**Техническое задание**

**Проект:** Spin Garden – Игровой Telegram-бот

**1. Введение**

**1.1. Назначение документа**

Техническое задание (ТЗ) определяет требования к разработке игрового Telegram-бота **Spin Garden**, включая функционал, архитектуру, интерфейсы и ограничения.

**1.2. Цель проекта**

Создание развлекательного бота с тремя азартными играми (Крутилка, 21, рулетка) для группы друзей с использованием виртуальной валюты.

**1.3. Целевая аудитория**

* Группы друзей в Telegram
* Пользователи, любящие казуальные игры

**2. Общее описание**

**2.1. Функционал**

Бот предоставляет:  
**3 игры**:

* **Крутилка** (3 барабана, множители)
* **Режим - 21** (До 4 игроков)
* **Рулетка** (европейский вариант, 37 секторов)

**Экономику**:

* Виртуальная валюта (монеты)
* Таблица лидеров
* Транзакции

**Web App**:

* Интерактивный интерфейс
* Анимации, звуки (опционально)

**2.2. Технологии**

|  |  |
| --- | --- |
| Компонент | Технологии |
| Backend | Python 3.x, python-telegram-bot |
| Frontend | HTML5, CSS3, JavaScript, WebSocket |
| База данных | SQLite |

**3. Подробные требования**

**3.1. Игровая механика**

**Крутилка**

* **Ставка:** от 5 монет
* **Выигрышные комбинации:**
  + 3 одинаковых символа → ×5

**Режим - 21**

* **Ставка:** от 15 монет
* **Лимит игроков:** 4
* **Правила:** классические

**Рулетка**

* **Типы ставок:**
  + Число (×10)
  + Красное/Чёрное (×2)
  + Чётное/Нечётное (×2)
  + Дюжина/Колонка (×3)

**4. Архитектура**

**4.1. База данных**

База данных SQLite состоит из 3 основных таблиц, связанных между собой

**4.1. Таблица**users**(Игроки)**

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Описание | Ограничения |
| user\_id | BIGINT | Уникальный ID пользователя в Telegram | PRIMARY KEY, NOT NULL |
| username | VARCHAR(32) | Имя пользователя (@username) | UNIQUE |
| balance | INTEGER | Текущий баланс (монеты) | DEFAULT = 50 |
| registration\_date | TIMESTAMP | Дата регистрации | DEFAULT CURRENT\_TIMESTAMP |
| last\_active | TIMESTAMP | Последняя активность |  |

**Пояснение:**  
Хранит основную информацию о пользователях. При первом запуске бота создается запись с начальным балансом 50 монет.

**4.2. Таблица**transactions**(Транзакции)**

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Описание | Ограничения |
| transaction\_id | INTEGER | Автоинкрементный ID | PRIMARY KEY, AUTOINCREMENT |
| user\_id | BIGINT | Ссылка на пользователя | FOREIGN KEY → users |
| amount | INTEGER | Сумма (+/-) | NOT NULL |
| type | ENUM | Тип: game\_win, game\_loss, bonus | NOT NULL |
| game\_type | VARCHAR(20) | Крутилка/рулетка/21 |  |
| created\_at | TIMESTAMP | Время операции | DEFAULT CURRENT\_TIMESTAMP |

**Пояснение:**Фиксирует все изменения баланса с указанием источника (игра, бонус и т.д.). Отрицательные значения amount — проигрыши.

**4.3. Таблица**game\_sessions**(Игровые сессии)**

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Описание | Ограничения |
| session\_id | INTEGER | Уникальный ID сессии | PRIMARY KEY |
| game\_type | VARCHAR(20) | Тип игры | NOT NULL |
| players | JSON | Участники: [{user\_id, bet, result}] | NOT NULL |
| outcome | JSON | Результаты: {winner\_id, prize} |  |
| created\_at | TIMESTAMP | Время начала | DEFAULT CURRENT\_TIMESTAMP |

**Пояснение:**Хранит полную историю игр. Поле players содержит ставки и результаты каждого участника.

**Пример запросов**

**Найти крупнейшие выигрыши:**

SELECT u.username, t.amount

FROM transactions t

JOIN users u ON t.user\_id = u.user\_id

WHERE t.type = 'game\_win'

ORDER BY t.amount DESC;

**Обоснование структуры**

1. **Нормализация:**
   * **Отдельные таблицы для пользователей и их действий**
   * **Минимизация дублирования данных**
2. **Производительность:**
   * **Индексы на user\_id, created\_at**
   * **JSON для гибкого хранения игровых данных**
3. **Масштабируемость:**
   * **Поддержка новых типов игр без изменения схемы**

**5. Интерфейс (UI/UX)**

**5.1. Основные экраны**

1. **Меню бота** (кнопки: Игры, Баланс, Лидерборд)
2. **Web App** (анимированные игры)
3. **Уведомления** (выигрыши, приглашения)

**5.2. Визуальный стиль**

* **Шрифты:** Times New Roman или Arial. Может будет замена на другой.
* **Цвета:** Красный, черный, желтый.
* **Анимации:** плавные переходы

**6. Безопасность**

* Валидация всех запросов
* Лимиты: 50 игр/час, 5 сек между ставками (Может потерпеть изменения)
* Логирование действий

**7. План тестирования**

**7.1. Цели тестирования**

* Проверить корректность работы всех игровых механик
* Убедиться в безопасности и защите от мошенничества
* Проверить работу Web App на разных устройствах
* Оценить производительность при высокой нагрузке

**7.2. Виды тестирования**

|  |  |  |
| --- | --- | --- |
| Тип тестирования | Методы | Инструменты |
| Функциональное | Проверка игр, транзакций, UI | PyTest, Postman |
| Нагрузочное | 50+ одновременных пользователей | Locust |
| Безопасность | Проверка валидации данных | OWASP ZAP, ручные тесты |
| Кросс-платформенное | iOS/Android/Desktop Web | BrowserStack |

**7.3. Этапы тестирования**

1. **Юнит-тесты** – Проверка отдельных функций (например, расчет выигрыша в рулетке).
2. **Интеграционное тестирование** – Взаимодействие бота + Web App + БД.
3. **Регрессионное тестирование** – После каждого обновления.

**7.4. Таблица тест-кейсов**

Общие функции

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Сценарий | Шаги | Ожидаемый результат | Статус |
| T1 | Регистрация нового пользователя | 1. Отправить /start боту | Создается запись в users |  |
| T2 | Проверка баланса | 1. Нажать кнопку баланса | Отображается текущий баланс |  |

**Игра "Крутилка"**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Сценарий | Шаги | Ожидаемый результат | Статус |
| T3 | Выигрышная комбинация (3 одинаковых) | 1. Сделать ставку 5 монет  2. Запустить барабаны | Баланс увеличивается на ×5 |  |
| T4 | Недостаточный баланс | 1. Попытаться поставить 1000 при балансе 500 | Ошибка "Недостаточно средств" |  |

**Игра "21"**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Сценарий | Шаги | Ожидаемый результат | Статус |
| T5 | Создание стола | 1. Создать комнату  2. Выбрать название комнаты, количество участников  3. Запустить комнату | Стол создан, ID игры в списке комнат |  |
| T6 | Подсчет очков (перебор) | 1. Взять карты до 22+ очков | Автоматический проигрыш |  |

**Игра "Рулетка"**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Сценарий | Шаги | Ожидаемый результат | Статус |
| T7 | Ставка на красное | 1. Поставить 10 монет на "красное"  2. Запустить рулетку | При выигрыше +20 монет |  |
| T8 | Одновременные ставки 2+ игроков | 1. Два пользователя ставят на разные поля | Балансы корректно обновляются |  |

**Безопасность**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Сценарий | Шаги | Ожидаемый результат | Статус |
| T9 | SQL-инъекция в поле "ставка" | 1. Ввести 10; DROP TABLE users; | Ошибка валидации |  |
| T10 | Подмена баланса через Web App | 1. Изменить balance в локальном хранилище | Сервер отклоняет запрос |  |

**7.5. Критерии успешного тестирования**

* 95% тест-кейсов выполнены с статусом **✅**
* Нагрузка в 50 пользователей — задержка ≤ 1 сек
* Отсутствие критичных уязвимостей