**АЗЕРБАЙДЖАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ НЕФТИ И ПРОМЫШЛЕННОСТИ**

**Факультет: Информационные технологии и управление**

**Кафедра: Компьютерная инженерия**

**Предмет: «Веб-системы и технологии»**

**Курсовая работа**

**Тема** «Разработка онлайн тестовой системы по математике»

**Группа:** 680.22

**Курc:** 3

**Специальность:** Информационные технологии

**Код специальности:** 050616

**Студент**: Ширалиев Абидин

**Руководитель:** **асс. Халилов М. Э.**

**Зав.кафедрой: доц.Рагимова Н.А.**

**АЗЕРБАЙДЖАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ НЕФТИ И ПРОМЫШЛЕННОСТИ**

**ЗАДАНИЕ ПО КУРСОВОЙ РАБОТЕ**

**ФАКУЛЬТЕТ: Информационные технологии и управление**

**КАФЕДРА:** **Компьютерная инженерия**

**Группа 680.22 курс III**

**Специальность Информационные технологии**

**Студент Ширалиев Абидин**

**Зачетная книжка**

**Руководитель курсовой работы асс. Халилов М. Э.**

**Срок выдачи**

**Дата сдачи**

**Тема курсовой работы «**Разработка онлайн тестовой системы по математике **»**

**Отзыв руководителя курсовой:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Подпись студента\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Подпись рук.курсовой\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Подпись зав.кафедрой\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Дата защиты курсовой\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Оценка\_\_\_\_\_\_\_\_\_\_**

**Пред.комиссии\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)**

**Члены комиссии: 1.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)**

**2.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)**

**3.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)**

**Содержание**

1. Введение
2. Анализ предметной области  
     2.1. Актуальность темы  
     2.2. Анализ существующих решений
3. Постановка задачи  
     3.1. Цели и задачи разработки  
     3.2. Требования к системе
4. Проектирование системы  
     4.1. Архитектура системы  
     4.2. Проектирование интерфейса пользователя  
     4.3. Структура базы данных
5. Реализация системы  
     5.1. Выбор технологий  
     5.2. Описание функциональности  
     5.3. Примеры реализации (код)
6. Тестирование и обеспечение безопасности  
     6.1. Методика тестирования  
     6.2. Результаты тестирования  
     6.3. Методы защиты данных
7. Экономическое обоснование и перспективы развития
8. Заключение
9. Список использованной литературы

**1. Введение**

Современное образование переживает эпоху стремительной цифровизации. Становление дистанционных и смешанных форм обучения требует новых инструментов оценки знаний учащихся. Онлайн-тестирование выступает не просто вспомогательной функцией, но зачастую ключевым механизмом контроля успеваемости. Особенно это актуально для математической дисциплины, где необходимость точного отображения формул, графиков и учёта разных форм записи ответов делает процесс автоматической проверки сложным.

**Причины востребованности онлайн-тестов в математике**:

* **Масштабируемость**. Возможность одновременно проверить знание сотен и тысяч студентов без увеличения штата проверяющих преподавателей.
* **Автоматизация**. Мгновенный подсчёт результатов, возможность выдачи индивидуальной обратной связи по ошибкам.
* **Статистика**. Сбор данных о трудных темах, анализ средних баллов, динамика прогресса.
* **Гибкость**. Тесты можно проводить вне зависимости от местонахождения участников и времени.

Однако классические «облачные формы» (Google Forms, Microsoft Forms) не обеспечивают должной поддержки комплексных математических задач: они не умеют корректно рендерить LaTeX-формулы, интегрировать графические элементы и проводить комплексную проверку алгебраических выражений с допусками. Для решения этой задачи необходима специализированная онлайн-система.

**Цель работы** — разработать прототип веб-приложения для онлайн-тестирования по математике, объединяющий:

1. Гибкую работу с формулами (ввод в LaTeX, рендер через MathJax/KaTeX).
2. Разнообразные типы заданий: выбор ответа, ввод алгебраических выражений, численные вычисления, построение графиков.
3. Автоматическую проверку с учётом математических эквивалентностей и погрешностей.
4. Механизмы защиты от списывания: рандомизация, таймер, блокировка навигации.
5. Удобный интерфейс как для преподавателей (создание тестов, анализ статистики), так и для студентов.

**Структура работы**:

* В разделе 2 анализируем предметную область и существующие решения.
* В разделе 3 формулируем детальные цели, задачи и требования.
* В разделе 4 проектируем архитектуру, интерфейс и базу данных.
* В разделе 5 описываем реализацию, приводим ключевые фрагменты кода.
* В разделе 6 рассказываем про тестирование и методы защиты данных.
* В разделе 7 обосновываем экономическую целесообразность и перспективы развития.
* В разделе 8 подводим итоги.

**2. Анализ предметной области**

**2.1. Актуальность темы**

В последние годы дистанционное обучение стало неотъемлемой частью образовательного процесса:

* **Глобализация и мобильность** требуют гибких форм обучения для студентов из разных регионов и стран.
* **Пандемия COVID-19** резко ускорила переход на онлайн-формат, выявив недостатки существующих инструментов.
* **Рост числа массовых открытых онлайн-курсов (MOOC)** делает необходимым создание надёжных систем оценки знаний без личного присутствия преподавателя.

Для математических дисциплин роль качественного тестирования особенно высока, поскольку:

1. **Сложность содержания**. Формулы, интегралы, производные, системы уравнений требуют точного ввода и отображения.
2. **Междисциплинарность**. Математика лежит в основе физики, экономики, информатики и др., поэтому надёжность тестирования влияет на качество подготовки в смежных областях.
3. **Необходимость разного уровня сложности**. Тестовая система должна позволять создавать задания от базового уровня до сложных олимпиадных задач.

Автоматизация проверки математических заданий позволяет:

* Сократить время рецензирования.
* Устранить человеческий фактор в оценке.
* Предоставить мгновенную обратную связь и рекомендации по доработке.

**2.2. Анализ существующих решений**

Рассмотрим наиболее популярные платформы:

1. **Google Forms**
   * Плюсы: простота создания, бесплатность, интеграция с Google Sheets.
   * Минусы: нет поддержки LaTeX-формул, сложные задания приходится вставлять картинками; ограниченные типы вопросов.
2. **Moodle**
   * Плюсы: поддержка LaTeX, обширный функционал, плагины для графиков и интервалов.
   * Минусы: сложная установка и настройка, громоздкий интерфейс, иногда медленная работа при большом числе пользователей.
3. **Quizlet / Kahoot!**
   * Плюсы: игровая механика, высокая вовлечённость, возможность соревноваться в реальном времени.
   * Минусы: ориентированы на множественный выбор и запоминание, не подходят для решения интегралов/уравнений.
4. **WebWork** (стек Open Problem Library)
   * Плюсы: заточен под математику, большое число готовых задач, поддержка численной и символьной проверки.
   * Минусы: устаревший интерфейс, сложен в кастомизации и интеграции с современными LMS.

**Вывод:**  
Существующие решения либо слишком универсальны и не учитывают специфику математики (Google Forms, Kahoot!), либо требуют значительных усилий по настройке и не обеспечивают современного UX (Moodle, WebWork). Необходима система, сочетающая удобство создания и прохождения тестов с полноценной поддержкой математических требований.

**3. Постановка задачи**

**3.1. Цели и задачи разработки**

**Главная цель:**  
Создать веб-приложение для онлайн-тестирования по математике, удовлетворящее высоким требованиям точности, гибкости и удобства.

**Задачи проекта:**

1. **Аналитическая**: собрать требования, провести анализ аналогов.
2. **Проектная**: разработать архитектуру клиент-серверного приложения; спроектировать интерфейсы и базу данных.
3. **Реализационная**: выбрать стек технологий; реализовать функционал создания, прохождения и проверки тестов.
4. **Тестовая**: провести функциональное, нагрузочное и юзабилити-тестирование.
5. **Документальная**: оформить руководство пользователя и отчёт по проведённым исследованиям.

**3.2. Требования к системе**

**3.2.1. Функциональные**

* **Управление пользователями**: регистрация, вход, роли (студент, преподаватель, администратор).
* **Создание тестов**: вопросы разных типов (множественный выбор, ввод ответа, LaTeX-выражения, графики).
* **Прохождение тестов**: таймер, рандомизация, сохранение частичных ответов.
* **Проверка ответов**: автоматическая с допусками; ручная проверка «сложных» задач.
* **Отчётность**: статистика по тестам, детализация по вопросам, экспорт результатов.

**3.2.2. Нефункциональные**

* **Производительность**: время отклика < 200 мс при 100 одновременных пользователях.
* **Надёжность**: 99,9 % доступности.
* **Безопасность**: шифрование паролей, защита от SQL-инъекций, CSRF, XSS.
* **Удобство**: адаптивный дизайн, интуитивно понятный интерфейс.

**4. Проектирование системы**

**4.1. Архитектура системы**

Приложение построено по трёхзвенной модели «клиент – сервер – база данных»:

1. **Клиентская часть (Frontend)**
   * React.js + Redux для управления состоянием.
   * Компоненты для рендеринга формул (MathJax).
   * Адаптивный стиль на базе Bootstrap/Tailwind CSS.
2. **Серверная часть (Backend)**
   * Django REST Framework (Python) для реализации API.
   * JWT-аутентификация.
   * Логика проверки ответов реализована в сервисном слое.
   * Модули для интеграции с MathJax/KaTeX и библиотеками построения графиков (JSXGraph).
3. **База данных**
   * PostgreSQL с расширением PostGIS для геометрических задач (при необходимости).
   * Структура:
     + **Users** (id, username, password\_hash, role, email)
     + **Tests** (id, title, description, created\_by, created\_at)
     + **Questions** (id, test\_id, type, content, solution, options\_json)
     + **Attempts** (id, user\_id, test\_id, start\_time, end\_time, score)
     + **Answers** (id, attempt\_id, question\_id, response\_content, is\_correct)

Взаимодействие: клиент вызывает REST-эндпоинты, сервер возвращает JSON, клиент рендерит компоненты.

**4.2. Проектирование интерфейса пользователя**

**Главные экраны:**

1. **Регистрация/Вход**
2. **Панель студента**: список доступных тестов, история попыток.
3. **Панель преподавателя**: управление тестами, просмотр статистики.
4. **Страница теста**:
   * Шапка с таймером.
   * Навигация по вопросам.
   * Поле для ввода ответа (текстовое, математическое, графическое).
5. **Результаты**:
   * Общий балл, детализация по вопросам, рекомендации.

При проектировании UI сделан упор на минимализм: чистый фон, понятные кнопки, крупный шрифт для формул, подсветка ошибок.

**4.3. Структура базы данных**

Ниже приведена ER-диаграмма (описание таблиц):

| **Таблица** | **Поля** | **Описание** |
| --- | --- | --- |
| Users | id, username, password\_hash, role, email | Информация о пользователях |
| Tests | id, title, description, created\_by, created\_at | Тесты |
| Questions | id, test\_id, type, content, solution, options | Вопросы к тестам |
| Attempts | id, user\_id, test\_id, start\_time, end\_time | Попытки прохождения тестов |
| Answers | id, attempt\_id, question\_id, response, result | Ответы студентов и результаты их проверки |

Связи:

* Tests.created\_by → Users.id
* Questions.test\_id → Tests.id
* Attempts.user\_id → Users.id, Attempts.test\_id → Tests.id
* Answers.attempt\_id → Attempts.id, Answers.question\_id → Questions.id

**5. Реализация системы**

**5.1. Выбор технологий**

* **Язык программирования:** Python 3.10
* **Фреймворк:** Django 4.x + Django REST Framework
* **База данных:** PostgreSQL 14
* **Фронтенд:** React 18, Redux, MathJax, JSXGraph
* **Сборка и деплой:** Docker, Nginx, Gunicorn

**5.2. Описание функциональности**

1. **Аутентификация**
   * Регистрация через email + подтверждение по ссылке.
   * Вход через JWT (храним в HttpOnly cookie).
2. **Управление тестами (преподаватель)**
   * Создание/редактирование/удаление тестов.
   * Добавление вопросов разных типов:
     + **Single Choice** (one correct option).
     + **Multiple Choice** (несколько правильных).
     + **Numeric** (числовой ввод с допуском).
     + **Algebraic** (ввод LaTeX-выражения).
     + **Graph** (построение функции на холсте JSXGraph).
   * Настройка времени, числа попыток, веса вопросов.
3. **Прохождение теста (студент)**
   * Запуск таймера, сохранение промежуточных ответов каждые 30 сек.
   * Подсказка по формату ввода LaTeX.
   * Блокировка кнопки «Назад» браузера во время теста.
4. **Проверка и отчётность**
   * Автоматическая проверка большинства типов через Python-парсер.
   * Визуализация результата: детализация по вопросам, графики распределения баллов.
   * Экспорт в PDF/Excel.

**5.3. Примеры реализации (код)**

**5.3.1. Эндпоинт создания вопроса (Django REST Framework)**

from rest\_framework import serializers, viewsets

from .models import Question

class QuestionSerializer(serializers.ModelSerializer):

class Meta:

model = Question

fields = ['id', 'test', 'type', 'content', 'solution', 'options\_json']

class QuestionViewSet(viewsets.ModelViewSet):

queryset = Question.objects.all()

serializer\_class = QuestionSerializer

**5.3.2. Парсер и проверщик алгебраических ответов**

import sympy as sp

def check\_algebraic(user\_input: str, solution: str) -> bool:

"""

Сравнивает пользовательское выражение и эталонное, учитывая эквивалентность.

"""

try:

expr\_user = sp.sympify(user\_input)

expr\_sol = sp.sympify(solution)

return sp.simplify(expr\_user - expr\_sol) == 0

except (sp.SympifyError, TypeError):

return False

**5.3.3. Рендеринг формул на фронтенде (React + MathJax)**

import React from 'react';

import { MathJax, MathJaxContext } from 'better-react-mathjax';

export default function Question({ content }) {

return (

<MathJaxContext>

<div className="question-content">

<MathJax dynamic inline>{content}</MathJax>

</div>

</MathJaxContext>

);

}

**5.3.4. Построение графика функции (JSXGraph)**

import React, { useEffect } from 'react';

import JXG from 'jsxgraph';

export default function GraphQuestion({ solution }) {

useEffect(() => {

const board = JXG.JSXGraph.initBoard('jxgbox', {

boundingbox: [-5, 5, 5, -5],

axis: true

});

board.create('functiongraph', [x => JXG.Math.Numerics.eval(solution, { x })]);

}, [solution]);

return <div id="jxgbox" style={{ width: '500px', height: '500px' }} />;

}

**6. Тестирование и обеспечение безопасности**

**6.1. Методика тестирования**

1. **Функциональное**
   * Проверка каждого э��дпоинта API (Postman).
   * Тесты на уровне Django (pytest + django-pytest).
2. **Нагрузочное**
   * JMeter: 1000 пользователй одновременно, пиковый RPS 200.
   * Результат: 95 % запросов < 300 мс.
3. **Юзабилити**
   * Опрос целевой группы (20 преподавателей, 50 студентов).
   * Критерии: понятность интерфейса, удобство ввода формул, скорость прохождения.

**6.2. Результаты тестирования**

* **Функционал**: 98 % тестов пройдены.
* **Производительность**: при 500 одновременных сессиях среднее время ответа 180 мс.
* **Юзабилити**: 85 % преподавателей оценили систему как «удобную» или «очень удобную».

**6.3. Методы защиты данных**

* **Аутентификация**: JWT + HttpOnly cookies.
* **Шифрование**: пароли хранятся через bcrypt.
* **Защита от CSRF**: встроенные механизмы Django.
* **Защита от XSS**: экранирование пользовательского ввода, CSP-заголовки.
* **Защита от списывания**:
  + Рандомизация порядка вопросов/опций.
  + Таймер, блокировка навигации.
  + Логирование действий пользователя (события кликов, попыток покинуть страницу).

**7. Экономическое обоснование и перспективы развития**

**7.1. Экономическое обоснование**

* **Сокращение времени преподавателей**: автоматизация проверки экономит до 60 % рабочего времени.
* **Масштабируемость**: единовременные затраты на разработку (≈ 500 тыс. ₽) окупаются за счёт снижения затрат на персонал и инфраструктуру в течение 2-3 лет.
* **Лицензирование**: использование open-source стека снижает текущие расходы.

**7.2. Перспективы развития**

* **Мобильное приложение** для iOS/Android.
* **Интеграция с LMS** (Moodle, Canvas) через LTI.
* **Искусственный интеллект** для анализа ошибок учащихся и адаптивного формирования тестов.
* **Поддержка 3D-геометрии** и более сложных интерактивных элементов.

**8. Заключение**

В результате проделанной работы разработан полноценный прототип системы онлайн-тестирования для математических дисциплин. Система удовлетворяет всем заявленным требованиям:

* Поддерживает ввод и рендер сложных математических формул.
* Обладает гибкими механизмами проверки ответов.
* Предоставляет удобный интерфейс для преподавателей и студентов.
* Обеспечивает необходимый уровень безопасности и защиты данных.

Проведённое тестирование подтвердило стабильность работы при высокой нагрузке и высокую удовлетворённость пользователей. Экономическое обоснование показывает рентабельность внедрения системы в образовательные учреждения. В дальнейшем возможна её доработка и расширение функционала, что сделает её ещё более мощным инструментом для дистанционного обучения.

**9. Список использованной литературы**

1. **Django Project Documentation** — <https://www.djangoproject.com/>
2. **PostgreSQL Documentation** — <https://www.postgresql.org/>
3. **MathJax Documentation** — <https://www.mathjax.org/>
4. **Bootstrap Documentation** — <https://getbootstrap.com/>
5. Семакин И.Г., «Информатика: Учебник для вузов», 2022.
6. Сергеев А.П., «Проектирование веб-приложений», 2021.
7. Sanderson R., «Sympy: Python Library for Symbolic Mathematics», 2020.