**Сайт**

Национальный исследовательский университет   
«Высшая школа экономики»

Лицей

Индивидуальная выпускная работа

Отчёт о проекте

**Система автоматизированного тестирования**

*Выполнил Мороз Александр Константинович*

Консультант:

Купцов Александр Александрович

Москва 2024

*Название проекта*: [Система Автоматизированного Тестирования](http://185.42.14.235:8000)

*Проблемное поле*: Продукт поможет устранить ряд недостатков других платформ. Проблемы и недоработки существующих систем и пути их решения в проекте:

1. Ограниченные возможности существующих систем по автоматизации оценки и анализа результатов

Ручной труд: Обработка результатов тестирования вручную занимает длительный промежуток времени, что, как показывает практика, дополнительно увеличивает вероятность ошибок, особенно в условиях большой выборки данных. Разработанное решение позволяет оценить знания тестируемого как с применением автоматизированной системы (средствами самого сервиса), так и с помощью интеграции через API с внешними сервисами.

Сложность анализа: Существующие системы онлайн тестирования не всегда могут предоставить удобные инструменты для анализа результатов, такие как: графики, отчеты, сравнительные таблицы. Отсутствие инструментов затрудняет выявление «пробелов» в знаниях тестируемых, что, в свою очередь, усложняет объективную оценку их уровня подготовки. В текущей версии системы, функционал подробного анализа в представленную систему не интегрирован, вместе с тем пользователю предоставлена возможность задействовать API продукта для решения данной задачи сторонними сервисами.

Ограниченные типы оценивания: Многие платформы предлагают только базовые форматы тестирования (например: выбор ответа), что не позволяет оценивать сложные навыки и знания. В реализуемом проекте, пользователю предоставлена возможность дать развернутый ответ, который будет оценен вручную через средства визуализации сервиса или за счет интеграции через API с внешним сервисом.

2. Отсутствие гибкости в настройке тестов и интеграции с другими образовательными инструментами

Невозможность кастомизации: Жесткие шаблоны тестов создают проблемы, не позволяя адаптировать задания под конкретные учебные цели и особенности тестируемых. В представленной системе появилась возможность добавить развернутый ответ на вопрос, без возможности прикрепления к ответу дополнительных материалов в виде файлов различного формата и без возможности использования специальной верстки для составления химических, физических, математических и иных видов формул.

Проблемы интеграции: Существующие системы не всегда поддерживают интеграцию с другими образовательными инструментами (например, LMS, платформами для онлайн-обучения, системами управления контентом), что ограничивает возможность использования ресурсов, лишает пользователя перспективы эффективно использовать нужный ему сервис. Вместе с тем предоставленный API открывает возможности для интеграции с любыми другими платформами, создавая мощные и гибкие решения, которые значительно расширяют функциональность и улучшают взаимодействие между системами.

Неудовлетворительные API: Для разработчиков, стремящихся создать собственные инструменты или расширения, часто отсутствуют качественные API, что усложняет процесс интеграции. В настоящий момент при реализации проекта был разработан базовый API создания и оценивания тестов, регистрации пользователей, сбора статистики. Настоящий набор методов обладает значительной гибкостью и может быть расширен как в ширину, так и в глубину, что предоставляет неограниченные возможности для дальнейшей адаптации и совершенствования.

3. Нехватка функционала для получения детализированной статистики по результатам тестирования

Отсутствие глубокого анализа: Многие системы не предоставляют детализированные отчеты по результатам тестирования: анализ по категориям знаний, времени выполнения заданий и т.д. Создатель теста не сможет понять, какой материал участники тестирования усвоили лучше и на чем стоит акцентировать внимание. Реализованный API позволяет сервисам обращаться к результатам и анализировать их, что в свою очередь решает проблему детализации отчетов.

Недостаточная визуализация данных: У разработчиков тестов отсутствует доступ к визуализированным данным, что затрудняет интерпретацию результатов и принятие решений на основе этих данных. Визуализация данных может быть достигнута за счет получения данных через API и с дальнейшим использованием сторонних систем.

Необходимость улучшения механизмов мониторинга прогресса: Без возможности отслеживания динамики успеваемости тестируемых на протяжении времени создатели тестов не могут корректировать свои методы подачи материала и адаптировать процесс обучения. Чтобы провести всестороннюю оценку динамики тестируемого, целесообразно задействовать сторонние сервисы, использующие API проекта, что позволит получить более полное представление о данных.

Вывод: Существующие образовательные платформы не удовлетворяют потребностям пользователей в области оценки знаний из-за ограниченного функционала, недостаточной гибкости и отсутствия интеграции с другими инструментами. Это препятствует эффективному обучению и оценке, что подчеркивает необходимость разработки более интегрированных и адаптивных решений в области образовательных технологий. В ходе разработки проекта был создан API позволяющий провести интеграцию со сторонними сервисами, получить все недостающие аналитические данные, была создана система оценивания тестов, регистрации пользователей и сбора статистики.

Носитель проблемы: В современных условиях, характеризующихся динамичным развитием и возрастающими требованиями, пользователь ограничен функционалом существующих систем, что не позволяет ему в полной мере реализовать свой потенциал. Это приводит к снижению эффективности работы и затрудняет адаптацию к новым вызовам, что в свою очередь сказывается на общей производительности и инновационности.

Выбор проблемы: Данная проблема чрезвычайно актуальна: Преподаватели дают тесты ученикам для выполнения на онлайн платформах таких как Online Test Pad и т.д. Сервис Online Test Pad имеет ряд значительных недостатков: нажатие клавиши Enter может завершить тестирование, отсутствует учёт человеческого фактора (т.е. небольшая опечатка, например: лишний пробел, точка или запятая. Тест сразу же считает такую ошибку неправильным ответом, и педагог не может вручную добавить баллы), проблема отсутствия статистики (педагог не может понять, какой средний балл у учеников за тест, если не произведет эти вычисления вручную). Идея проекта родилась в результате анализа вышеперечисленных недостатков уже существующих сервисов.

*История работы над проектом*: Тема данного проекта была четко определена на основании актуальных потребностей в области образования, а также тщательного анализа существующих решений и сервисов, что позволило выявить ключевые направления для дальнейшего развития.

Этап 1: Поэтапное планирование

1)Постановка целей (1-3 дня)

2)Разработка структуры базы данных (5-8 дней)

3) Разработка документации для API (7-10 дней)

4)Разработка web-сервиса (2-3 месяца)

5)Разработка web-интерфейса (1-1,5 месяц)

6)Развёртывание web-сервиса на виртуальном сервере (1-2 дня)

Этап 2: Исследование и анализ

На начальном этапе был проведен глубокий анализ существующих систем автоматизированного тестирования, выявлены их преимущества и недостатки. Особое внимание было уделено функционалу, гибкости настройки, интеграции с другими платформами и аналитическим возможностям. Был проведен опрос среди лицеистов ([результаты](https://docs.google.com/spreadsheets/d/1Ff0Ay7-MBfTftIWehmD8IEUCccJoTxf4bmyD7hQn9xM/edit?resourcekey=&gid=30682252#gid=30682252)), благодаря которому удалось более точно определить актуальность проекта. Дополнительно было проведено качественное исследование, участниками которого стали репетиторы по математике, по информатике, по русскому языку и по английскому языку ([результаты](https://docs.google.com/spreadsheets/d/1kQNZg3Y4dk7QcsCkrbg3BM93uKvr_CwOjQXd6CeWWJc/edit?usp=sharing)).

Этап 3: Разработка структуры базы данных

1)Выбор технологий для реализации web-сервиса

В данном разделе представлено обоснование выбора технологий и инструментов, применяемых для разработки веб-сервиса на платформе FastAPI. Данный сервис будет предоставлять пользователям возможность регистрации, создания и прохождения тестов, что значительно повысит их взаимодействие с системой.

В условиях ограниченного времени не представилось возможным уделить равное внимание как разработке бэкенда, так и фронтенда. В связи с этим, основной акцент был сделан на разработку бэкенда, что позволило обеспечить стабильную функциональность и производительность сервиса.

1.1. Бэкенд

1) FastAPI: Выбранный фреймворк для создания API благодаря высокой производительности, поддержке асинхронного программирования и автоматической генерации документации (Swagger), а также быстрой скорости разработки и доступности документации на сам фреймворк. Django был признан тяжеловесным и неподходящим для быстрого старта. Flusk требует огромного количества библиотек для превращения в некое подобие Fastapi. FastAPI уже готовый вариант, который позволяет легко начать разработку. Основной идея проекта была предоставить API, поэтому FastAPI подходил нам наилучшим образом.

2)SQLite: Легковесная реляционная база данных, подходящая для хранения данных о пользователях и тестах. Не требует установки отдельного сервера, что упрощает развертывание и тестирование продукта в условиях, приближенных к реальным. На начальном этапе не ожидается большого потока пользователей, тестировщиками выступит ограниченный круг лиц, поэтому функционала SQLite будет достаточно для проведения тестирования. В дальнейшем планируется заменить SQLite на PostgreSQL как более производительную СУБД.

1.2. Фронтенд

Разработка собственного web-интерфейса была вынужденной мерой и преследовала цель продемонстрировать возможности предоставленного API, а также познакомиться с разработкой web-интерфейсов для понимания того, как создается фронтенд.

1)JavaScript: Основной язык для web-разработки, обеспечивающий интерактивность интерфейса. JavaScript – это язык с одним из самых низких порогов кода и для старта во фронтенде он подходит наилучшим образом. В идеале если проект будет дальше разрабатываться, то JavaScript будет заменен на TypeScript в комплекте с одним из современных фронтенд фреймворков на подобии react, angular, vue или их аналогов.

2)CSS: Простой и мощный язык стилей, обеспечивающий гибкость, кроссбраузерную совместимость и возможность быстрой интеграции с фреймворками для создания привлекательных web-интерфейсов. Было принято решение в начале использования CSS препроцессоров оценить возможности самого CSS и сложность его применения к подобным проектам.

1.3. Структура Базы Данных (PK – primary\_key; FK – ForeignKey)

1) Структура сессии пользователя при прохождении теста

"user\_quiz\_session"

Id-сессии, Integer, PK

Id-пользователя, Integer, FK(users.id)

Id-теста, Integer, FK(quizes.quiz\_id)

Id-вопросов в сессии, String

Время начала сессии, DateTime

Время окончания сессии, DateTime, nullable

Время затраченное на сессию, Integer

Время обновления сессии, DateTime, nullable

Результат сессии, Integer, nullable

Дополнительная попытка, Integer, FK (user\_extra\_try.id), nullable

Номер вопроса в сессии, Integer

Кол-во пройденных вопросов, Integer, default=0

Представленная таблица вызывает особый интерес, поскольку именно из-за нее база данных теряет свою нормальную структуру, что приводит к нарушению атомарности. Это происходит вследствие наличия поля, содержащего идентификаторы вопросов в сессии, которое хранит сразу несколько значений для идентификаторов вопросов. Указанные идентификаторы представлены в виде списка и конвертированы в строку формата JSON.

Дополнительным недостатком данного подхода является нарушение целостности данных: при удалении вопроса его идентификатор продолжает оставаться в строке идентификаторов вопросов в сессии, что означает, что база данных не осуществляет контроль за целостностью данных в данной колонке.

Среди преимуществ данного решения можно выделить увеличение производительности работы базы данных. При большом объеме вопросов и значительном количестве сессий выполнение выборки вопросов может оказаться весьма затратным как по времени, так и по вычислительным ресурсам. В связи с этим, при проектировании базы данных были приняты следующие упрощения: после завершения формирования теста автор крайне редко удаляет вопросы из своего теста, что делает проблему целостности данных менее актуальной для данного проекта.

Таким образом, важным преимуществом становится возможность определения следующего вопроса для отображения пользователю за один запрос к базе данных.

2)Структура пользователя

"users"

Id-пользователя, Integer, PK

Электронная почта, String, unique

Пароль, String

3)Структура создателя

"quiz\_creators"

Id-создателя, Integer, PK

Id-пользователя, Integer, FK (users.id)

4)Структура теста

"quizes"

Id-теста, Integer, PK

Дата создания, String

Id-создателя, Integer, FK(quiz\_creators.id)

Время на тест, Integer, nullable

Количество вопросов, Integer, default=0

Количество вопросов для завершения, Integer

Имя теста, String

Описание теста, String, nullable

Количество попыток на тест, Integer, nullable

Показывать результаты, Boolean, default=False

Тип оценки теста, Integer, тип оценки теста in (1, 2, 3, 4), nullable=True

Возможность переключения между вопросами, Boolean, default=False

Количество простых вопросов, Integer, nullable, default=None

Количество средних вопросов, Integer, nullable, default=None

Количество сложных вопросов, Integer, nullable, default=None

Возможность изменения ответа, Boolean, default=False

5)Структура вопроса теста

"quiz\_questions"

Id-вопроса, Integer, PK

Текст вопроса, String

Описание вопроса, String, nullable

Время на вопрос, Integer, nullable

Баллы за вопрос, Integer, nullable

Тип вопроса, Integer, тип вопроса in (1, 2, 3)

Id-теста, Integer, FK (quizes.quiz\_id)

Номер вопроса, Integer, nullable

Сложность вопроса, Integer, сложность вопроса in (1, 2, 3), default=1

Подсказка на вопрос, String, nullable

6)Структура ответов на вопрос теста

"question\_answers"

Id-ответа, Integer, PK

Текст ответа, String

Id-вопроса, Integer, FK (quiz\_questions.id)

Количество баллов за ответ, Integer, nullable

Номер ответа, Integer, nullable

Правильный ответ, Boolean

7)Структура ответа на вопрос с выбором ответа

"user\_choice\_answers"

Id-ответа, Integer, PK

Id-сессии, Integer, FK (user\_quiz\_session.id)

Id-вопроса, Integer, FK (quiz\_questions.id)

Id-ответа, Integer, FK (question\_answers.id)

Время ответа, DateTime

Баллы за ответ, Integer, default=0

Правильный ответ, Boolean

8)Структура ответа на вопрос со свободным ответом

"user\_text\_answer"

Id-ответа, Integer, PK

Id-сессии, Integer, FK (user\_quiz\_session.id)

Id-вопроса, Integer, FK (quiz\_questions.id)

Id-ответа, Integer, FK (question\_answers.id)

Время ответа, DateTime

Баллы за ответ, Integer, default=0

Ответ, String

Правильный ответ, Boolean, nullable

9)Структура запроса на прохождение теста сверх лимита попыток

"user\_extra\_try"

Id-запроса, Integer, PK

Id-пользователя, Integer, FK (users.id)

Id-теста, Integer, FK (quizes.quiz\_id)

Время создания, DateTime

Одобрение, Boolean, default=None, nullable

Количество попыток, Integer, default=1

На данном этапе разработки проекта эта структура не используется, но планируется её дальнейшее применение при модернизации проекта.

10)Структура проверки ответа на вопрос в сессии

'user\_quiz\_questions'

Id-ответа на вопрос в сессии, Integer, PK

Id-сессии, Integer, FK (users.id)

Id-вопроса, Integer, FK (quiz\_questions.id)

Ответ есть, Boolean, default=False

11)Структура переподключения сессии

'session\_reconnect'

Id-переподключения, Integer, PK

Id-сессии, Integer, FK (user\_quiz\_session.id)

Id-вопроса, Integer, FK (quiz\_questions.id), nullable

Продолжительность теста, Integer

Время переподключения, DateTime

На текущем этапе разработки данного проекта указанная структура не находит применения; однако в будущем предполагается её использование в рамках предстоящей модернизации проекта.

Этап 4: Разработка схемы обмена данными клиент-сервер

1) AddUserRequest

Электронная почта, EmailStr

Пароль, str

2) LogInUserRequest

Электронная почта, EmailStr

Пароль, str

3) UpdateUserRequest

Электронная почта, EmailStr

Новый пароль, str

Старый пароль, str

4) AddQuizRequest

Имя теста, str

Длительность теста, int or None

Id пользователя, int or None

Id создателя, int or None

Количество вопросов для завершения, int

Описание теста, str or None

Количество попыток на тест, int or None

Показывать результаты после прохождения, bool

Тип оценки теста, int

Переключение между вопросами, bool

Возможность повторного ответа на вопрос, bool

Количество простых вопросов, int or None

Количество средних вопросов, int or None

Количество сложных вопросов, int or None

5) UpdateQuizRequest

Имя теста, str

Длительность теста, int or None

Количество вопросов для завершения, int

Описание теста, str or None

Количество попыток на тест, int or None

Показывать результаты после прохождения, bool

Тип оценки теста, int

Переключение между вопросами, bool

Возможность повторного ответа на вопрос, bool

Количество простых вопросов, int or None

Количество средних вопросов, int or None

Количество сложных вопросов, int or None

6) AddQuizQuestion

Текст вопроса, str

Описание вопроса, str or None

Время на вопрос, int or None

Баллы за вопрос, int or None

Тип вопроса, int

Id теста, int

Номер вопроса, int or None

Сложность вопроса, int or None

Подсказка на вопрос, str or None

7) UpdateQuizQuestion

Текст вопроса, str

Описание вопроса, str or None

Время на вопрос, int or None

Баллы за вопрос, int or None

Номер вопроса, int or None

Сложность вопроса, int or None

Подсказка на вопрос, str or None

8) AddQuestionAnswer

Текст ответа, str

Id вопроса, int

Баллы за ответ, int or None

Номер ответа, int or None

Правильный ответ, bool

9) UserAnswer

Список ответов, List[int]

Текстовый ответ, str or None

10) Points

Баллы, int

11) UserAnswerChoice

Id сессии, int

Id вопроса, int

Список ответов, List[int]

12) UserAnswerText

Id сессии, int

Id вопроса, int

Текстовый ответ, str

13) UserUpdateAnswerText

Текстовый ответ, str

14) UserUpdateAnswerChoice

Список ответов, List[int]

15) RefreshSession

Id сессии, int

Время отключения, int or None

Этап 5: Функции обработки запросов

Были разработаны следующие функции обработки запросов:

1)Пользователь

1) Добавление пользователя

2) Вход пользователя в систему

3) Получение всех пользователей

4) Получение пользователя по id

5) Получение пользователя по email

6) Изменение пользователя

2)Тест

1) Добавление теста

2) Удаление теста

3) Изменение теста

4) Получение списка всех тестов

5) Получение теста по id

6) Получение списка вопросов на тест

7)Получение списка тестов по id пользователя

3)Вопрос к тесту

1) Добавление вопроса

2) Удаление вопроса

3) Изменение вопроса

4) Получение вопроса по id

5) Получение списка вариантов ответа на вопрос

4)Ответ на вопрос к тесту

1) Добавление варианта ответа на вопрос

2) Удаление варианта ответа на вопрос

3) Изменение варианта ответа на вопрос

4) Получение варианта ответа по id

5)Ответ пользователя на вопрос к тесту

1) Добавление выбранного варианта ответа на вопрос

2) Изменение выбора ответа

3) Удаление выбора ответа

4) Добавление текстового ответа

5) Изменение текстового ответа

6) Удаление текстового ответа

6)Сессия теста

1) Начало сессии

2) Получение следующего вопроса сессии

3) Окончание сессии

4) Сохранение ответов

5) Получение результатов пользователя

6) Получение вопросов в сессии для отображения результатов

7) Получение вопросов в сессии для отображения статистики

8) Добавление баллов

9) Получение сессии по id

10) Получение средних результатов и количества попыток на тест

11) Получение всех сессий на тест по id пользователя

Этап 6: Тестирование бэкенда

Для данного этапа был использован инструмент для тестирования и автоматизации работы с API, Postman. С его помощью появилась возможность отправлять запросы и получать ответы без необходимости писать логику обращения к серверу вручную. В Postman был создан открытый [workspace](https://app.getpostman.com/join-team?invite_code=4059215423f35b9afd8f6670cf7dea5d), в котором вы можете просмотреть начальные запросы. Не все возможные модели запросов были внесены в Postman, потому что часть из них была создана во время написания фронтенда. [Полный список функций обработки запроса](http://185.42.14.235:8000/docs)

Этап 7: Написание фронтенда

Разработан интерактивный фронтенд, обеспечивающий взаимодействие пользователя с сервером. Реализованы обработчики событий, отправка/получение данных с помощью fetch, отображение информации и валидация данных.

Этап 8: Стилизация фронтенда

Стилизация значительно усовершенствовала разработанный фронтенд, придав ему не только более насыщенную цветовую палитру, но и повысив удобство восприятия информации, а также улучшив взаимодействие пользователей с интерфейсом.

Этап 9: Форматирование кода

На финальном этапе весь код на языке программирования Python был приведен к единому формату. С помощью Flake8 были найдены и исправлены все ошибки и недочеты в оформлении кода.

*Целевая аудитория*: Преподаватели (учителя, профессора, тьюторы и т.д.), студенты (школьники, учащиеся, аспиранты и т.д.). [Отзывы](https://docs.google.com/spreadsheets/d/1IrCUL2N4r1c9qAkLHuFbbe-74laOpYBs77y8a2gcUD4/edit?resourcekey=&gid=184338167#gid=184338167)

*Описание продукта*: Web-сервис с API и web интерфейсом. Для проведения тестирования web-сервис развёрнут на виртуальной машине в сети интернет, доступ к сервису открытый, любой пользователь может пройти регистрацию и пользоваться возможностями сервиса. Функционал сервиса доступный через web-интерфейс и api - создание, каталогизация и поиск тестов, также прохождение тестирования, выставление оценок и просмотр статистики. Статистика имеет 2 вида: статистика участника теста и статистика создателя теста. Любой участник может быть как создателем, так и участником теста. Были реализованы следующие пользовательские сценарии:

1)Регистрация пользователя

1. Пользователь вводит адрес электронной почты, с которым он будет заходить в систему.

2. Пользователь вводит пароль, с которым он будет заходить в систему.

3. Если выбранный адрес электронной почты уже используется в системе, то пользователю сообщается об этом и предлагается войти.

4. Если пароль содержит менее 8 символов или не содержит спец. символов система сообщает, что пароль должен быть более 8 символов и пользователь должен придумать новый пароль.

5. Если введённый адрес электронной почты не соответствует формату, то система выводит сообщение об ошибке и просит ввести адрес ещё раз.

6. Если все введённые данные соответствуют требованиям регистрации, то система подтверждает успешную регистрацию.

2)Вход в систему пользователя

1. Пользователь вводит адрес электронной почты.

2. Пользователь вводит пароль.

3. Если адрес электронной почты или пароль неверны, система сообщает об ошибке и предлагает повторить ввод.

4. Если данные введены правильно, пользователь переходит в главное меню системы.

3)Создание теста

1. В главном меню пользователь выбирает опцию "Мои тесты".

2. Пользователь выбирает опцию "Создать тест".

3. Пользователь вводит название теста.

4. Пользователь задает описание теста (по желанию).

5. Пользователь задает количество вопросов, которое будет в его тесте.

6. Пользователь задает время на прохождение теста (по желанию).

7. Пользователь задает количество попыток на прохождение теста (по желанию).

8. Пользователь выбирает, показывать ли человек, который проходит тест результаты после выполнения теста.

9. Пользователь выбирает, будет ли человек, который проходит тест иметь возможность возвращаться к предыдущим вопросам.

10. Пользователь выбирает, будет ли человек, который проходит тест иметь возможность изменить ответ к предыдущим вопросам.

11. Пользователь задает количество простых вопросов, которые будут показаны при прохождении теста(по желанию).

12. Пользователь задает количество вопросов средней сложности, которые будут показаны при прохождении теста(по желанию).

13. Пользователь задает количество сложных вопросов, которые будут показаны при прохождении теста(по желанию).

14. После завершения ввода данных система сохраняет тест если все данные введены корректно, иначе возвращает пользователю ошибку.

4)Добавление вопроса к тесту

1. В главном меню пользователь выбирает опцию "Мои тесты".

2. Пользователь выбирает опцию "Список тестов".

3. Пользователь выбирает конкретный тест, к которому хочет добавить вопрос.

4. Пользователь выбирает опцию "Вопросы к тесту".

5. Пользователь выбирает опцию "Добавить вопрос".

6. Пользователь вводит текст вопроса.

7. Пользователь вводит описание вопроса(по желанию).

8. Пользователь вводит время выделенное на вопрос(по желанию).

9. Пользователь вводит количество баллов за правильный ответ на вопрос(по желанию).

10. Пользователь выбирает тип вопрос(вопрос с выбором одного варианта ответа, вопрос с выбором нескольких вариантов ответа, вопрос с свободным ответом).

11. Пользователь вводит номер вопроса(по желанию).

12. Пользователь выбирает сложность вопроса(по желанию).

13. Пользователь делает возможность просмотреть подсказку на вопрос(по желанию).

14. После завершения ввода данных система сохраняет вопрос, если все данные введены корректно, иначе возвращает пользователю ошибку.

5)Добавление ответа к вопросу

1. В главном меню пользователь выбирает опцию "Мои тесты".

2. Пользователь выбирает опцию "Список тестов".

3. Пользователь выбирает конкретный тест, к которому хочет добавить вопрос.

4. Пользователь выбирает опцию "Вопросы к тесту".

5. Пользователь выбирает конкретный вопрос, к которому хочет добавить ответ.

6. Пользователь выбирает опцию "Добавить ответ".

7. Пользователь вводит текст вопроса.

8. Пользователь вводит количество баллов за ответ(по желанию).

9. Пользователь вводит номер ответа(по желанию)

10. Пользователь выбирает, будет ли этот ответ являться правильным или нет.

11. После завершения ввода данных система сохраняет, ответ если все данные введены корректно, иначе возвращает пользователю ошибку.

6)Начать тестирование

1. В главном меню пользователь выбирает опцию "Тесты".

2. Пользователь выбирает конкретный тест, который он хочет пройти.

3. Пользователь выбирает опцию "Начать тест".

7)Смена данных профиля

1. В главном меню пользователь выбирает опцию "Профиль".

2. Пользователь выбирает, что он хочет изменить(почту или пароль).

3. При желании изменить пароль пользователь выбирает опцию "Изменить пароль".

3.1. Пользователь вводит новый пароль и затем – старый.

4. При желании изменить почту пользователь выбирает опцию "Изменить почту".

4.1. Пользователь вводит новую почту.

5. После завершения ввода данных система сохраняет изменения, если все данные введены коректно, иначе возвращает пользователю ошибку.

8)Ответ на вопрос с развернутым ответом

1. Пользователь видит вопрос с развернутым ответом.

2. Пользователь вводит свой ответ в текстовое поле.

3. После ввода ответа пользователь нажимает "Следующий вопрос".

4. Система сохраняет ответ и переходит к следующему вопросу.

9)Ответ на вопрос с выбором одного или нескольких вариантов ответа

1. Пользователь видит вопрос с вариантами ответов.

2. Пользователь выбирает один или несколько вариантов ответа.

3. После выбора пользователь нажимает "Следующий вопрос".

4. Система сохраняет ответ и переходит к следующему вопросу.

10)Просмотр результатов для создателя теста

1. В главном меню создатель теста выбирает опцию "Результаты".

2. Создатель выбирает опцию "Результаты прохождения моих тестов".

3. Система отображает список тестов создателя.

4. Создатель выбирает тест, по которому хочет просмотреть результаты.

5. Создатель может видеть результаты каждого пользователя и общее количество участников определенного теста.

11)Просмотр результата пользователем, прошедшим тест

1. В главном меню пользователь выбирает опцию "Результаты".

2. Система показывает результаты всех тестов, которые пользователь прошел.

3. Пользователь может видеть свои результаты и статистику по каждому тесту.

*Рефлексия*: Проект был выполнен как индивидуальный, в связи с этим был недостаток времени. Стоило сделать проект как групповой, в котором бы состояли бэкенд и фронтенд разработчик. Для улучшения пользовательского интерфейса стоило обратиться за помощью к профессиональному дизайнеру. К сожалению, не удалось сделать проверку email’а путем отправки на него письма с кодом подтверждения, учесть возможность прерывания сессии пользователя при прохождении теста, создать автоматическое развертывание из-за нехватки времени. В процессе были приобретены навыки по работе с фреймворками FastAPI, SQLite, SqlAlchemy, сервисом совместной разработки GitHub. В ходе анализа была найдена возможность улучшения функциональности и удобства проекта за счет использования таких популярных фреймворков, как React, Vue и Angular. Были освоены основы языков программирования JavaScript, CSS и HTML. В процессе работы с бэкендом были усовершенствованы навыки работы с Python.

*Как можно дальше развивать проект*: На данном этапе проект имеет минимальный функционал для работы. Для дальнейшего расширения продукта целесообразно сосредоточить усилия на оптимизации пользовательского опыта. Это может быть достигнуто через внедрение высокоэффективных методов, таких как персонализированные рекомендации и усовершенствованная навигация по веб-ресурсу. Важно также рассмотреть возможность расширения функциональных возможностей как на фронтенд, так и на бэкенд-уровнях. Это может включать в себя добавление новых API, оптимизацию существующих функций и интеграцию с современными фреймворками, такими как React или Vue.js. Кроме того, привлечение квалифицированного дизайнера для разработки современного визуального оформления веб-сайта может значительно улучшить его эстетическое восприятие и повысить юзабилити. Данные меры способны не только улучшить общее восприятие продукта, но и привести к увеличению коэффициента конверсии и уровню удовлетворенности пользователей. В совокупности все предложенные инициативы будут способствовать созданию более привлекательного и функционального продукта, соответствующего актуальным ожиданиям пользователей и требованиям современного рынка.В дополнение стоит добавить интеграцию со сторонними сервисами, такими как YandexGPT для автоматизации проверки тестов. Планируется изменить СУБД при популяризации проекта, в связи с тем, что SQLite подходит как СУБД для мини-проекта, но в больших масштабах стоит заменить её на PostgreSQL.

*Отзывы пользователей сервиса*:



















