Техническое задание на разработку сайта «Идеальный день»

# Аннотация

Данное техническое задание описывает функциональные и технические требования к разработке проекта «Идеальный день» — веб-приложения для создания, редактирования и анализа персонального распорядка дня пользователей.

# 1. Цели создания сайта

Создать удобный инструмент, с помощью которого пользователи могут составлять и редактировать идеальный распорядок дня, анализировать настроение и активности, а также сравнивать дни с другими пользователями по критерию похожести.

# 2. Целевая аудитория

Школьники и студенты, интересующиеся планированием, самонаблюдением и саморазвитием. Пользователи хотят отслеживать и оптимизировать свой день.

# 3. Требования к сайту

## 3.1. Общие требования

- Название проекта: Идеальный день  
- Язык интерфейса: Русский  
- Бэкенд: Python  
- REST API: Реализуется вручную  
- ORM: SQLAlchemy  
- Фронтенд: HTML   
- Аутентификация: через email и пароль (хранение пароля в хэше)

## 3.2. Ролевая модель

|  |  |  |
| --- | --- | --- |
| Роль | Возможности и полномочия | Кем создается |
| Незарегистрированный | Просмотр главной страницы, регистрация, вход в аккаунт | — |
| Зарегистрированный | Создание/редактирование дня, просмотр графика, анализ | Пользователь |
| Администратор (опц.) | Управление пользователями (через базу данных или консоль) | Разработчиком вручную |

## 3.3. Функциональные требования к публичной части

- Главная страница: краткое описание проекта и ссылки на регистрацию/вход  
- Страница регистрации: форма с полями (имя, email, пароль)  
- Страница входа: форма (email, пароль)

## 3.4. Функциональные требования к личному кабинету клиента

- Просмотр текущего дня  
- Добавление события (время, тип, настроение, музыка, описание)  
- Редактирование и удаление событий  
- Просмотр предыдущих дней

## 3.5. Функциональные требования к панели управления

Отдельной панели не предусмотрено. Администрирование выполняется вручную через базу данных или консоль разработчика.

## 3.6. Требования к структуре сайта

### 3.6.1. Структура публичной части сайта

|  |  |  |  |
| --- | --- | --- | --- |
| Раздел / подраздел | Содержимое | Функциональные требования | Требования к контенту |
| Главная | Описание сервиса, кнопки Вход/Регистрация | Переход к формам | Краткий текст о проекте |
| Регистрация | Форма | Проверка email, хэш пароля | — |
| Вход | Форма | Аутентификация | — |

## 3.7. Архитектура сайта

Сайт реализован по архитектуре клиент-сервер. Фронтенд основан на HTML, генерируемом сервером с использованием Python. Бэкенд написан на чистом Python и библиотеке SQLAlchemy для работы с БД

## 3.9. Технические требования

- Язык разработки: Python  
- Работа с БД: SQLAlchemy  
- API: REST  
- СУБД: SQLite  
- Вёрстка: HTML, CSS   
- Система контроля версий: Git  
- Операционные системы: Windows

## 3.10. Требования к дизайну

- Чистый и минималистичный интерфейс  
- Использование базовых HTML-элементов и CSS  
- Поддержка светлой темы

## 3.11. Требования к контенту

1. Контент создается разработчиком.
2. Тексты и данные тестовых пользователей также создаются вручную.

## 3.12. Требования к внутренней SEO оптимизации

- ЧПУ (человекопонятные URL)  
- Семантически правильная HTML-структура

## 3.14. Дополнительные требования

### 3.14.1. Управление изменениями

- Использование Git для отслеживания изменений

# 4. Требования к документации

- Документация по API  
- Описание структуры базы данных  
- Инструкция по запуску проекта

# 5. Стадии и этапы разработки

## 5.1. Прототипирование

- Создание прототипа на бумаге или в графическом редакторе  
- Проработка навигации  
- Минимальное тестирование логики

## 5.2. Создание дизайна

Минималистичный HTML+CSS макет без интерактива. Использование Bootstrap

## 5.3. Вёрстка и разработка

1. Реализация логики на Python, SQLAlchemy
2. Библиотеки: flask, flask-login, os, datetime, werkzeug.utils

## 5.4. Тестирование

- Ручное тестирование всех функций  
- Минимальное тестирование ключевых функций API

## 5.5. Документирование

Создание документации API.