**Задание: Создание технического анализа для разрабатываемого программного обеспечения игрового приложения  “Составь слово”**

**1. Введение**

Цель анализа: Определение сроков и сложности выполнения задачи по разработке игры.

Краткое описание задачи: создание игры «Собери Слово» на языке

программирования Python

Обзор текущего состояния проекта или системы: программа находится на стадии разработки.

**2. Описание требований**

Операционная система: Программа должна быть совместима с операционными системами Windows.

Библиотеки: При разработке игры могут потребоваться дополнительные библиотеки для работы с графикой, звуком, пользовательским интерфейсом и т.д.

Ресурсы игры: Для игры «Составь Слово» потребуются изображения для

отображения игрового поля, букв и других элементов интерфейса, а также

звуковые эффекты.

Хранение данных: Для сохранения игры и других данных может потребоваться использование файловой системы или базы данных.

Обработка пользовательского ввода: Программа должна корректно обрабатывать пользовательский ввод с клавиатуры или мыши для управления игровым процессом.

Геймплей и логика игры: Необходимо разработать логику игры «Собери Слово» определить правила, условия победы/поражения и другие игровые механики.

Тестирование: После разработки программы необходимо провести тестирование

на соответствие требованиям, а также на наличие ошибок и недочетов.

Пользовательские требования:

Генерация случайного слова из списка доступных слов.

Возможность ввода слова пользователем.

Проверка правильности введенного слова.

Предоставление подсказки или подсказок при неудачных попытках Уведомление результате игры (победа/поражение).

**3. Анализ сложности**

Создание игры «Составь слово» может быть сложным из-за необходимости работы с различными аспектами, такими как генерация случайных слов, обработка пользовательского ввода, управление состоянием игры и дизайн интерфейса, также, могут возникнуть следующие сложности:

**1**. Выбор подходящей архитектуры: необходимо определить, какие классы, модули и компоненты будут включены в приложение, чтобы обеспечить его работу.

Управление памятью: в Python управление памятью происходит вручную, поэтому

2

необходимо быть осторожным при выделении и освобождении памяти, чтобы

избежать утечек памяти.

**2**. Обработка ошибок: обработка ошибок и исключений в коде, для корректной

работы в случае возникновения проблем.

**3**.Оптимизация производительности: оптимизация кода, избежание лишних

операций или использования неэффективных алгоритмов.

**4**. Тестирование: проведение тестирования приложения для отслеживания

корректной работы и отсутствия ошибок.

**4. Планирование и распределение работ**

Подготовительный этап (5дней)

Написание кода ( 5 дней)

Идентификация препятствий ( 5 дней)

Разработка плана действий( 5 дней)

Тестирование и проверка (5 день)

Документирование результатов (5 дней)

Завершающие действия (1 день)

**5. Анализ рисков и стратегии управления ими**

Отсутствие или недостаточное количество документации о зависимостях может затруднить понимание взаимосвязей между комплектами или модулями.

Сложность кода: Если код комплектов или модулей сложен и плохо структурирован, это может затруднить изучение зависимостей и понимание логики работы системы.

Изменения внешних компонентов: Изменения внешних компонентов или сторонних библиотек могут вызвать изменения в зависимостях, что требует дополнительного анализа и проверки.

**Стратегии управления рисками:**

Для достаточной документации – углубленно изучить связь между комплексами и модулями.

Для избежания ошибок в коде, проводим тестирование.

Для избежания в изменений в зависимости, проводим тестирование.

**6. Документирование результатов**

Техническая документация (схема алгоритма, диаграмма UML, инструкция

пользователя)

Результаты  тестирования.