Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Дисциплина: Жизненный цикл разработки программного обеспечения

ОТЧЕТ
по лабораторной работе № 3

Студент:

Проверил:

МИНСК 2024

ЧАСТЬ 1

Приложение содержащее сразу некоторое количество игр достаточно удобно и развлекательно. Поэтому вот как мы бы реализовали указанные пункты:

1. Тип приложения: Это настольное приложение, содержащее несколько игр которые можно выбрать, а также базу данных с помощью которой возможно с помощью одного устройства играть сразу нескольким пользователям без утери данных.

2. Стратегия развертывания: Приложение будет скачиваться. Это обеспечит быстрый доступ к данным и функциям приложения без необходимости подключения к
интернету.

3. Выбор технологии:
   - Python: Для реализации основной логики приложения и взаимодействия с базой данных.
   - Pycharm2024: Для разработки самого приложения.
   - SQL lite: Для создания базы данных в которой будут хранится данные пользователей :пароль, логин и их рекорд в каждой игре.

4. Показатели качества:
   - Производительность: Приложение должно быстро обрабатывать запросы и возвращать результаты.
   - Надежность: Приложение должно обеспечивать целостность данных.
   - Использование ресурсов: Приложение должно эффективно использовать системные ресурсы.
   - Пользовательский интерфейс: Интерфейс должен быть интуитивно понятным и удобным для пользователя.

5. Пути реализации сквозной функциональности:
   - База данных: Все данные связанные с пользователем, будут храниться в базе данных.
   - Многопользовательность: Возможность использования приложения большому количеству игроков с сохранением их достижений и рекордов.

6. Структурная схема приложения:
   - Блок Меню: Этот блок будет отвечать за выбор нужного вам действия.
   - Блок Регистрации: Этот блок будет отвечать за создание учетной записи пользователя.
   - Блок Входа: Этот блок будет включать проверку подлинности введенного пароля и логина1.

- Блок База данных: Этот блок будет отвечать за хранение учетных записей пользователя.
- Блок Пользовательский интерфейс: Этот блок будет отвечать за взаимодействие с пользователем.

ЧАСТЬ 2

основные поля:

db - поле отвечающее за все поля в БД

root - поле используется для хранения ссылки на корневое окно (root window) графического интерфейса Tkinter

username - поле отвечающее за имя пользователя согласно его БД

self.title() - поле отвечающее за название окна

screen_size_display / self.geometry - поля отвечающее за размеры окна

lambda: self.show_game_menu_callback - лябда-функция отвечающая за выход из функции

def show_main_menu(self): - функция отвечающая за вывод окна с главным меню

Работа с

audio - папка со звуковым аккомпанементом для игры "Гонки";

images - папка с картинками для игры "Гонки";

resources - папка с ресурсами для игры "Динозавр";

textfile - папка с лучшим рекордом в игре "Гонки";;

admin.py - файл с кодом режима администратора;

database.py - файл с кодом базы данных;

dino.py - файл с кодом игры "Динозавр";

main.py - осовной файл, который включает в себя главное меню и последующие окна;

race.py - файл с кодом игры "Гонки";

snake.py - файл с кодом игры "Змейка";

statistics_window.py - файл с кодом статистики пользователя;

sudoku.py - файл с кодом игры "Судоку";

tictactoe.py - файл с кодом игры "Крестики-нолики";

users.bd - файл для хранения данных.

ЧАСТЬ 3

1. Сравнение архитектур:
    - As is: Архитектура сосредоточена на базовых функциях приложения , связанных с главным графическим окном.
    - To be: Предлагаемая архитектура включает в себя большее количество игр, более широкий функционал и лучшую организацию, а также просмотр статистики пользователя.

2. Отличия и причины:
    - Блок Аналитика: В новой архитектуре появляется большее количество возможностей,безопасности и оформления самого приложения .
    - Блок База данных: Отдельный блок для управления данными улучшает целостность и безопасность информации.
    - Блок Пользовательский интерфейс: Разделение интерфейса от основной логики приложения упрощает разработку и тестирование.

3. Пути улучшения:
    - Модульность: Применение принципов модульного дизайна улучшит масштабируемость и качество самого продукта.
    - MVC (Model-View-Controller): Использование архитектурного шаблона MVC может помочь в разделении логики приложения, данных и пользовательского интерфейса.
    - Безопасность: Внедрение механизмов безопасности на уровне базы данных и пользовательского интерфейса для защиты данных и операций.