МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**
Кафедра математической кибернетики и компьютерных наук

РАЗРАБОТКА СИСТЕМЫ АДАПТИВНОГО ТЕСТИРОВАНИЯ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

Студентки 4 курса 451 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Гамаюновой Виктории Олеговны

Научный руководитель
к. пед. н., доцент _____ В. А. Векслер

Заведующий кафедрой
к. ф.-м. н., доцент _____ С. В. Миронов

Саратов 2022

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время существует такой метод контроля результатов обучения как компьютерное адаптивное тестирование. Данный вид тестирования основан на том, что каждый последующий вопрос теста выбирается на основании предыдущих ответов испытуемого, его текущей оценки и иных параметров.

Адаптивное тестирование является эффективным методом оценки знаний, так как различные алгоритмы генерации теста позволяют ликвидировать недостатки классических способов оценки знаний.

Актуальность данной работы состоит в том, что использование систем адаптивного тестирования позволяет получать более объективные результаты, чем при прохождении классического тестирования, а также создавать индивидуальный маршрут испытуемого при прохождении теста в зависимости от степени его подготовленности. В связи с этим, велика потребность в таких системах, которые позволяют повысить качество оценки результатов обучения.

Объектом исследования является система адаптивного тестирования.

Предметом исследования являются инструменты и способы реализации системы адаптивного тестирования.

Цель настоящей работы – разработка системы адаптивного тестирования средствами языка программирования Python.

В связи с поставленной целью, основными **задачами** являются:

- изучить учебную и научную литературу по теме адаптивного тестирования;
- изучить существующие алгоритмы адаптивного тестирования;
- применить в разработке системы возможности языка программирования Python.

Методологические основы разработки системы адаптивного тестирования представлены в работах А. А. Малыгина, В. А. Глебова, А. С. Фарфорова, С. П. Дударова.

Структура и объём работы. Бакалаврская работа состоит из введения, двух разделов, заключения, списка использованных источников и двух приложений. Общий объем работы – 72 страницы, из них 43 страницы – основное содержание, включая 28 рисунков, список использованных источников информации – 21 наименование.

1 Компьютерное адаптивное тестирование

В данном разделе рассматриваются теоретические основы адаптивного тестирования.

1.1 Понятие и история развития адаптивного тестирования

Появление такого способа оценки знаний как адаптивное тестирование было обусловлено стремлением к уменьшению количества решаемых заданий, сокращению затрачиваемого на процесс тестирования времени и повышению точности оценки, выставляемой испытуемому.

Обычно выделяют три основных периода.

В первом периоде идеи адаптивного тестирования поддерживались в основном не учеными и исследователями, а практиками, и заключались в интуитивных попытках добавить к процессам генерации классических тестов элементы индивидуализации. Во втором – классическая теория тестов начала уступать место новой теории IRT (Item Response Theory), или современной теории тестов. На её основе активно разрабатывались теоретические и технологические основы современных методов генерации адаптивных тестов. Главной чертой третьего периода стало повсеместное использование современных технологий в образовательных целях, что позволяет говорить о компьютерном адаптивном тестировании как об эффективном процессе генерации и предъявления адаптивных тестов. [1]

1.2 Классификация стратегий адаптивного тестирования

Стратегии адаптивного тестирования можно разделить на две группы – *двухшаговые* и *многошаговые*.

Двухшаговая стратегия предполагает предварительное прохождения входного теста для определения начального уровня знаний, после чего организуется адаптивное тестирование испытуемого. В многошаговых стратегиях во время тестирования испытуемый проходит по индивидуальному маршруту. Такие стратегии в свою очередь делятся на *фиксировано-ветвящиеся* и *варьирующе-ветвящиеся* в зависимости от способа конструирования теста.

В фиксировано-ветвящихся стратегиях для всех испытуемых используется один набор заданий адаптивного теста, который зафиксирован на оси трудности с определенным расстоянием между ними.

В варьирующе-ветвящейся стратегии очередное задание выбирается из банка заданий согласно некоторому алгоритму. На каждом шаге также переоценивается уровень подготовленности, что является преимуществом перед другими стратегиями. [2]

1.3 Алгоритмы адаптивного тестирования

Все алгоритмы адаптивного тестирования состоят из основных этапов:

— инициализация процедуры тестирования;
— цикл предоставления задания и предъявление испытуемым ответа;
— завершение тестирования.

Как правило, различия обнаруживаются на втором этапе. [3] При организации адаптивного тестирования могут учитываться такие критерии как время испытания, количество предъявляемых заданий, точность измерения результата.

Данный раздел содержит описание трёх алгоритмов адаптивного тестирования.

Первой рассматривается пирамидальная стратегия. Вопросы в данном алгоритме схематично расположены в виде пирамиды, начинается тестирование со среднего уровня сложности. При условии получения правильного ответа вопросы усложняются, иначе – предоставляются более легкие. Условие окончания тестирования – выполнение заранее обозначенного количества заданий. До начала тестирования задается количество стадий и рассчитывается минимальное количество вопросов каждого уровня сложности.

Еще один алгоритм адаптивного тестирования - стратегия половинного деления. Работа данной стратегии устроена таким образом, что испытуемому не будут предлагаться задания, результат решения которых можно предсказать с высокой долей вероятности. [4]

Также рассматривается алгоритм, основанный на теории нечетких множеств, который позволяет более качественно интерпретировать результаты тестирования. [5]

1.4 Преимущества и недостатки адаптивного тестирования

Среди преимуществ адаптивного тестирования [6] были отмечены следующие:

— подбор заданий подходящей для испытуемого сложности;
— сокращение времени тестирования и количества вопросов;

— отсутствие влияния субъективного мнения на выставление оценки.

Несмотря на все преимущества, адаптивное тестирование также имеет свои недостатки:

— невозможность пропустить задание и вернуться к нему позже;
— невозможность изменить ответы на предыдущие задания;
— большие затраты на создание компьютерного адаптивного теста по сравнению с классическим тестом.

2 Программная реализация системы адаптивного тестирования на Python

Данный раздел содержит в себе описание программной реализации системы адаптивного тестирования средствами языка программирования Python с использованием Flask и его расширений, а также некоторых дополнительных библиотек.

В качестве алгоритма адаптивного тестирования была выбрана пирамидальная стратегия.

2.1 Используемые для разработки технологии

Система адаптивного тестирования была реализована в формате web-приложения на языке программирования Python [7].

Для создания приложений использовался микрофреймворк Flask, расширение flask-sqlalchemy для работы с базой данных, шаблонизатор Jinja2 для конструирования страниц. [8] Страницы описаны с использованием HTML и CSS.

Также используются следующие расширения:

— Flask-Login – предоставляет функционал, с помощью которого было реализовано разграничение доступа и управление авторизацией пользователя.

— WTForms – позволяет генерировать формы, проверять их содержимое и предлагает защиту от CSRF, для этого нужно установить секретный ключ.

Используемая СУБД – PostgreSQL.

2.2 Создание моделей таблиц базы данных

В файле «db_models.py» описаны классы-модели, соответствующие всем таблицам базы данных. При описании классов-моделей каждому полю указывается тип данных. Аналогичные типы данных имеются в базе данных. Задаются параметры первичного ключа, уникальности поля, возможность принимать значение NULL, а также внешние ключи.

Класс Users представляет собой описание таблицы «users», которая хранит следующую информацию о пользователе: имя пользователя, его Email, хэш пароля, идентификатор роли.

Класс Roles хранит список возможных ролей пользователя, а именно «Учитель» или «Ученик». В базе данных соответствует таблице «roles».

Следующие модели – это класс Tests для таблицы «tests» и класс Topics таблицы «topics». Tests содержит основную информацию о тесте, такую как название теста, его описание, количество стадий для реализации пирамидального алгоритма, а также идентификаторы автора теста и его категории. Данные этой таблицы используются при отображении на странице описания каждого теста. Класс Topics соответствует таблице названий категорий тестов. Данная таблица необходима для создания удобного интерфейса пользователя, который сможет выбрать тесты по интересующей его тематике.

В каждом тесте должен быть определенный набор вопросов. Для хранения данных об этих вопросах в базе данных существует таблица «questions», а в программе соответствующая ей модель – класс Questions. Данная таблица хранит текст вопроса, идентификатор теста, которому он принадлежит, уровень сложности и идентификатор типа вопроса. Типы вопросов, а точнее их названия, находятся в таблице «type_questions», соответствующая модель – класс TypeQuestions.

В приложении предполагается, что вопросы теста имеют хотя бы два ответа, среди которых может быть один или несколько правильных. Класс Answers включает в себя текст ответа, идентификатор вопроса, которому он принадлежит, и флаг корректности данного ответа. Ответы на вопросы хранятся в базе данных в таблице «answers».

В программе реализован алгоритм адаптивного тестирования. Поэтому, необходимо хранить результаты прохождения тестов пользователями. Для этого в базе существуют таблицы «results» и «detail_results». Соответствующий первой таблице класс Results описывает результаты тестирования в целом, хранит идентификаторы теста и ученика, который его прошел, дату прохождения теста. Также хранит набранные баллы и оценку по 5-балльной шкале.

Таблица «detail_results» содержит более подробную информацию о каждом прохождении теста. В процессе тестирования фиксируется предоставленный вопрос и баллы, полученные пользователем за ответ на него. Таблица хранит идентификатор результата из таблицы «results», идентификатор вопроса и количество баллов, набранных за этот вопрос. Этой таблице соответствует класс DetailResults.

По всем описанным выше моделям были сгенерированы таблицы в базе данных и все внешние ключи.

2.3 Создание приложения и задание конфигурации

Для того чтобы создать web-приложение, нужно создать экземпляр класса Flask. Также необходимо задать конфигурацию для этого приложения, она хранится в атрибуте config объекта Flask. [9] После чего, необходимо задать параметры конфигурации приложения.

Для подключения к базе данных необходимо указать строку подключения. [10] Данную строку можно получить из переменной окружения. Для корректной работы модуля Flask-Login необходим секретный ключ, который также хранится в переменной среды.

В параметр конфигурации «PERMANENT_SESSION_LIFETIME» записано время, в течение которого пользователь будет авторизован в браузере при отсутствии активности. Например, закрыв браузер более чем на тридцать минут, пользователю придется авторизоваться заново при следующем посещении сайта. При каждом взаимодействии с приложением время хранения пользователя в сессии сбрасывается снова до тридцати минут.

При описании моделей базы данных был создан экземпляр db класса SQLAlchemy, через который и осуществляется работа с базой данных. Необходимо передать ему ссылку на текущее приложение, с которым он будет связан.

Для работы с модулем Flask-Login нужно создать экземпляр класса LoginManage, который будет управлять процессом авторизации.

Для каждого URL-адреса имеется функция представления – в приложении Flask это обычные функции, которые возвращают ответ для данного URL. Обычно у представлений присутствует декоратор @app.route(), который определяет URL-адрес. [11] Также, здесь указываются методы запроса. Функция render_template() отрисовывает шаблон страницы и возвращает его клиенту в качестве ответа. В шаблон можно передавать параметры, которые используются для создания разметки с помощью шаблонизатора.

2.4 Регистрация и авторизация

Для работы с формами сайта используется модуль WTForms. [12] Данный модуль позволяет сгенерировать форму, заполнять ее данными и проверять введенные пользователем значения. В данный модуль встроена защита от CSRF. Созданы классы LoginForm и RegisterForm, расширяющие базовый класс FlaskForm, а поля формы являются переменными этих классов. Для каждого

поля имеется список валидаторов, с помощью которых происходит проверка введенных данных. Экземпляры данных классов далее передаются шаблону как параметр, после чего из них будут сгенерированы формы входа и регистрации, для которых уже не требуется задавать собственную проверку полей. [13]

Если обращение к странице регистрации было выполнено с помощью GET запроса, то пользователю возвращается страница с пустой сгенерированной формой регистрации. Иначе, в базе данных проверяется, существует ли пользователь с указанным Email. [14] Если нет, то создается новый экземпляр класса User, данные из формы считываются в него, при этом, пароль хэшируется. После чего созданный объект добавляется в базу данных [15] и пользователь перенаправляется на страницу авторизации. В случае успешной регистрации или возникновения ошибок выводится соответствующее сообщение. [16]

Сначала происходит проверка, авторизован ли пользователь, пытающийся зайти на страницу авторизации. Если да, то он перенаправляется на страницу профиля. Если не авторизован и запрос типа GET, то пользователю возвращается страница с пустой сгенерированной формой авторизации. После ввода данных на сервер приходит POST запрос. В этом случае производится попытка получить из базы данных пользователя с указанным Email. Если такой пользователь есть, то сравнивается хэш пароля пользователя в базе данных и хэш введенного в форму пароля. [17]

Если пара Email и пароль верные, то создается экземпляр класса UserLogin для проверки пользователей с помощью Flask-Login. [18] В данном случае он описывает авторизованного пользователя.

Функция login_user, используемая в функции представления для страницы авторизации, заносит информацию об авторизованном пользователе в сессию. Далее, эта информация передается вместе с каждым запросом, что позволяет получить из сессии данные о пользователе, доступные в глобальной переменной current_user. Также модуль предоставляет функцию logout_user для выхода из профиля, которая удаляет из сессии данные о пользователе.

Когда пользователь успешно авторизован, он перенаправляется на страницу профиля. Ученик в своем профиле может посмотреть краткую статистику по всем тестам: название теста, свою лучшую оценку и лучший балл. Клик на блок с тестом ведет на страницу данного теста.

2.5 Просмотр тестов

При клике на какую-либо категорию, появляется страница с тестами из этой категории и их краткими описаниями, либо сообщение о том, что таких тестов нет. Данная информация доступна любому пользователю сайта, даже не авторизованному.

Если пользователь – учитель, то ему также доступна категория «Мои тесты». Она включает в себя набор тестов, созданных текущим пользователем, а также ссылку «Создать тест».

2.6 Создание адаптивного теста учителем

Пользователь, который зарегистрирован как учитель, имеет возможность создавать тесты. Для этого нужно в категории «Мои тесты» перейти по ссылке «Создать тест». Создание теста проходит в несколько этапов. Сначала необходимо заполнить общую информацию о тесте. Категорию можно выбрать из списка или создать новую.

Сначала проверяется корректность заполнения поля имени и количества стадий тестирования. Категория по умолчанию NULL, то есть тест без категории, а описание может остаться пустым.

Далее, если в базе найден тест с таким названием, то выводится сообщение об этом. Иначе, создается новый объект Tests с параметрами, полученными от пользователя и запись добавляется в базу данных. После этого происходит перенаправление на страницу создания вопросов. [19]

По количеству стадий теста рассчитывается число вопросов, необходимое для каждого уровня сложности и формирует словарь вида {уровень : количество вопросов}. Далее этот словарь передается в шаблон «create_questions.html», в котором генерируется форма для ввода необходимого количества вопросов и выбора его типа: вопрос с одним правильным ответом или множественным выбором.

После получения ответа от пользователя, все вопросы добавляются в базу данных, а пользователь перенаправляется на страницу настройки количества ответов на вопросы. [20] На этой странице пользователю необходимо настроить количество ответов для каждого из добавленных на предыдущем этапе вопроса.

После заполнения указания количества ответов пользователь попадает на страницу создания ответов к тесту. Здесь имеются поля для ввода текста ответа

и возможность выбрать правильные. В группе ответов с одним правильным вариантом по умолчанию последний ответ отмечен как верный, а в группе с множественным выбором - все.

После отправки этих данных пользователь перенаправляется в раздел «Мои тесты». Только что добавленный тест появляется в данном разделе.

Если пользователь кликнет на тест, который он создал, то сможет ознакомиться со списком вопросов и статистикой прохождения данного теста учениками. Для этого необходимо кликнуть на нужный заголовок и необходимая информация отобразится на странице. [21]

Вопросы теста отображаются в виде таблицы, включают в себя текст вопроса, его тип и уровень сложности. Список вопросов отсортирован по уровню сложности. Под списком вопросов находится ссылка «Добавить вопрос». Результаты учеников также выводятся в виде таблицы. Каждая запись содержит информацию об ученика, а именно имя и Email, а также его лучшую оценку и лучший балл за данный тест.

2.7 Прохождение адаптивного теста учеником

Пользователь, который зарегистрирован как ученик, имеет возможность проходить тесты. Для этого нужно на странице желаемого теста перейти по ссылке «Начать».

За прохождение теста отвечает функция представления process_testing, в которой реализована пирамидальная стратегия адаптивного тестирования. Также, имеется вспомогательный класс Testing. Данный класс представляет собой набор атрибутов, хранящих текущее состояние теста, а именно:

- test – текущий тест;
- questions_by_levels – словарь вида уровень сложности : перемешанный список вопросов этой же сложности;
- current_stage – текущую стадию тестирования;
- current_level – текущий уровень;
- current_sub_level – текущий подуровень;
- points – счетчик набранных баллов;
- current_question – текущий вопрос;
- detail – список детальной информации о прохождении теста.

Метод get_random_question реализует выбор случайного, еще не задействованного в процессе тестирования вопроса, подходящего по уровню слож-

ности.

В начале тестирования функция process_testing получает GET запрос, создается новый объект класса Testing, так его еще не существует. Также создается новый файл, имя которого включает идентификатор пользователя и идентификатор текущего теста, что гарантирует уникальность такого файла. Впоследствии этот объект будет записываться в файл при окончании обработки ответа пользователя на вопрос и считываться из него в начале обработки. После завершения теста файл удаляется.

Выбирается случайный вопрос текущего уровня сложности, из базы данных выбираются ответы этого вопроса и перемешиваются для предоставления их пользователю в случайном порядке.

В шаблон «process_testing.html» передаются: текущее состояние теста, вопрос, список ответов на вопрос. После этого пользователь видит вопрос с вариантами ответа и отвечает на него.

Получая ответ от пользователя, его ответы проверяются, определяется балл за вопрос и прибавляется к текущему счетчику баллов. Также в список детальной информации добавляется кортеж из вопроса и баллов, полученных за ответ на него.

В зависимости от корректности ответа, происходит корректировка уровня сложности, причем вопрос с множественным выбором считается правильным, если полученный балл не менее $0,5$. После – переход на следующую стадию.

После этого все шаги повторяются до тех пор, пока не будут пройдены все стадии тестирования. Далее, набранные баллы переводятся в 5-бальную оценку, создается экземпляр класса Resutl, где фиксируется результат прохождения теста учеником. После этого запись заносится в базу данных, а также создается список объектов класса DetailResults, которые тоже добавляются в базу данных.

Файл, который использовался как хранилище промежуточных результатов, удаляется, а пользователь перенаправляется на страницу данного теста, где отображается его лучший результат и статистика прохождения данного теста.

ЗАКЛЮЧЕНИЕ

При написании настоящей работы поставленная цель была достигнута. Разработана система адаптивного тестирования в форме web-приложения. В качестве алгоритма адаптивного тестирования выбрана пирамидальная стратегия. В приложении реализован следующий функционал:

- регистрация и авторизация пользователя
- создание адаптивного теста
- прохождение адаптивного теста
- просмотр результатов и статистики прохождения тестов

Были проведены испытания данной системы, демонстрирующее его работоспособность.

Поставленные задачи были решены следующим образом:

- изучена научная литература по теме адаптивного тестирования;
- изучены существующие алгоритмы адаптивного тестирования;
- при разработке системы использовался язык программирования Python с использованием web-фреймворка Flask, а также его расширений.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 *Звонников, В. И.* Современные средства оценивания результатов обучения / В. И. Звонников, М. Б. Челышкова. — 5 изд. — Москва: Академия, 2013. — 304 с.

2 *Глебов, В. А.* Адаптивное тестирование как современное средство контроля результатов обучения / В. А. Глебов // *Актуальные вопросы современной науки*. — 2012. — № 23. — С. 75–87.

3 *Зайцева, Л. В.* Модели и методы адаптивного контроля знаний / Л. В. Зайцева, Н. О. Прокофьева // *Образовательные технологии и общество*. — 2004. — Т. 7, № 4. — С. 30–49.

4 *Малыгин, А. А.* Стратегии и алгоритмы реализации адаптивных технологий педагогических измерений / А. А. Малыгин // *Вестник университета*. — 2013. — № 15. — С. 393–402.

5 *Фарфоров, А. С.* Алгоритмы адаптивного тестирования знаний на основе теории нечётких множеств / А. С. Фарфоров, С. П. Дударов // *Успехи в химии и химической технологии*. — 2008. — Т. 22, № 1. — С. 63–67.

6 *Малыгин, А. А.* Компьютерное адаптивное тестирование в обеспечение качества дистанционного обучения / А. А. Малыгин // *Вестник университета*. — 2010. — № 4. — С. 166–169.

7 *Федоров, Д. Ю.* Основы программирования на примере языка Python / Д. Ю. Федоров. — Санкт-Петербург: Издательство Юрайт, 2019. — 161 с.

8 Template Designer Documentation [Электронный ресурс]. — URL: https://jinja.palletsprojects.com/en/3.1.x/templates/ (Дата обращения 15.05.2022). Загл. с экр. Яз. англ.

9 Configuration Handling [Электронный ресурс]. — URL: https://flask.palletsprojects.com/en/2.1.x/config/ (Дата обращения 12.05.2022). Загл. с экр. Яз. англ.

10 Define and Access the Database [Электронный ресурс]. — URL: https://flask.palletsprojects.com/en/2.1.x/tutorial/database/ (Дата обращения 12.05.2022). Загл. с экр. Яз. англ.

11 Quickstart [Электронный ресурс]. — URL: https://flask.palletsprojects.com/en/2.1.x/quickstart/ (Дата обращения 12.05.2022). Загл. с экр. Яз. англ.

12 Forms [Электронный ресурс]. — URL: https://wtforms.readthedocs.io/en/3.0.x/forms/ (Дата обращения 15.05.2022). Загл. с экр. Яз. англ.

13 Validators [Электронный ресурс]. — URL: https://wtforms.readthedocs.io/en/3.0.x/validators/ (Дата обращения 17.05.2022). Загл. с экр. Яз. англ.

14 Fields [Электронный ресурс]. — URL: https://wtforms.readthedocs.io/en/3.0.x/fields/ (Дата обращения 16.05.2022). Загл. с экр. Яз. англ.

15 Select, Insert, Delete [Электронный ресурс]. — URL: https://flask-sqlalchemy.palletsprojects.com/en/2.x/queries/ (Дата обращения 13.05.2022). Загл. с экр. Яз. англ.

16 Message Flashing [Электронный ресурс]. — URL: https://flask.palletsprojects.com/en/2.1.x/patterns/flashing/ (Дата обращения 17.05.2022). Загл. с экр. Яз. англ.

17 Security Helpers [Электронный ресурс]. — URL: https://werkzeug.palletsprojects.com/en/2.1.x/utils/#module-werkzeug.security (Дата обращения 14.05.2022). Загл. с экр. Яз. англ.

18 Your User Class [Электронный ресурс]. — URL: https://flask-login.readthedocs.io/en/latest/#your-user-class (Дата обращения 14.05.2022). Загл. с экр. Яз. англ.

19 *Мэтиз, Э.* Изучаем Python: программирование игр, визуализация данных, веб-приложения / Э. Мэтиз. — Санкт-Петербург: Издательский дом «Питер», 2019. — 512 с.

20 *Любанович, Б.* Простой Python. Современный стиль программирования / Б. Любанович. — Санкт-Петербург: Питер, 2019. — 480 с.

21 HTMLElement.style [Электронный ресурс]. — URL: https://developer.mozilla.org/en-US/docs/Web/API/HTMLElement/style (Дата обращения 20.05.2022). Загл. с экр. Яз. англ.