Итоговый проект

Самостоятельно разработать полноценное одностраничное приложение (SPA), используя обязательный стек технологий и покрыв его тестами.

Этапы выполнения проекта

Проект выполняется в два основных этапа:

Этап 1: Проектирование и защита идеи (срок: до 1 декабря)
Перед началом разработки вы должны подготовить и представить на утверждение краткую презентацию вашего будущего проекта. Это поможет заранее продумать архитектуру и получить обратную связь.

Представьте что вы основатель стартапа и вам предстоит убедить что ваша идея заслуживает внимания.

Презентация должна включать:

- Описание идеи: Какую проблему решает ваше приложение? Каков его основной функционал? Чем ваше приложение может быть лучше существующих?
- Пользовательские сценарии: Описание 2-3 ключевых сценариев использования. Например: "Пользователь заходит на сайт -> видит список товаров -> использует фильтр по категории -> переходит на страницу товара".
- Дизайн: Простые схематичные макеты для основных страниц (главная, страница элемента, корзина и т.д.). Их можно нарисовать от руки или использовать инструменты вроде Figma, Miro.
- Описание API для взаимодействия с бэкендом (REST API).

Формат сдачи этапа 1: Простой статический сайт на Github Pages (чистый HTML & CSS & JS или Reveal.js), презентация на занятии.

Этап 2: Разработка и сдача проекта (срок: до 20 декабря)

После утверждения идеи вы приступаете к разработке, следуя требованиям.

1. Основные требования к проекту

Ваш проект должен быть реализован с использованием следующего стека технологий:

1. **TypeScript:** Весь код приложения (логика, компоненты, стейт-менеджмент) должен быть написан на TypeScript
2. **React:** Пользовательский интерфейс должен быть построен на React. Используйте функциональные компоненты и хуки.
3. **Redux Toolkit:** Для управления состоянием приложения необходимо использовать Redux, настоятельно рекомендуется применять Redux Toolkit как современный стандарт для работы с Redux. Это включает в себя создание слайсов (slices), использование `createAsyncThunk` для асинхронных запросов и селекторов для получения данных в компонентах.
4. **React Router:** В приложении должна быть реализована клиентская маршрутизация с помощью библиотеки react-router-dom. Проект должен содержать как минимум 3-4 уникальные страницы (например, главная, страница элемента, форма создания/редактирования, страница "о проекте").

2. Технические требования

- Взаимодействие с API: Приложение должно получать данные из базы данных. В качестве сервера можно или реализовать бэкенд самостоятельно (в качестве базы данных можно выбрать sqlite) или использовать сервисы Firebase/Supabase (есть бесплатный тариф достаточный для реализации проекта). Необходимо реализовать как минимум операции чтения данных (GET), создания (POST), обновления (PUT/PATCH) и удаления (DELETE). Реализация API должна соответствовать REST API.

- Асинхронные операции: Все асинхронные запросы к API должны обрабатываться через Redux с использованием `createAsyncThunk`. Состояние запросов (загрузка, успех, ошибка) должно корректно обрабатываться и отражаться в интерфейсе (например, показом прелоадера или сообщения об ошибке).
- Стилизация: Вы можете использовать **CSS/SCSS Modules** для стилизации. Приложение должно быть адаптивным и корректно отображаться как на десктопных, так и на мобильных устройствах.
- Качество кода: Код должен быть хорошо структурирован, разделен на логические модули и компоненты. Понятные названия переменных и функций, а также следование принципам чистого кода обязательны. Наличие ESLint и Prettier для поддержания единого стиля кода будет плюсом.
- Деплой: Готовый проект должен быть развернут на любом бесплатном хостинге (например, **GitHub Pages**) и быть доступен по публичной ссылке.
- Для получения высокой оценки необходимо решить более сложную задачу. Например:
    - использовать GraphQL для взаимодействия с бэкендом
    - WebRTC для реализации совместной работы над данными (или чата, или синхронизации данных без единого бэкенд сервера)
    - интеграция с GPT или другими нейронками (локально на сервере с помощью ollama или через API сторонних сервисов) для суммаризации, категоризации или автоматизации задач приложения
    - Интеграция с чат ботом (Telegram)
    - Разработка и интеграция WebExtension (например для сохранения заметок из интернета, или анализа посещения сайтов, или реализация своего сервиса для url закладок)
    - ваша крутая идея

3. Тестирование

Важной частью задания является написание тестов для вашего приложения.

Необходимо реализовать все три следующих вида тестов:

1. Unit-тесты, проверить корректность работы изолированных частей логики, функций, редьюсеров, хуков:
    - Инструменты: **Jest, React Testing Library**.
    - Требование: Написать тесты для нескольких ключевых редьюсеров (проверка изменения стейта на разные экшены) и как минимум для 2-3 сложных функций вашего приложения.
2. Компонентные (скриншотные) тесты, проверить, что компоненты отображаются корректно и их внешний вид не изменился после внесения правок:
    - Инструменты: Storybook для визуализации компонентов в изоляции + аддон для скриншот-тестирования (например, storyshots или Loki).
    - Требование: Для 3-4 основных переиспользуемых компонентов (например, Button, Input, Card) создать "истории" в Storybook. Для этих же компонентов настроить скриншот-тестирование, которое будет создавать и сравнивать снимки их внешнего вида.
3. End-to-end (E2E) тесты, проверить ключевые пользовательские сценарии от начала до конца, имитируя действия реального пользователя в браузере:
    - Инструменты: **Cypress**, **Playwright**.
    - Требование: Написать 1-2 E2E-теста для самых важных сценариев вашего приложения (например: "пользователь заходит на главную, использует поиск, переходит на страницу товара и видит его описание" или "пользователь добавляет товар в корзину и проверяет, что он там появился").

4. Формат сдачи

В качестве результата необходимо предоставить:

1. Ссылку на развернутое (задеплоенное) приложение.
2. Ссылку на публичный репозиторий на GitHub с исходным кодом проекта.
3. README.md файл в корне репозитория, в котором должны быть описаны:

- Краткое описание проекта.
- Инструкция по запуску проекта локально (npm install, npm start и т.д.).
- Инструкция по запуску тестов (npm test, npm run storybook, npm run cypress:open и т.д.).
- Краткое описание архитектуры и принятых решений (например, почему выбрали ту или иную библиотеку, структура проекта).

5. Критерии оценки

- Корректное использование TypeScript: типизация пропсов, состояния, асинхронных экшенов и т.д.
- Архитектура React-компонентов: грамотное разделение на контейнерные и презентационные компоненты, переиспользование.
- Управление состоянием: правильная настройка Redux Toolkit, работа с асинхронными операциями, использование селекторов.
- Наличие и качество тестов (unit, скриншотные, E2E): тесты должны быть осмысленными и проверять ключевой функционал.
- Маршрутизация: корректная работа навигации между страницами.
- UI/UX и адаптивность: приложением удобно пользоваться, оно адекватно выглядит на разных разрешениях экрана.
- Качество кода и репозитория: структура проекта, читаемость кода, наличие полного README.md.
- Работоспособность приложения: отсутствие критических ошибок, работоспособность основного функционала, корректная работа API.
- Реализация сложной функциональности

Идеи для проектов

Если у вас нет своей идеи, вы можете выбрать одну из предложенных ниже. Вы можете упрощать или усложнять предложенные проекты на свое усмотрение.

1. Каталог фильмов
    - Главная страница со списком популярных фильмов.

- Страница с детальной информацией о фильме (постер, описание, рейтинг, список актеров).
- Поиск фильмов по названию.
- Добавление фильмов в "Избранное" или "Хочу посмотреть" (состояние хранится в Redux и, возможно, в localStorage).

2. Блог-платформа
    - Список всех постов.
    - Страница для просмотра одного поста и его комментариев.
    - Форма для создания нового поста (данные можно отправлять на API, но они не сохранятся; главное — продемонстрировать работу с POST-запросом).
    - Возможность удаления поста (DELETE-запрос).

3. Книга рецептов
    - Поиск рецептов по названию или ингредиенту.
    - Страница с детальным описанием рецепта (ингредиенты, инструкция по приготовлению, фото).
    - Главная страница со случайными рецептами или списком категорий блюд.
    - Сохранение любимых рецептов в "Избранное".

4. Каталог видеоигр
    - Главная страница со списком популярных или новых игр.
    - Поиск игр по названию.
    - Фильтрация игр по платформе (PC, PlayStation, Xbox) и жанру (Action, RPG, Strategy).
    - Страница с детальной информацией об игре: скриншоты, описание, рейтинг, доступные платформы, дата релиза.

- Создание персонального списка "Избранное" или "Хочу сыграть".
- Сортировка игр по рейтингу или дате выхода.

5. Книжный поисковик
    - Поле для поиска книг по названию, автору или ISBN.
    - Страница с результатами поиска, отображаемыми в виде карточек (обложка, название, автор).
    - Возможность загружать больше результатов (пагинация или "бесконечная прокрутка").
    - Страница с подробной информацией о книге: обложка, описание, количество страниц, издатель, категории.
    - Сортировка найденных книг (по релевантности, по дате публикации).
    - Создание "Списка для чтения", который хранится в Redux и localStorage.

6. Персональный планировщик задач по Scrum

7. Распределенный чат

8. Платформа для совместного редактирования заметок с историей

9. Приложение для управления сервером или стационарным компьютером через веб интерфейс (управление аудио или видео плеером, возможность добавления команд через веб интерфейс)

10 Магазин (корзина товаров)

11. Социальная сеть (создания страницы профиля, чат с другими пользователями, подписка на других)

12. Новостной сайт (лента новостей)

13. Рекламная сеть (объявления)