МИНИСТЕРСТВО ПРОСВЕЩЕНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ ПЕДАГОГИЧЕСКИЙ
УНИВЕРСИТЕТ им. А. И. ГЕРЦЕНА»

**ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И
ТЕХНОЛОГИЧЕСКОГО ОБРАЗОВАНИЯ**
Кафедра информационных технологий и электронного обучения

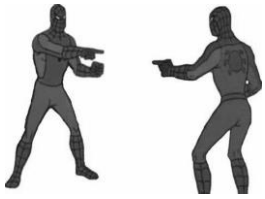
Основная профессиональная образовательная программа
Направление подготовки 09.03.01 Информатика и вычислительная техника
Направленность (профиль) «Технологии разработки программного обеспечения»
форма обучения – очная

ТЕХНИЧЕСКОЕ ЗАДАНИЕ
к выпускной квалификационной работе на тему

«Разработка адаптивной системы обучения с использованием алгоритмов машинного обучения для персонализации учебного процесса»

Заказчик
Клементьев Алексей Александрович

Разработчик
Клементьев Алексей Александрович



Санкт-Петербург
2024

Оглавление

1 Общие положения

Целью разработки является создание адаптивной системы обучения, использующей алгоритмы машинного обучения для персонализации учебного процесса. Система будет обеспечивать автоматическую адаптацию учебных материалов и задач под индивидуальные потребности студентов, основываясь на анализе их уровня знаний, который выполняется с помощью моделей машинного обучения, таких как Deep Q-Learning, Few-Shot Learning и Bayesian Knowledge Tracing. Система будет интегрирована с экспертной системой для предоставления обратной связи и использования методов обучения с подкреплением, что позволит динамически изменять рекомендации для каждого студента в зависимости от полученных данных.

Объектом разработки является программная система, включающая несколько ключевых компонентов, взаимодействующих между собой для реализации адаптивного обучения. В систему входит модель обучения с подкреплением (Deep Q-Learning), которая будет использовать данные о предыдущих заданиях и результатах тестирования студентов для определения наилучших рекомендаций по учебным материалам. Также система будет интегрировать Bayesian Knowledge Tracing для отслеживания и прогноза уровня знаний студентов по различным навыкам, а также Few-Shot Learning для инициализации модели на основе ограниченного объема начальных данных.

Система будет включать экспертную систему, которая будет оценивать выполнение заданий и предоставлять рекомендации по корректировке учебного процесса, используя базы данных SQL или NoSQL для хранения данных. Оценка будет осуществляться с помощью алгоритмов, основанных на вероятностных моделях и логических выводах, что обеспечит качественное взаимодействие с пользователем и точность рекомендаций. Локальная языковая модель будет генерировать объяснения к действиям модели, предоставляя студенту интерпретацию рекомендаций на естественном языке.

Разработка будет осуществляться в соответствии с нормативными документами, регулирующими процесс создания программного обеспечения, включая ГОСТ Р 34.602-89 «Техническое задание на создание автоматизированной системы» и ГОСТ Р 7.0.97-2016 «Системы автоматизированного проектирования». Система будет соответствовать современным стандартам информационной безопасности, включая требования по защите данных пользователей и соблюдению конфиденциальности, а также стандартам эргономики пользовательских интерфейсов.

2 Назначение разработки

Разработка предназначена для применения в образовательных учреждениях, как в рамках академического процесса, так и для внешнего обучения. Система будет использовать алгоритмы машинного обучения для персонализации учебного процесса, предоставляя студентам адаптированные рекомендации по учебным материалам и заданиям. Алгоритмы, такие как Deep Q-Learning, Bayesian Knowledge Tracing и Few-Shot Learning, будут использоваться для анализа уровня знаний студентов и динамической адаптации учебных материалов. В частности, система будет автоматически формировать рекомендации по задачам на основе текущего уровня знаний студента, что позволит повысить эффективность усвоения материала.

Преподаватели смогут использовать систему для мониторинга успеваемости студентов, получения отчетности о прогрессе по каждому студенту и корректировки учебного процесса в реальном времени. Для этого будет предусмотрена административная панель, позволяющая преподавателям получать доступ к данным о результатах тестов, выполненных заданиях и анализу успеваемости с использованием данных из Bayesian Knowledge Tracing. Преподаватель будет иметь возможность вручную корректировать параметры обучения, а также вносить изменения в историю подкреплений и наказаний, если это необходимо, с помощью интегрированной экспертной системы.

Система обеспечит интеграцию технологий машинного обучения в образовательный процесс, что позволит автоматизировать процессы адаптации учебных материалов и заданий. В частности, алгоритм Deep Q-Learning будет использоваться для формирования рекомендаций по заданиям, принимая решения на основе полученных данных о предыдущих действиях студента, а также его результатах тестирования. Bayesian Knowledge Tracing будет использоваться для отслеживания уровня знаний студента по конкретным навыкам, что позволит более точно настраивать систему под потребности каждого учащегося. Few-Shot Learning обеспечит возможность инициализации модели с минимальными данными о студенте, что будет полезно в условиях ограниченных стартовых данных.

Применение этих технологий позволит значительно сократить время, необходимое преподавателям для подготовки и организации учебного процесса, а также повысить мотивацию студентов, поскольку каждый студент будет получать индивидуальные рекомендации, соответствующие его текущему уровню знаний.

3 Требования к функциональным характеристикам

3.1 Функции системы

Система автоматически анализирует уровень знаний студентов на основе результатов начального тестирования, а также исторических данных о выполненных заданиях и их результатах. Для начальной оценки знаний студентов используется методика, включающая тестирование, которое проводится с целью сбора данных о текущем уровне навыков студента в различных областях. Полученные данные передаются в агентную модель, основанную на алгоритме Deep Q-Learning. Модель использует эти данные для формирования рекомендаций по учебным материалам и заданиям, которые адаптированы к индивидуальным потребностям студентов. В процессе обучения используются методы обучения с подкреплением, при которых агент анализирует, какие действия (рекомендации) наилучшим образом способствуют улучшению знаний студента. Система применяет Bayesian Knowledge Tracing (BKT) для отслеживания уровня знаний студентов по каждому конкретному навыку и теме. Модель BKT помогает динамически обновлять состояние знаний студента, основываясь на его ответах и успехах в выполнении заданий.

Для оценки выполнения заданий и корректировки учебного процесса используется экспертная система, которая анализирует ответы студентов и принимает решения о корректировке рекомендаций. Экспертная система использует базу данных правильных и неправильных ответов для оценки вероятности успешного ответа на новые вопросы. В случае необходимости экспертная система позволяет преподавателям вносить изменения в логику адаптации системы через административную панель. Локальная языковая модель генерирует текстовые обоснования рекомендаций, основываясь на данных, предоставленных Bayesian Knowledge Tracing и другими компонентами системы. Эти обоснования объясняют студентам, почему были выбраны те или иные материалы или задания, что повышает прозрачность работы системы и помогает студентам лучше понять логику обучения.

3.2 Взаимодействие компонентов системы

Все компоненты системы работают в тесной связке для обеспечения адаптивного учебного процесса, в котором каждый элемент системы взаимодействует с другими для формирования наиболее эффективных рекомендаций для студентов. Начальное тестирование выполняется на основе набора заданий, результат которых используется для создания начального профиля знаний студента. Эти данные передаются в модель агента, которая использует алгоритм Deep Q-Learning для принятия решений о том, какие учебные

материалы и задания наиболее подходят студенту. Модель оценивает текущее состояние студента и обновляет свои рекомендации с учетом изменений в уровнях знаний, отслеживаемых с помощью модели Bayesian Knowledge Tracing. Модель BKT анализирует ошибки и правильные ответы студента, обновляя вероятность того, что студент освоил конкретный материал. Эти данные используются агентом для формирования более точных рекомендаций по задачам и материалам, что способствует персонализированному подходу в обучении.

Экспертная система в реальном времени анализирует ответы студентов на задания и корректирует параметры обучения модели, например, изменяя веса для различных типов заданий или материалов. Это может происходить автоматически, на основе предварительно заданных правил, или вручную, через административную панель, где преподаватель может внести изменения в настройки адаптивного процесса. Локальная языковая модель генерирует текстовые пояснения к рекомендациям, опираясь на данные из Bayesian Knowledge Tracing и другие компоненты, что позволяет студентам получать обоснованные объяснения того, почему определенные задания или материалы были рекомендованы. Эти текстовые обоснования необходимы для повышения прозрачности и понимания принятия решений системой.

4 Требования к надежности

Система гарантирует сохранность данных студентов и их прогресса через использование защищённой базы данных, которая хранит всю информацию о выполненных заданиях, оценках, прогрессе в обучении и других важных метках. Для обеспечения долговечности данных и предотвращения их потери в случае сбоев предусмотрены механизмы регулярного резервного копирования. В качестве базы данных используется PostgreSQL, обеспечивающая высокую степень надежности и поддерживающая функционал транзакций для защиты целостности данных. Резервные копии данных создаются на регулярной основе с использованием средств автоматизации, таких как cron jobs для планирования периодического создания резервных копий, а также настройка бэкапов в облачных хранилищах с использованием API AWS S3 для удалённого хранения данных.

Для восстановления данных после сбоев предусмотрены процедуры восстановления, включая методы восстановления на уровне отдельных таблиц базы данных и полное восстановление данных из резервных копий. В случае сбоев в работе системы или сервера выполняется восстановление данных на основе последних успешных бэкапов с минимальными временными потерями.

Система включает механизмы обработки ошибок и обеспечения целостности данных. Все ключевые операции с базой данных, такими как создание, обновление и удаление записей, контролируются с использованием транзакций. Для отслеживания возможных ошибок и сбойных ситуаций реализован функционал логирования с использованием библиотеки logging в Python, что позволяет вести журнал всех исключений, ошибок и аномальных событий. Лог-файлы содержат детализированную информацию, включая время возникновения ошибки, тип исключения, а также контекст и данные, которые привели к сбою. В случае критических ошибок система автоматически генерирует уведомления для администраторов с помощью интеграции с системой мониторинга, например, через использование API платформы Prometheus или интеграции с системой уведомлений, такой как Slack или электронной почтой.

Кроме того, система спроектирована с учетом устойчивости к нагрузкам. В случае частичных сбоев, например, недоступности отдельных микросервисов или компонентов системы, она продолжает функционировать с ограниченной производительностью или с отключением несущественных для работы компонентов. Для обеспечения отказоустойчивости используются подходы, такие как кластеризация базы данных и распределённая обработка данных с использованием контейнеризации (например, Docker и Kubernetes). Эти технологии позволяют системе масштабироваться в зависимости от объема нагрузки и обеспечивать высокую доступность сервисов даже при увеличении числа пользователей.

5 Условия эксплуатации

5.1 Описание предполагаемой среды

Система размещается на серверной архитектуре, использующей облачные технологии для обеспечения гибкости и масштабируемости. В качестве облачной платформы применяется Amazon Web Services (AWS), предоставляющая возможность автоматического масштабирования и отказоустойчивости через сервисы Amazon EC2 и Amazon RDS для управления виртуальными машинами и базами данных. Все вычисления и обработка данных машинного обучения, включая обработку данных для алгоритмов Deep Q-Learning, Bayesian Knowledge Tracing и других методов, выполняются на серверной стороне, где используются виртуализированные серверы, развернутые с помощью Docker-контейнеров. Контейнеризация обеспечивает изоляцию и гибкость, а оркестрация с использованием Kubernetes позволяет автоматически распределять нагрузку между

различными узлами системы, обеспечивая её высокую доступность и стабильность при изменении объема данных и нагрузки.

Взаимодействие между компонентами системы осуществляется через RESTful API, использующее HTTP/HTTPS протоколы для передачи данных в формате JSON. API реализует все основные операции взаимодействия между компонентами, включая передачу данных о текущем уровне знаний студентов, а также интеграцию с внешними сервисами, такими как платформы для дистанционного обучения и системы управления контентом. API для внешних интеграций и взаимодействия между компонентами использует аутентификацию с помощью токенов JWT (JSON Web Token), что позволяет обеспечивать безопасность передачи данных.

Интерфейс пользователя реализуется как веб-приложение, использующее современные веб-технологии, включая HTML5 для структуры страниц, CSS3 для стилизации и JavaScript для интерактивности. Для управления фронтендом используется фреймворк React, что позволяет организовать динамическое обновление данных на страницах и повышает отзывчивость интерфейса. Серверная часть веб-приложения реализована с использованием Django, фреймворка Python, что позволяет эффективно управлять запросами, базами данных и логикой приложения. Все данные, включая результаты тестирования и прогресса студентов, обрабатываются и хранятся в базе данных, интегрированной через ORM (Object-Relational Mapping) для упрощения взаимодействия между приложением и базой данных.

5.2 Условия для пользователей

Система доступна через веб-браузеры, которые поддерживают современные стандарты HTML5 и CSS3. Требования к браузерам включают поддержку современных JavaScript-API и стандартов безопасности, таких как HTTPS и Content Security Policy. Минимальное разрешение экрана для корректного отображения интерфейса системы составляет 1280x720 пикселей, что обеспечивает удобство работы с интерфейсом, включая все динамические элементы и визуализацию данных.

Для преподавателей система предоставляет административную панель, которая реализована с использованием интерфейса React и поддерживает управление учебными процессами, настройку алгоритмов обучения и просмотр отчетов о прогрессе студентов. Панель предоставляет возможность настройки параметров модели обучения, корректировки данных и выводов, а также включает механизмы мониторинга текущего состояния системы и просмотра статистики работы всех компонентов.

Доступ студентов к системе осуществляется через персонализированные аккаунты. Аутентификация студентов проводится с использованием безопасной системы OAuth 2.0 с двухфакторной аутентификацией (2FA), что гарантирует высокий уровень безопасности и защиту учетных данных пользователей. Система поддерживает возможность создания различных ролей пользователей с ограничением прав доступа в зависимости от их статуса (студент, преподаватель, администратор).

6 Требования к составу и параметрам технических средств

6.1 Серверная часть

Серверная часть системы должна быть способна выполнять все вычислительные задачи, включая обработку запросов пользователей, обучение и работу моделей машинного обучения, а также взаимодействие с клиентской частью. Для этого сервер должен оснащаться многоядерным процессором, например, Intel Xeon или AMD EPYC, с минимум 16 ГБ оперативной памяти, что обеспечивает достаточные вычислительные ресурсы для параллельной обработки данных и выполнения алгоритмов. Для ускорения вычислений, связанных с обучением моделей машинного обучения, таких как Deep Q-Learning и Few-Shot Learning, необходимы графические процессоры (GPU), например, NVIDIA Tesla V100 или A100, которые обеспечивают значительное увеличение скорости обработки больших объемов данных. Обучение моделей с использованием TensorFlow или PyTorch будет происходить на этих GPU, что значительно уменьшает время на тренировку моделей.

Для обеспечения отказоустойчивости и масштабируемости серверная инфраструктура должна использовать технологии виртуализации и контейнеризации. В качестве виртуализационного решения будет использован гипервизор VMware или KVM для создания виртуальных машин, а контейнеризация будет реализована с помощью Docker, что позволит эффективно распределять ресурсы между различными компонентами системы. Оркестрация контейнеров будет осуществляться с помощью Kubernetes, что обеспечит автоматическое масштабирование в зависимости от текущей нагрузки и минимизирует простои при сбоях.

Данные пользователей, включая результаты тестирования и прогресса обучения, будут храниться в защищенных базах данных, которые обеспечивают поддержку транзакций и высокую степень масштабируемости. Для хранения структурированных данных будет использована реляционная база данных PostgreSQL с поддержкой ACID-свойств транзакций, а для хранения нереляционных данных, таких как результаты анализа учебных материалов, будет применена NoSQL база данных MongoDB. Оба типа баз данных

будут настроены для горизонтального масштабирования с помощью репликации и шардинга, что обеспечит высокую доступность и производительность.

Для защиты данных сервер будет оснащен системой резервного копирования, которая будет регулярно сохранять копии всех данных на удаленные серверы с использованием технологий, таких как Amazon S3 или аналогичных облачных решений. Также будут реализованы механизмы для защиты данных от несанкционированного доступа с использованием технологий шифрования (например, AES-256) и аутентификации с помощью OAuth 2.0 и JWT для безопасной передачи данных между клиентом и сервером.

Интеграция с внешними сервисами, такими как API для языковых моделей или серверы для тяжелых вычислений, будет реализована через RESTful API с использованием JSON для передачи данных. Это обеспечит удобное взаимодействие между компонентами системы и внешними сервисами, такими как API обработки естественного языка (например, OpenAI GPT), и позволит расширить функциональность системы.

6.2 Клиентская часть

Клиентская часть системы представляет собой веб-приложение, которое будет реализовано с использованием HTML5, CSS3 и JavaScript. Для разработки интерактивного интерфейса будет использован фреймворк React, который обеспечивает эффективное управление состоянием приложения и быструю отрисовку компонентов. Взаимодействие с серверной частью будет осуществляться через RESTful API, что обеспечит гибкость и масштабируемость взаимодействия клиент-сервер. Для повышения производительности клиентской части используется библиотека Axios, которая оптимизирует HTTP-запросы.

Клиентское веб-приложение должно поддерживать работу на популярных браузерах, таких как Google Chrome, Mozilla Firefox, Safari и Microsoft Edge, и обеспечивать кроссбраузерность за счет использования современных стандартов веб-разработки. Адаптивность интерфейса будет достигаться с помощью фреймворка Bootstrap или медиазапросов CSS, что обеспечит корректное отображение приложения на различных устройствах, включая смартфоны и планшеты с минимальным разрешением экрана 1280x720 пикселей.

Для обеспечения безопасности клиентской части системы будет реализована защита от атак типа Cross-Site Scripting (XSS) и Cross-Site Request Forgery (CSRF) с помощью соответствующих фильтров и механизмов защиты. Все данные, передаваемые между клиентом и сервером, будут шифроваться с использованием протокола HTTPS, что обеспечит безопасность передачи данных и защиту от атак типа Man-in-the-Middle (MITM). Для предотвращения XSS-атак будет использоваться библиотека React, которая

автоматически экранирует данные, выводимые на страницу, и предотвращает выполнение нежелательных скриптов.

7 Требования к программному обеспечению

Основным языком программирования для реализации системы будет Python. Этот язык предоставляет высокую гибкость для разработки серверной логики и интеграции с различными библиотеками для машинного обучения и обработки данных. Для реализации моделей машинного обучения и взаимодействия с внешними сервисами Python является оптимальным решением, так как его экосистема включает в себя множество специализированных библиотек, которые могут быть использованы для решения различных задач в рамках проекта. Для разработки фронтенд-части системы используется JavaScript с фреймворками React и Vue.js. React будет выбран для создания динамичного, компонентного пользовательского интерфейса с возможностью эффективного обновления отдельных компонентов на основе состояния приложения, что особенно важно для образовательных платформ с динамично меняющимся контентом. Vue.js может быть использован в случае, если требуется менее громоздкая структура, но с возможностью расширения и гибкости для более простых пользовательских интерфейсов.

Для реализации алгоритмов машинного обучения в проекте будут использованы следующие библиотеки. TensorFlow и PyTorch применяются для разработки и обучения моделей машинного обучения, включая модели Deep Q-Learning (DQN) для адаптации учебного процесса и Few-Shot Learning для работы с ограниченными данными. TensorFlow и PyTorch предоставляют оптимизированные решения для обучения нейронных сетей, включая поддержку GPU и распределенных вычислений, что значительно ускоряет процесс обучения и позволяет эффективно обрабатывать большие объемы данных. В системе PyTorch будет использоваться для создания нейронных сетей с алгоритмами обучения с подкреплением, в частности для реализации DQN, где модель будет обучаться на основе взаимодействия с окружающей средой и получать награды за правильные действия, оптимизируя обучение в реальном времени. В дополнение к этому, для реализации методов предобработки данных и алгоритмов машинного обучения, таких как классификация и кластеризация, будет использоваться Scikit-learn. Эта библиотека предоставляет широкий спектр алгоритмов для работы с данными, включая алгоритмы для нормализации, уменьшения размерности, классификации и кластеризации, которые будут применяться на этапе подготовки данных для обучения моделей.

Для реализации модели Bayesian Knowledge Tracing (BKT), которая будет отслеживать уровень знаний студентов по различным учебным темам, будет использована

библиотека PyMC3. Эта библиотека предоставляет инструменты для работы с байесовскими моделями и будет использоваться для построения вероятностной модели, которая обновляется на основе данных о текущем уровне знаний студентов. Важным аспектом является использование байесовского подхода для учёта неопределенности и изменения знаний студента в процессе обучения. PyMC3 обеспечит реализацию алгоритмов MCMC (Markov Chain Monte Carlo) для выборки из распределений и оптимизации параметров модели.

Для разработки экспертной системы, которая будет автоматически оценивать результаты тестов и формулировать рекомендации, будет использован фреймворк для создания логических систем, таких как CLIPS или PyKE. Эти фреймворки позволяют разрабатывать экспертные системы, включающие базы правил и фактов, с возможностью интеграции с другими компонентами системы. CLIPS и PyKE будут использоваться для создания базы знаний, состоящей из логических правил, которые будут применяться для генерации рекомендаций на основе результатов тестов студентов.

Для серверной части системы будет выбран фреймворк Django или Flask, в зависимости от требований к масштабу и сложности проекта. Django предоставляет более высокоуровневые инструменты для разработки веб-приложений, включая встроенные механизмы работы с базами данных и администрирования, что делает его подходящим для крупных и масштабируемых приложений. Flask, с другой стороны, предоставляет более легковесную структуру и больше гибкости, что может быть предпочтительно в случае создания простых сервисов с минимальной серверной логикой. Оба фреймворка обеспечат создание RESTful API, а также будут использоваться для обработки запросов и маршрутизации. Для работы с базами данных будет использоваться Django ORM, если выбран Django, или SQLAlchemy, если используется Flask. Django ORM позволит разработчикам работать с базами данных на уровне объектов, а SQLAlchemy предоставляет более тонкую настройку для работы с SQL-запросами и моделями данных.

Для реализации взаимодействия с внешними сервисами, такими как API языковых моделей или серверы для выполнения тяжелых вычислений, будет использована библиотека Requests для обработки HTTP-запросов и получения данных от внешних сервисов. Requests будет использоваться для отправки запросов в формате JSON, а также для получения и обработки ответов от серверов внешних систем. Для создания высокопроизводительных API, обеспечивающих быструю обработку запросов, будет применен фреймворк FastAPI. FastAPI использует асинхронную обработку запросов и может работать с типами данных в Python, что обеспечит высокую скорость работы с API и поддержку асинхронных операций для обработки запросов пользователей.

Для визуализации данных и построения отчетности по результатам тестирования и обучения будет использована библиотека Matplotlib для создания статичных графиков и Seaborn для визуализации данных с более высокоуровневыми функциями. В случае необходимости создания интерактивных визуализаций, например, для динамичных графиков, будет использован Plotly, который позволяет создавать визуализации, взаимодействующие с пользователем в реальном времени.

Для тестирования системы будут применяться библиотеки unittest и pytest, которые обеспечат автоматизацию тестирования на разных уровнях. Эти библиотеки будут использоваться для тестирования серверной логики, интерфейса и интеграции компонентов системы. unittest предоставляет базовые средства для написания модульных тестов, а pytest будет использоваться для более сложных тестов с дополнительными возможностями для асинхронных операций и интеграционного тестирования.

8 Требования к информационному обеспечению

8.1 Описание структуры базы данных

Структура базы данных системы должна быть спроектирована для эффективного хранения данных о студентах, их знаниях, результатах тестирования и взаимодействии с обучающим процессом. База данных должна обеспечивать хранение информации о студентах, включая уникальные идентификаторы, данные о завершенных заданиях, полученные оценки, информацию о типичных ошибках, а также время выполнения заданий. Для этого структура таблиц должна включать поля, предназначенные для хранения таких данных, как дата и время выполнения, идентификаторы конкретных тестов или заданий, а также оценки, присвоенные студентам по результатам их выполнения. Важным аспектом является также хранение данных о заданиях, включая их тип, сложность, рекомендации по выполнению, а также связи с соответствующими учебными материалами и темами. Для хранения этих данных будет использоваться реляционная база данных, такая как PostgreSQL или MySQL, обеспечивающая возможность работы с транзакциями, а также поддержку отношений между таблицами (например, через внешние ключи и связи между таблицами «студенты», «задания», «оценки»). В случае необходимости хранения неструктурированных данных, например, результатов моделей машинного обучения или параметров обучения, также предусмотрено использование NoSQL решений, таких как MongoDB, для гибкости в управлении данными, которые могут варьироваться по структуре.

Для хранения и управления моделями машинного обучения, таких как веса нейронных сетей и параметры обучения, база данных должна обеспечивать поддержку

хранения бинарных данных, например, в формате Blob, что позволит эффективно управлять большими объемами данных, связанными с обучением моделей. Каждая модель будет храниться как отдельный объект в базе данных, с уникальными идентификаторами для каждой версии модели и метаданными, связанными с процессом обучения (например, датой последнего обновления, точностью модели и использованными данными). Это обеспечит возможность регулярного обновления моделей в зависимости от новых данных о студенте и его успехах.

База данных должна поддерживать транзакционность и целостность данных. Для этого будет настроено использование механизмов ACID, что гарантирует, что все операции с базой данных (например, добавление новых записей или обновление существующих) выполняются атомарно, с сохранением целостности данных. Для масштабируемости и обеспечения высокоскоростной работы при увеличении объема данных будет применяться шардирование и репликация. Эти технологии позволят эффективно распределять нагрузку между несколькими серверами и обеспечивать отказоустойчивость системы.

8.2 Виды и форматы входных и выходных данных

Входные данные для системы включают информацию о студенте, результаты начальных тестов, данные об успехах и ошибках, а также информацию, поступающую в процессе обучения. Для большинства входных данных будет использован формат JSON, поскольку он предоставляет удобную структуру для представления данных и широко поддерживается во всех современных технологиях. В зависимости от типа задачи, данные могут быть представлены в виде строк или списков (например, для представления ответов студентов на вопросы), либо в виде матриц или тензоров (для представления параметров задания или характеристик студента). Для передачи таких данных через API будет использоваться формат JSON, что позволит интегрировать систему с внешними сервисами, а также с другими компонентами экосистемы.

Для нейронных сетей входные данные, такие как характеристики студентов и их ответы, будут преобразованы в числовые векторы или тензоры, которые подаются в модель для обучения или оценки. Векторы признаков будут представлять собой данные о каждом студенте, включая его успехи и ошибки по предыдущим заданиям, а также результаты выполнения текущих заданий. Эти данные будут масштабироваться и нормализоваться перед подачей в модели машинного обучения, что обеспечит их корректную работу и ускорит процесс обучения.

Выходные данные системы включают рекомендации по заданиям, оценки, прогнозы уровня знаний студентов, а также обновления состояний моделей машинного обучения.

Рекомендации и прогнозы, генерируемые системой, будут представлены в виде числовых значений или векторов вероятностей, которые указывают на вероятность успешного выполнения задания студентом на основе его текущих знаний. Эти данные могут быть использованы для принятия решений в системе, таких как выбор подходящих заданий для следующего этапа обучения или изменение параметров модели в зависимости от прогресса студента. Результаты, полученные от нейронных сетей, будут представлены в виде числовых значений или векторов вероятностей, которые служат в качестве предсказаний для следующего шага в обучении.

Для передачи выходных данных в другие системы будет использоваться формат JSON, что обеспечит удобство интеграции и обмена данными между компонентами системы. Эти данные будут структурированы с учетом требований к взаимодействию с внешними сервисами через API. JSON позволит передавать как простые данные, такие как текстовые сообщения или оценки, так и более сложные структуры данных, такие как списки рекомендаций или параметры обучения моделей.

9 Требования к интерфейсу

9.1 Пользовательский интерфейс по видам пользователей

Пользовательский интерфейс системы должен быть адаптирован для различных категорий пользователей: студентов и преподавателей, с учетом их ролей и целей взаимодействия с системой. Для студентов будет разработан интерфейс, ориентированный на удобное прохождение тестирований, получение рекомендаций по заданиям и отслеживание прогресса обучения. Интерфейс будет включать в себя динамичные страницы, созданные с использованием фреймворков React или Vue.js, которые обеспечат гибкость и интерактивность. Студенты смогут просматривать результаты своих тестов, информацию о текущем уровне знаний и рекомендации по дальнейшему обучению. Данные о прогрессе студента будут извлекаться из базы данных с использованием SQL-запросов, оптимизированных для быстрого получения информации о последних результатах тестирования и изменениях в знаниях студента. Рекомендации будут генерироваться на основе анализа данных о выполненных заданиях и предпочтений студента, полученных через API, интегрированные с моделями машинного обучения, реализованными с использованием TensorFlow или PyTorch.

Для преподавателей интерфейс будет предоставлять доступ к подробной информации о результатах работы студентов. Преподаватели смогут видеть статистику успеваемости, анализировать динамику прогресса и управлять заданиями. Для отображения

статистических данных будут использоваться визуализации, построенные с помощью библиотек Matplotlib и Plotly. Преподаватели смогут корректировать рекомендации, назначать новые задания или изменять параметры текущих заданий на основе своей экспертизы. Вся информация о студентах и их заданиях будет храниться в базе данных, с доступом через ORM-интерфейс Django ORM или SQLAlchemy, что позволит быстро и эффективно получать актуальные данные. Преподаватели смогут обновлять настройки оценки и корректировать параметры рекомендаций через интерактивные формы, реализованные на фронтенде с использованием JavaScript, обеспечивающего динамическое обновление данных на страницах.

9.2 Административная панель

Административная панель системы будет предоставлять полный контроль над всеми аспектами работы системы, включая управление пользователями, заданиями, параметрами обучения и настройками моделей машинного обучения. Интерфейс панели будет реализован с использованием фреймворка Django или Flask, что позволит интегрировать административную панель с серверной частью системы и обеспечит простоту работы с реляционными или NoSQL базами данных для хранения данных о пользователях и заданиях. Администратор сможет создавать, редактировать и удалять пользователей через интерфейс, интегрированный с базой данных, что обеспечит управление доступом к системе на основе ролей. Панель будет поддерживать функционал назначения студентов на курсы, а также управления заданиями, включая создание новых заданий, настройку их параметров и выбор сложности.

Администратор также будет иметь доступ к мониторингу состояния системы в реальном времени, включая данные о производительности серверов, состоянии базы данных и параметрах работы нейронных сетей. Для мониторинга производительности и состояния базы данных будут использоваться инструменты, такие как Grafana или Prometheus, которые обеспечат визуализацию ключевых метрик системы. Важно, чтобы администратор мог вмешиваться в работу экспертной системы для корректировки параметров оценки и рекомендаций, что будет осуществляться через формы и панели управления, интегрированные с логикой экспертной системы, реализованной с использованием библиотеки CLIPS или PyKE. Эти системы позволят задать правила оценки и формирования рекомендаций, которые могут быть изменены в реальном времени в зависимости от требований и экспертной оценки администратора. Панель также будет поддерживать генерацию отчетности и аналитики по ключевым параметрам работы

системы, используя встроенные механизмы для создания отчетов и анализа данных в формате JSON, доступных для выгрузки и дальнейшей обработки.

10 Порядок контроля и приемки

10.1 Описание тестовых сценариев

Тестирование системы будет включать серию сценариев, направленных на проверку всех функциональных и эксплуатационных аспектов. Каждый тестовый сценарий будет четко нацелен на проверку соответствия функционала системе требований и обеспечению стабильной работы всех компонентов. Основными направлениями тестирования будут алгоритмы машинного обучения, включая модель агента, Bayesian Knowledge Tracing и экспертную систему. Для проверки корректности работы модели Deep Q-Learning будет проведено тестирование на базовых и усредненных тестах с использованием различных наборов входных данных. Для этого будет осуществлено проведение экспериментальных тестов с набором студентов, чтобы обеспечить корректность рекомендаций и предсказаний. Алгоритм BKT будет тестироваться путем анализа реальных данных об успехах студентов и проверки на соответствие прогнозов с фактическим прогрессом. Для этого будут применяться специфичные методы анализа, такие как валидация с использованием k-fold cross-validation.

Необходимым будет тестирование взаимодействия всех компонентов системы, таких как процесс начального тестирования студентов, процесс генерации рекомендаций, взаимодействие с экспертной системой и обновление параметров модели на основе полученных данных. Для этого будет проверено, как каждый компонент взаимодействует с другими в условиях нормальной эксплуатации, используя интеграционные тесты, созданные с использованием библиотеки pytest. Для тестирования обработки данных, включая корректность ввода и вывода, будет использована база данных PostgreSQL или MySQL, с автоматическим тестированием через создание тестовых данных и проверку их корректного сохранения и извлечения с использованием ORM-фреймворка Django ORM или SQLAlchemy. Особое внимание будет уделено тестированию API через автоматическое тестирование с использованием библиотеки requests или Postman для обеспечения корректности взаимодействия между сервером, клиентом и сторонними API.

Тестирование пользовательского интерфейса будет проводиться с использованием инструментов для автоматизации тестирования интерфейсов, таких как Selenium, что позволит проверить доступность всех функций для различных типов пользователей, а также стабильно ли работает интерфейс при различных условиях эксплуатации, включая разные

разрешения экранов и устройства. Будет проведено тестирование на разных платформах с применением тестов кросс-браузерной совместимости, что обеспечит стабильную работу интерфейса на всех популярных браузерах, таких как Chrome, Firefox и Safari.

10.2 Условия приемки

Приемка системы будет осуществляться на основании выполнения всех функциональных требований, указанных в проектной документации, и успешного прохождения всех тестовых сценариев. Условия приемки включают обеспечение корректного функционирования алгоритмов машинного обучения, с особым вниманием к точности рекомендаций и правильности анализа уровня знаний студентов. Для этого будет проверено, насколько система точно адаптирует учебный процесс на основе индивидуальных потребностей студентов, что будет оцениваться с помощью анализа точности прогноза модели и дальнейших корректировок на основе реальных данных. В частности, должна быть продемонстрирована способность алгоритмов, таких как Deep Q-Learning и Bayesian Knowledge Tracing, поддерживать точность рекомендаций при реальных данных о студентах.

Также необходимо подтвердить, что все данные, получаемые от пользователей, корректно обрабатываются и сохраняются в базе данных без ошибок, что будет проверяться с помощью юнит-тестов на уровне базы данных, выполняемых на каждом этапе работы с данными. Все данные о пользователях, результатах тестов и рекомендациях должны быть корректно записаны и доступны для дальнейшего анализа, что будет проверяться через запросы к базе данных и консольный вывод.

Все пользовательские интерфейсы должны быть полностью функциональными, а взаимодействие между пользователем и системой должно быть интуитивно понятным и удобным, что будет оцениваться через юзабилити-тестирование с реальными пользователями, а также автоматизированное тестирование интерфейса. Система должна быть протестирована на наличие багов и ошибок, с проведением стресс-тестов для проверки стабильности работы при увеличении нагрузки, включая количество одновременных пользователей и запросов к базе данных. Для этого будут проведены нагрузочные тесты с использованием инструментов, таких как Apache JMeter или Locust. Также будет проведена проверка совместимости программного обеспечения с серверной и клиентской инфраструктурой, в том числе тестирование на различных операционных системах и устройствах.

11 Пользовательские требования

Пользовательские требования определяют условия, при которых система будет удовлетворять потребности конечных пользователей и обеспечивать эффективное взаимодействие с системой. Основными пользователями системы являются студенты, преподаватели и администраторы, все из которых должны иметь доступ к системе через веб-интерфейс, разработанный с использованием фреймворка React для фронтенда и Django для бэкенда, что обеспечит удобство и интуитивно понятное взаимодействие.

Для студентов система должна обеспечивать возможность прохождения начального тестирования, результаты которого будут обрабатываться через API, реализованный с использованием Python и Flask, и на основе которого будет формироваться индивидуальная модель знаний студента. Система будет использовать алгоритмы машинного обучения, такие как Bayesian Knowledge Tracing и Deep Q-Learning, для формирования персонализированных рекомендаций по заданиям и учебным материалам, что обеспечит эффективное усвоение учебного материала. Вся информация о прогрессе студентов, включая их результаты и время выполнения заданий, будет сохраняться в реляционной базе данных PostgreSQL или NoSQL базе данных, в зависимости от сложности и объема данных, с поддержкой транзакций и репликации для обеспечения целостности и надежности. Рекомендации и прогресс студента будут обновляться на основе полученных данных через вызовы API, которые будут производиться на каждом этапе обучения. Студент должен иметь возможность просматривать историю выполнения заданий и оценок через интерфейс, который будет взаимодействовать с сервером через RESTful API. Система также будет предоставлять предложения по улучшению результатов обучения на основе анализа предыдущих ошибок и достижений студента, используя методы анализа данных и отчетности, реализованные на основе Python и библиотеки Pandas для обработки статистических данных.

Преподаватели должны иметь возможность просматривать данные о студентах, включая их успеваемость, результаты тестирований и прогресс в освоении учебных материалов, с использованием административного интерфейса, построенного с использованием React и Django. Преподавательский интерфейс будет обеспечивать доступ к инструментам для настройки и корректировки материалов и заданий. Настройки будут храниться в базе данных, и для их изменения преподаватель будет взаимодействовать с системой через защищенные API-запросы, использующие механизм аутентификации и авторизации с помощью OAuth 2.0. Также преподаватель должен иметь возможность взаимодействовать с экспертной системой для корректировки рекомендаций, используя

инструменты для адаптации моделей обучения, с возможностью выполнения ручной настройки рекомендаций через интерфейс, реализованный на Django. Важным требованием является возможность анализа и оценки работы студентов в реальном времени, что будет обеспечено через использование websocket-соединений для передачи данных о текущем состоянии выполнения заданий и результатов тестирования.

Администраторы системы должны иметь полный доступ ко всем функциональным возможностям, включая управление пользователями, настройку параметров системы, модификацию и обновление материалов и заданий, а также мониторинг состояния системы. Администраторский интерфейс будет построен с использованием фреймворка Django, предоставляющего административные панели, которые будут взаимодействовать с базой данных и обеспечивать управление учетными записями студентов и преподавателей. Администраторы смогут управлять данными пользователей через интерфейс, поддерживающий возможности создания, редактирования и удаления пользователей, а также назначения студентов на курсы и управление доступом к образовательным материалам. Также они будут иметь доступ к инструментам для мониторинга производительности системы, отслеживания состояния серверных приложений и производительности базы данных, включая автоматические алерты и логи, реализованные с использованием системы мониторинга Prometheus и интеграции с Grafana для визуализации.

Кроме того, система должна обеспечивать защиту персональных данных пользователей, включая студентов и преподавателей, с использованием технологий шифрования данных на всех уровнях. Для этого будет использован стандарт HTTPS для шифрования данных при их передаче через сеть. Все данные, получаемые и обрабатываемые системой, включая персональные данные и результаты тестирований, будут храниться в базе данных с использованием шифрования на уровне базы данных, что обеспечит защиту от несанкционированного доступа и утечек. Вся система будет соответствовать законодательным требованиям безопасности данных, таким как Общий регламент защиты данных (GDPR), с учетом обеспечения конфиденциальности, целостности и доступности информации.

12 Анализ и управление рисками

12.1 Аппаратные риски

Аппаратные риски связаны с возможными сбоями в работе серверного оборудования, которые могут привести к утрате данных или снижению производительности системы. Для

минимизации этих рисков требуется регулярное обслуживание серверов, включая мониторинг состояния компонентов через систему мониторинга на базе Prometheus и интеграцию с Grafana для отображения производительности. Все серверы должны быть подключены к инфраструктуре резервного копирования, использующей решения, такие как регулярное создание снимков данных и использование репликации базы данных в реальном времени для предотвращения потери данных. В целях обеспечения отказоустойчивости необходимо использовать виртуализацию серверов с автоматическим масштабированием через Kubernetes или Docker Swarm для динамического распределения нагрузки между узлами, а также интеграцию с облачными сервисами, такими как Amazon Web Services (AWS) или Microsoft Azure, для быстрого увеличения вычислительных мощностей при увеличении трафика. Использование систем RAID для зеркалирования данных также снизит риски потери информации при сбоях на уровне физического оборудования.

12.2 Программные риски

Программные риски могут возникать из-за ошибок в коде, несовместимости библиотек или компонентов системы, что может повлиять на ее работоспособность. Для минимизации этих рисков применяется тщательное тестирование с использованием автоматизированных тестов на базе фреймворков, таких как pytest для Python и Jest для JavaScript, что позволит проверять функциональность каждого компонента системы на этапе разработки. Регулярные обновления и исправления безопасности должны осуществляться через систему непрерывной интеграции и развертывания (CI/CD), настроенную с использованием Jenkins или GitHub Actions, что обеспечит своевременное внедрение исправлений и предотвращение ошибок, вызванных устаревшими версиями библиотек. Все компоненты системы должны быть интегрированы через стандартные API, для которых проводится тестирование с использованием Postman и мок-сервисов для обеспечения их совместимости и бесперебойной работы. Для минимизации воздействия ошибок в алгоритмах машинного обучения, таких как Deep Q-Learning и Bayesian Knowledge Tracing, необходимо внедрить систему мониторинга на основе логирования с использованием ELK Stack (Elasticsearch, Logstash и Kibana) для анализа работы алгоритмов и своевременного выявления аномальных ситуаций. Разработанные модели должны регулярно проверяться с использованием кросс-валидации и тестов на неизведанных данных, а также подвергаться анализу на наличие переобучения или недообучения.

12.3 Законодательные риски

Законодательные риски связаны с возможными изменениями в законодательстве, которые могут повлиять на обработку персональных данных студентов, особенно с учетом требований законодательства о защите данных. Для предотвращения этих рисков система должна быть разработана с соблюдением нормативных требований, таких как Общий регламент защиты данных (GDPR) в Европе и аналогичные законы в других юрисдикциях. Все персональные данные студентов и преподавателей должны храниться в зашифрованном виде с использованием современных криптографических методов, таких как AES-256 для хранения данных и RSA для обмена ключами. Система должна предусматривать аутентификацию пользователей с помощью многофакторной аутентификации (MFA) и защищенного канала передачи данных через HTTPS с использованием SSL/TLS сертификатов для всех внешних соединений. Регулярные аудиты безопасности, включая penetration testing, должны проводиться для выявления уязвимостей в системе. Также необходимо реализовать систему управления доступом на основе ролей (RBAC) для предотвращения несанкционированного доступа к данным, а также обеспечить возможность отслеживания всех операций с персональными данными с помощью журналов доступа. Для соблюдения прав пользователей на доступ, исправление и удаление их персональных данных, должна быть разработана система автоматизированных запросов на изменение данных, соответствующих требованиям GDPR.

12.4 Пользовательские риски

Пользовательские риски могут включать недостаточную квалификацию пользователей для работы с системой, что может привести к неправильной интерпретации результатов или неэффективному использованию функционала. Для снижения этих рисков необходимо разработать интуитивно понятный интерфейс с учетом принципов юзабилити, используя фреймворки, такие как React для динамических интерфейсов и Material-UI для создания стандартных компонентных библиотек. Весь функционал будет описан в доступных и понятных инструкциях, а также обучающих материалах, интегрированных в систему. Для преподавателей и студентов будет предусмотрен раздел с часто задаваемыми вопросами (FAQ) и видеоуроками, а также предоставление инструментов для взаимодействия с системой через чат-ботов, построенных на базе технологий обработки естественного языка (NLP) с использованием Python-библиотек, таких как SpaCy или NLTK. В рамках системы будет реализована система подсказок и валидации ввода для уменьшения ошибок при взаимодействии с интерфейсом, что обеспечит более эффективное

использование функционала. Для студентов будет также предложен модуль для анализа прогресса, который будет включать визуализацию в реальном времени с помощью графиков, построенных с использованием библиотеки D3.js или Chart.js.

13 Стадии и этапы разработки

13.1 Этап подготовки

На данном этапе осуществляется сбор и анализ исходных данных, необходимых для разработки системы, включая определение целей и задач проекта. Анализ осуществляется с использованием стандартных методик, таких как структурированное обследование и анализ требований, направленный на определение функциональных, нефункциональных и технических характеристик системы. Разрабатываются технические требования, в том числе требования к архитектуре системы, функциональным модулям, модели данных и протоколам взаимодействия. Создается предварительная документация, которая включает проектное видение, бизнес-кейсы и проектный план. Выбор технологий и инструментов осуществляется на основе оценки современных технологий, таких как Python, TensorFlow, PyTorch и Django, а также облачных сервисов, таких как Amazon Web Services (AWS) или Google Cloud Platform (GCP). На этом этапе также определяются ключевые компоненты системы, включая фронтенд и бэкэнд, серверное программное обеспечение, базы данных, а также интеграция с системами машинного обучения и анализа данных.

13.2 Этап проектирования

На этапе проектирования разрабатывается архитектура системы, которая включает определение компонентов системы, их взаимодействие и распределение функциональности по слоям. Основное внимание уделяется проектированию серверной инфраструктуры, разработке схемы базы данных и проектированию взаимодействия между моделями машинного обучения, такими как Deep Q-Learning, Bayesian Knowledge Tracing и экспертная система. Архитектурные решения основываются на шаблонах проектирования, таких как микросервисная архитектура, использование API-first подхода и применение RESTful сервисов для взаимодействия компонентов системы. Создаются схемы базы данных, включая таблицы, отношения и типы данных, с использованием реляционных баз данных, таких как PostgreSQL или NoSQL решений, таких как MongoDB. Проектируются интерфейсы для пользователя и администратора, включая веб-интерфейсы с использованием фреймворков, таких как React или Vue.js, и мобильные интерфейсы с использованием фреймворков, таких как Flutter или React Native. Разрабатываются

протоколы взаимодействия между компонентами системы через RESTful и GraphQL API, а также описываются методы аутентификации и авторизации пользователей с использованием OAuth2 или OpenID Connect

13.3 Этап разработки

На данном этапе осуществляется непосредственно программирование всех компонентов системы. Это включает разработку и интеграцию моделей машинного обучения, создание базы данных, реализацию пользовательских и административных интерфейсов, а также настройку серверной части системы. Алгоритмы машинного обучения, такие как Deep Q-Learning, Bayesian Knowledge Tracing и экспертная система, реализуются с использованием библиотеки TensorFlow или PyTorch, в зависимости от специфики задачи. Базовые данные хранятся в реляционной базе данных, такой как PostgreSQL, и оптимизируются с использованием индексирования и кэширования данных. Реализация пользовательских интерфейсов осуществляется с использованием фреймворков, таких как React, с использованием технологий для динамического рендеринга интерфейсов, таких как JavaScript и TypeScript. Внедрение фронтенд-библиотек, таких как Material-UI или Bootstrap, обеспечивает создание интуитивно понятных и функциональных пользовательских интерфейсов. Серверная часть системы на основе Django или Flask отвечает за взаимодействие с базой данных и предоставляет RESTful или GraphQL API для взаимодействия между фронтенд и бэкэнд.

13.4 Этап тестирования

После завершения разработки проводится тестирование всех компонентов системы. Тестирование осуществляется на базе автоматизированных тестов, которые включают функциональное тестирование, интеграционное тестирование, юнит-тестирование и нагрузочное тестирование. Проведение тестов безопасности включает проверку системы на наличие уязвимостей, таких как XSS, CSRF и SQL-инъекции, а также тестирование соответствия системы требованиям по защите персональных данных. Используются стандартные инструменты для тестирования, такие как JUnit для Java и pytest для Python. Нагрузочное тестирование проводится с использованием средств, таких как Apache JMeter или LoadRunner, для проверки производительности системы при максимальной нагрузке. Проверяется также правильность работы моделей машинного обучения, их точность и корректность в рамках поставленных задач.

13.5 Этап внедрения

На этапе внедрения система устанавливается на сервере, который может находиться в облачной среде, такой как AWS, GCP или на локальном сервере. Настройки системы выполняются с учетом требований к инфраструктуре, включая оптимизацию базы данных, конфигурацию веб-сервера, настройку контейнеризации с использованием Docker и настройку контейнеров с контейнерным оркестратором, таким как Kubernetes. Миграция данных проводится через инструменты для импорта и экспорта данных, такие как pg_dump или MySQLDump, в зависимости от используемой базы данных. Обучение пользователей и администраторов системы является важной частью этого этапа. Это включает в себя предоставление документации, обучения пользователя в системе управления, а также предоставление инструкций по использованию системы, интеграции с внешними системами и управлению данными. Внедрение сопровождается мониторингом работы системы и устранением возможных проблем, связанных с совместимостью и производительностью.

13.6 Этап эксплуатации и поддержки

После внедрения система переходит в стадию эксплуатации, где она будет использоваться конечными пользователями. В этот период осуществляется регулярный мониторинг производительности с использованием инструментов, таких как Prometheus и Grafana, а также сбор обратной связи от пользователей с целью выявления ошибок и недочетов в работе системы. Регулярное обновление и поддержка системы включает внедрение новых функций, исправление ошибок и оптимизацию производительности на основе анализа использования системы. На этом этапе также проводятся регулярные аудиты системы безопасности, обновления программного обеспечения, исправление уязвимостей и тестирование производительности с использованием инструментов, таких как Apache Bench и Sysbench. Внесение улучшений основывается на анализе логов ошибок, обратной связи от пользователей и данных мониторинга.

14 Требования к документации

14.1 Описание структуры и содержания руководства пользователя

Руководство пользователя должно предоставлять исчерпывающее описание функциональных возможностей системы, включая все аспекты взаимодействия с конечными пользователями через веб-интерфейс. Документация должна охватывать процесс начальной настройки системы, включая установку серверной части и

конфигурацию базы данных, таких как PostgreSQL или MongoDB, а также настройку клиентских интерфейсов с использованием фреймворков, таких как React или Angular. Важно, чтобы пользователи получили пошаговые инструкции по выполнению начальных тестов на платформе, в том числе интеграцию с моделями машинного обучения, такими как Deep Q-Learning или Bayesian Knowledge Tracing. Также следует предоставить рекомендации по получению персонализированных рекомендаций на основе данных, полученных от системы, и информации о том, как реагировать на подсказки и оценки, генерируемые экспертной системой на основе алгоритмов машинного обучения. Руководство должно включать примеры использования функционала, включая взаимодействие с данными в системе через административные и пользовательские интерфейсы, такие как системы аутентификации и интерфейсы для просмотра результатов тестирования и оценок. Документация должна также предусматривать разделы для устранения типичных ошибок, таких как проблемы с подключением к серверу или неправильные настройки конфигурации, а также предоставлять пошаговые инструкции по их устранению. Особое внимание должно быть уделено разделу обновлений системы, включающему инструкции по применению новых версий с использованием системы управления версиями, такой как Git, и платформы CI/CD, например, Jenkins или GitLab CI. Раздел для обращения за поддержкой должен включать контактные данные технической поддержки, а также процедуры для отправки запросов и получения решения проблем.

14.2 Документация по API для взаимодействия компонентов

Документация по API должна содержать детальное описание всех доступных интерфейсов для взаимодействия компонентов системы, включая спецификации RESTful API или GraphQL. В описании каждого метода API должны быть указаны входные и выходные параметры, включая типы данных (например, JSON или XML), формат запросов и ответов, а также механизмы аутентификации и авторизации, такие как использование OAuth2 для защиты API. Документация должна детально объяснять, как компоненты системы обмениваются данными, включая описание методов взаимодействия с базой данных, например, использование SQL-запросов в PostgreSQL или NoSQL-запросов в MongoDB для извлечения, вставки или обновления данных. Взаимодействие с моделями машинного обучения, реализованными в TensorFlow или PyTorch, должно быть представлено через четко определенные API-методы, которые позволяют интеграцию модели в процесс обработки данных и предоставления рекомендаций. Также следует описать способы обращения к внешним сервисам через API, такие как системы управления пользователями, почтовые сервисы или внешние базы данных. В документации должны

быть приведены примеры запросов и ответов для каждого метода API, а также возможные ошибки, такие как ошибки валидации данных или проблемы с подключением к серверу, с соответствующими рекомендациями по их обработке.

15 Технико-экономические показатели

15.1 Ожидаемая производительность системы

Ожидаемая производительность системы будет определяться расчетами на основе предполагаемой нагрузки и характеристик вычислительных операций. Время отклика системы на запросы пользователей будет критично для работы с веб-интерфейсом, и оно должно быть оптимизировано для минимизации задержек. Время отклика будет измеряться с учетом использования серверной части, реализованной на базе облачных платформ, таких как AWS или Google Cloud, что позволит масштабировать ресурсы в зависимости от нагрузки. Производительность системы также будет зависеть от скорости обработки тестов, что включат как оценку базовых заданий, так и сложных аналитических операций с использованием моделей машинного обучения, таких как Deep Q-Learning и Bayesian Knowledge Tracing. Для эффективной работы системы необходимо обеспечить быструю генерацию рекомендаций и оценок, что потребует интеграции алгоритмов с базой данных в реальном времени, используя, например, PostgreSQL или MongoDB. Важным элементом производительности является время, необходимое для обучения моделей машинного обучения, что будет напрямую зависеть от объема данных, сложности моделей и мощности серверного оборудования. Для обучения сложных моделей потребуется использование высокопроизводительных вычислительных ресурсов, таких как GPU на базе NVIDIA или облачные вычисления с использованием TensorFlow или PyTorch. Важно также учитывать потребность в адаптивности системы, которая должна корректировать рекомендации в ответ на изменение данных о пользователе, поддерживая при этом минимальные задержки на каждом этапе обработки данных.

15.2 Объем данных и требования к серверу

Объем данных, который будет обрабатываться системой, будет определяться числом пользователей, частотой выполнения тестов и объемом задач, решаемых студентами. Для работы системы потребуется хранить данные о студентах, их результатах тестирования, выполненных заданиях и метаданных, а также данные для моделей машинного обучения, такие как уровень знаний студентов и результаты рекомендаций. Для этого сервер должен быть оснащен мощной многозадачной серверной инфраструктурой с многоядерным

процессором, например, Intel Xeon или AMD EPYC, что позволит эффективно обрабатывать параллельные вычисления для обработки данных и обучения моделей. Оперативная память должна быть достаточной для выполнения сложных вычислительных операций, включая параллельные вычисления и анализ больших объемов данных, что потребует от 32 ГБ до 128 ГБ в зависимости от размера и сложности обучаемых моделей. Система хранения данных должна включать быстрые SSD-диски для хранения активных данных и надежные системы резервного копирования, а также предусматривать масштабируемость для поддержания роста базы пользователей и объема данных. При увеличении объема данных и количества пользователей необходимо обеспечить масштабирование инфраструктуры, что может быть достигнуто с помощью распределенных вычислительных систем и кластерных решений на базе Kubernetes или Docker. Важно также учитывать использование решений для автоматического резервного копирования и восстановления данных для обеспечения отказоустойчивости системы.