и
информатики»
(СибГУТИ)

Кафедра телекоммуникационных систем и вычислительных средств
(ТС и ВС)

РГЗ по теме: «Создание тестовой видеоплатформы с использованием
технологий Python для бэкенда, React JS для фронтенда и Figma для
дизайна»
по дисциплине
«*WEB-технологии*»

Студент:
Группа ИКС-433

Штейнбрехер С.В.

Преподаватель:
Старший преподаватель

А.В. Андреев

Новосибирск 2025

1 ВВЕДЕНИЕ

1.1 Цель проекта:

создать функциональную веб-платформу для потокового видео, где пользователи могут просматривать, загружать и управлять видеоконтентом. Проект должен продемонстрировать ваши навыки в разработке веб-приложений с использованием современного стека технологий и соблюдением принципов UI/UX-дизайна.

1.2 Технические требования:

1.2.1 Бэкенд на Python:

- Разработать API, использующее Django или Flask для обработки запросов от фронтенда.
- Реализовать функционал регистрации и авторизации пользователей с использованием JWTтокенов.
- Разработать возможности для загрузки и потокового воспроизведения видео.
- Создать базу данных для хранения информации о пользователях, видео и метаданных (используйте Postgres или MySQL).

1.2.2 Фронтенд на React JS:

- Интеграция с бэкендом для регистрации, входа и воспроизведения видео.
- Разработать компоненты для добавления, просмотра и управления видеоконтентом.
- Реализовать маршрутизацию (React Router) для навигации между различными страницами приложения.

1.2.3 Дизайн с использованием Figma:

- Воспользоваться предоставленным дизайном в Figma для реализации интерфейса.
- Проанализировать и понять основные элементы дизайна, включая цветовую палитру, типографику и расположение элементов на странице.
- Соблюдать принципы респонсивного дизайна для обеспечения корректного отображения на различных устройствах.

2 ПРОЦЕСС ВЫПОЛНЕНИЯ ПРОЕКТА

2.1 Документация API

2.1.1 Методы API

Метод	Эндпоинт	Описание	Параметры	Пример
POST	/register	Регистрация	username, email, password	<code>{"username": "u</code>
POST	/login	Авторизация	username, password	<code>{"username": "u</code>
GET	/logout	Выход	—	—
POST	/upload	Загрузка видео	video_file (FormData)	<code>curl -F "video_ file=@video.mp</code>
GET	/videos	Список видео	—	—
DELETE	/video/<id>	Удаление видео	id	—
POST	/chat/send	Отправить сообщение	message	<code>{"message": "Пр</code>
GET	/chat	Получить сообщения	—	—
DELETE	/chat/<id>	Удалить сообщение	id	—

2.2 Отчёт о разработке

2.2.1 Используемые технологии

- Backend: Flask (Python) + SQLite
- Frontend: HTML5/CSS3, JavaScript (Fetch API)
- Дополнительно: Werkzeug (аутентификация)

2.2.2 Трудности и решения

Проблема	Решение
Загрузка видео	Использование FormData и request.files в Flask
Аутентификация	Сессии Flask + хеширование паролей (werkzeug.security)
Динамический чат	AJAX (Fetch API) с периодическим опросом сервера
Удаление данных	Добавлена проверка прав (user_id == message.author_id)

2.2.3 Пример кода

```
@app.route('/upload', methods=['POST'])
def upload_video():
    if 'video_file' not in request.files:
        return "Ошибка: нет файла", 400
    file = request.files['video_file']
    if file.filename == '':
        return "Ошибка: не выбран файл", 400
    if file:
        filename = secure_filename(file.filename)
        file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
        return "Файл успешно загружен", 200
```

2.3 Архитектура проекта

```
/my_videoplatform
/app
  __init__.py
  routes.py
  database.py
/static
/uploads
```

```
/css
/templates
  account.html
  base.html
  index.html
  login.html
  register.html
  video.html
requirements.txt
main.py
Procfile
```

2.4 Заключение

Создана функциональная веб-платформа для потокового видео, где пользователи могут просматривать, загружать и управлять видеоконтентом.

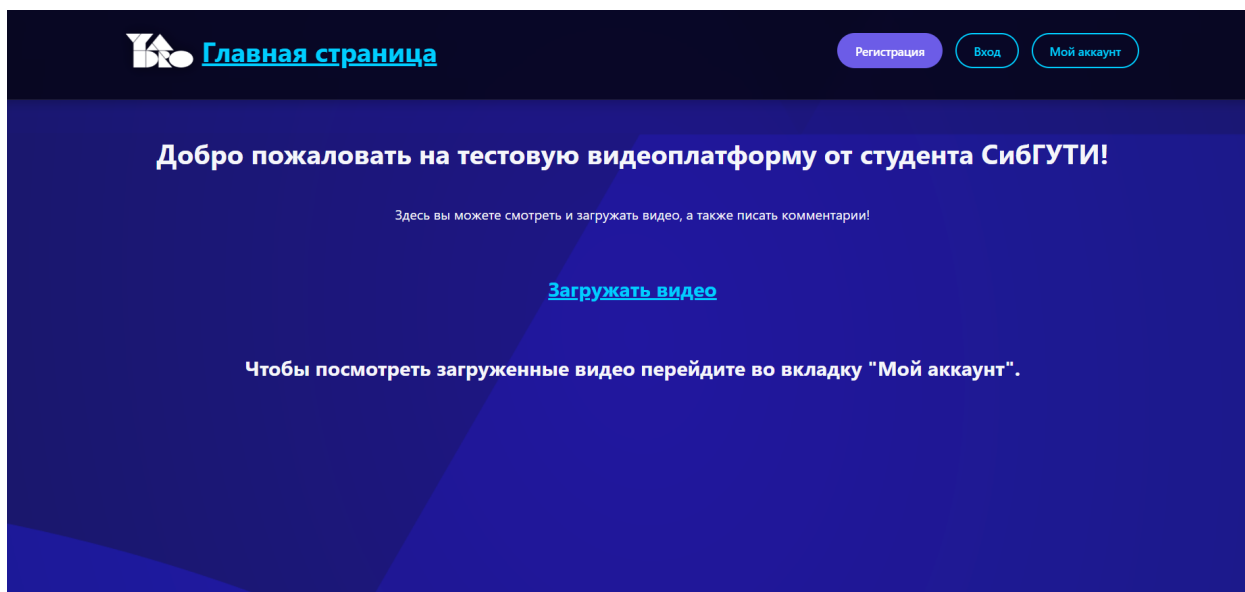


Рисунок 1 — Главный интерфейс приложения

Ссылка на работу в Github: https://github.com/Ssteynbreher/WEB_site