### Введение

Целью данного Технического задания является описание требований к разработке веб-сайта, посвященного обучению интернет-грамотности детей. Веб-сайт будет разработан на основе предоставленного макета и должен быть адаптирован для просмотра на устройствах с различными экранами.

### Описание проекта

Веб-сайт должен быть построен на базе фреймворка React и использовать для верстки основным инструментом Flexbox. Для работы с состоянием приложения должен использоваться State Manager Redux Toolkit.

На веб-сайте должны быть представлены 8 тем, посвященных интернет-грамотности детей:

1. Персональные данные
2. Распространенные угрозы
3. Цифровой след
4. Сетевая безопастность
5. Антивирусы
6. Общение в сети
7. Кибербуллинг
8. Фейки

Каждая тема должна содержать теоретический материал и несколько небольших тестов для закрепления знаний.

Для реализации функциональности веб-сайта должны использоваться следующие технологии:

* HTML/CSS для верстки интерфейса;
* JavaScript (без TypeScript) для реализации клиентской логики;
* Django для написания API, взаимодействующего с базой данных;
* PostgreSQL в качестве базы данных.

### Функциональные требования

1. Разработка и внедрение дизайна веб-сайта, соответствующего макету;
2. Создание страниц, содержащих материалы и тесты по каждой из 8 тем;
3. Создание API для обработки запросов с фронтенда;
4. Реализация функции регистрации и авторизации пользователей;
5. Реализация функции сохранения прогресса прохождения тестов для каждого пользователя;
6. Разработка интерфейса для просмотра пройденных тестов и прогресса для каждого пользователя;
7. Адаптация веб-сайта для просмотра на устройствах с различными экранами.

### Нефункциональные требования

Безопасность: веб-сайт должен быть защищен от взлома и несанкционированного доступа к данным пользователей. Для этого необходимо использовать проверенные инструменты для защиты от XSS, CSRF и SQL-инъекций. Также необходимо реализовать защиту паролей пользователей, в том числе хэширование и соль.

* Скорость: веб-сайт должен загружаться быстро и отвечать на запросы пользователей в течение нескольких секунд. Для этого необходимо оптимизировать код и изображения, использовать кэширование и минимизировать количество запросов к серверу.
* Масштабируемость: веб-сайт должен быть масштабируемым и готовым к увеличению числа пользователей и объема данных. Для этого необходимо использовать горизонтальное масштабирование и архитектуру, которая позволит легко добавлять новые функции и компоненты.
* Доступность: веб-сайт должен быть доступен для пользователей с ограниченными возможностями, такими как слабовидящие и слабослышащие. Для этого необходимо следовать стандартам доступности и использовать соответствующие технологии, такие как ARIA и Web Content Accessibility Guidelines.
* Надежность: веб-сайт должен быть надежным и работать без сбоев и ошибок. Для этого необходимо тестировать код и использовать мониторинг и логирование ошибок.
* Поддерживаемость: веб-сайт должен быть легко поддерживаемым и обновляемым. Для этого необходимо использовать стандарты кодирования и документировать код, чтобы другие разработчики могли быстро разобраться в проекте. Также необходимо использовать инструменты для автоматического тестирования и развертывания, такие как Continuous Integration и Continuous Deployment.

5 Требования к архитектуре

5.1. Веб-приложение должно быть построено с использованием модульной архитектуры.

5.2. Бекенд и фронтенд должны быть разделены и связаны между собой через REST API.

5.3. Бекенд-сервер должен быть реализован с использованием Django и базы данных PostgreSQL.

5.4. Фронтенд-приложение должно быть построено на React и Redux Toolkit.

5.5. Для стилизации пользовательского интерфейса следует использовать CSS-препроцессор SASS.

1. Требования к тестированию

6.1. Веб-приложение должно иметь модульные, интеграционные и функциональные тесты.

6.2. Для написания и запуска тестов следует использовать библиотеку Jest для фронтенд-кода и Pytest для бекенд-кода.

6.3. Тесты должны запускаться автоматически в ходе сборки и деплоя приложения.

1. Требования к документации

7.1. Должна быть предоставлена документация по API бекенд-сервера в формате Swagger.

7.2. Документация по фронтенд-приложению должна содержать инструкции по запуску, сборке и развертыванию.

7.3. Документация по базе данных должна содержать описание схемы и связей между таблицами.

1. Требования к безопасности

8.1. Все запросы к серверу должны осуществляться через HTTPS.

8.2. Приложение должно обеспечивать защиту от SQL-инъекций и других атак на базу данных.

8.3. Все пароли пользователей должны храниться в базе данных в зашифрованном виде.

8.4. Приложение должно иметь систему аутентификации и авторизации пользователей.

8.5. Сессии пользователей должны быть корректно управляемыми и иметь ограниченное время действия.

1. Требования к развертыванию и масштабированию

9.1. Приложение должно корректно работать на платформах Linux, MacOS и Windows.

9.2. Приложение должно иметь инструкции по развертыванию в локальной среде разработки и на удаленном сервере.

9.3. Приложение должно иметь возможность масштабироваться горизонтально в зависимости от нагрузки.

9.4. Для автоматического развертывания и масштабирования

6. Требования к безопасности

- Сайт должен быть защищен от взлома и несанкционированного доступа к данным пользователей.

- Необходимо реализовать механизмы защиты от CSRF-атак и XSS-атак.

- Данные пользователей должны храниться в зашифрованном виде.

- Сайт должен соответствовать требованиям законодательства о защите персональных данных.

7. Требования к тестированию и отладке

- Перед запуском сайта необходимо провести полное тестирование на различных устройствах и браузерах.

- Необходимо проводить регулярное тестирование сайта на наличие ошибок и уязвимостей.

- На сайте должны быть реализованы механизмы отладки, для удобства поиска и исправления ошибок.

8. Требования к документации

- Необходимо создать документацию на проект, описывающую все этапы разработки и подробно описывающую функционал сайта.

- Документация должна содержать инструкцию по развертыванию и настройке сайта.

- В документации должен быть указан список используемых технологий и сторонних библиотек.

9. Требования к сопровождению

- После запуска сайта, необходимо обеспечить его бесперебойную работу и производить регулярное обновление и поддержку.

- В случае обнаружения ошибок или уязвимостей, необходимо незамедлительно принимать меры по их устранению.

- Предусмотреть возможность расширения функционала сайта в будущем.

10. Требования к команде разработчиков

- Команда должна состоять из высококвалифицированных специалистов, владеющих современными технологиями разработки веб-сайтов.

- Команда должна обладать опытом разработки проектов аналогичной сложности.

- Разработчики должны владеть английским языком на уровне, достаточном для чтения документации и общения с коллегами.

4. Технические требования

4.1. Frontend

4.1.1. Верстка и стилизация

- Сайт должен быть построен на основе React

- Для верстки используется HTML5 и CSS3

- Для стилизации используется препроцессор SCSS

- Использование CSS-фреймворков и библиотек не допускается

- Для адаптивной верстки используется Flexbox и Grid

- Сайт должен корректно отображаться на различных устройствах и разрешениях экрана

- Верстка должна быть семантической и доступной

4.1.2. Функциональность

- Сайт должен быть реализован в виде одностраничного приложения (SPA)

- Навигация по сайту должна осуществляться без перезагрузки страницы (использование React Router)

- Для отображения динамических данных используется Redux Toolkit

- Все пользовательские действия должны выполняться без перезагрузки страницы (использование AJAX)

- Сайт должен быть совместим с современными браузерами и иметь плавную анимацию

- Сайт должен обладать высокой скоростью загрузки (использование ленивой загрузки и компрессии изображений)

4.2. Backend

4.2.1. Язык программирования и фреймворк

- Backend должен быть построен на основе Django

- Для написания кода используется язык Python 3

- Использование Django REST Framework для создания API

- Использование Celery для асинхронной обработки задач

4.2.2. База данных

- Использование PostgreSQL в качестве базы данных

- Создание необходимых моделей и миграций для базы данных

- Реализация связей между моделями для удобного доступа к данным

4.2.3. Функциональность

- Создание API для получения и обновления данных

- Реализация аутентификации и авторизации пользователей (использование JWT токенов)

- Реализация логирования запросов и ошибок на сервере

- Реализация защиты от возможных угроз безопасности (например, SQL-инъекций)

4.3. Дизайн

- Дизайн сайта должен быть современным, красивым и интуитивно понятным

- Использование палитры цветов, соответствующей тематике проекта

- Использование соответствующих шрифтов для удобного чтения и привлек