Decomposition Plan: Deadlock Analytics System

Модуль 1: Data Engineering (Фундамент)

Цель: Данные текут автоматически.

1. DB Schema: Спроектировать таблицы matches, heroes, items.
2. ETL Script: Скрипт load_data.py. Он читает JSON/CSV -> Чистит -> Пишет в SQLite.
3. Delta Update: (Опционально) Скрипт проверяет, какие матчи уже есть в БД, и не дублирует их.

Модуль 2: Analytical Core (Мозги)

Цель: Превратить данные в метрики. Здесь мы проверяем гипотезы. Тебе нужно заранее решить, какие вопросы мы задаем системе. Для MVP выберем 3 классических для MOBA:

1. Гипотеза 1 (Баланс): "Винрейт героев зависит от длительности матча". (Кто тащит в лейте — Haze или Vindicta?).
2. Гипотеза 2 (Экономика): "Корреляция Net Worth (золота) и Победы". (Насколько критично перефармить врага на 10-й минуте?).
3. Гипотеза 3 (Мета): "Самые популярные связки героев" (Или "Герои с самым высоким KDA на низком vs высоком рейтинге").

Модуль 3: Visualization & App (Лицо)

Цель: Сделать красиво и доступно. Используем Streamlit.

1. Dashboard Page: Главная страница с ключевыми метриками (Всего матчей, Топ-3 героя по винрейту сегодня).
2. Hero Drill-down: Страница, где можно выбрать конкретного героя и посмотреть его статистику.
3. Deployment: Заливаем код на GitHub -> Подключаем Streamlit Cloud (это бесплатно) -> Получаем красивую ссылку deadlock-analytics-taras.streamlit.app.

Почему это круто для тебя?

1. *Инженерный подход: Ты не просто "посчитал", ты "создал продукт".*
2. *Живая система: Выйдет новый патч в Deadlock — ты просто запустишь скрипт обновления, и твое приложение покажет новую мету. Ты сможешь кидать скриншоты в чаты/LinkedIn и быть тем самым "Светилом", который объясняет людям, что происходит в игре.*

🎯 Цель спринта

Создать и задеплоить веб-приложение (Streamlit), которое автоматически анализирует матчи Deadlock, хранит их в локальной БД и визуализирует ключевые метрики баланса.

Модуль 1: Data Engineering (Фундамент)

Твоя роль: Backend Engineer / Data Engineer Инструменты: Python (sqlite3, pandas, requests), Antigravity.

🗓️ Воскресенье, 18 января (10:30) — База Данных и ETL

- Задача 1.1: Схема БД (Schema Design)
    - Создать файл database.py.
    - Спроектировать структуру SQLite. Нам нужны минимум 2 таблицы:
        - matches (id матча, длительность, дата, кто победил, режим игры).
        - player_stats (связка с matches по id, id героя, net worth, K/D/A, урон, хилок).
    - *Конкретика:* Убедись, что match_id — это Primary Key, чтобы не было дублей.
- Задача 1.2: Загрузчик (ETL Script)
    - Создать файл etl_pipeline.py.
    - Написать функцию load_matches_from_csv(path), которая читает твои CSV и делает INSERT в базу данных.
    - Написать функцию update_from_api(), которая (в будущем) будет стучаться в API за новыми матчами.
- Definition of Done (DoD): У тебя есть файл deadlock.db, и ты можешь открыть его через любой DB Browser (или DBeaver) и увидеть там данные.

Модуль 2: Analytical Core (Мозги)

Твоя роль: Data Analyst Инструменты: SQL, Pandas, Jupyter Notebook.

📅 Понедельник, 19 января (10:30) — SQL-Логика и Гипотезы

- Гипотеза 1 (Late Game Heroes): "Винрейт Haze растет после 30-й минуты, а Bebop падает".
    - *Действие:* Написать SQL-запрос, который группирует матчи по бакетам длительности (0-15 мин, 15-25 мин, 25-40 мин, 40+ мин) и считает Winrate для каждого героя в каждом бакете.
- Гипотеза 2 (Snowball Effect): "Если команда ведет по золоту >5k на 10-й минуте, она выигрывает в 90% случаев".
    - *Действие:* Если в данных есть таймлайны золота — написать запрос корреляции. Если нет — анализировать итоговый Net Worth победителей vs проигравших.
- Гипотеза 3 (Meta Tier List): "Самые эффективные герои текущего патча".
    - *Действие:* Рассчитать взвешенный рейтинг: Pick Rate * Win Rate.
- Definition of Done (DoD): У тебя есть Jupyter Notebook, где ячейки кода выдают таблицы с ответами на эти три вопроса.

Модуль 3: Visualization & App (Лицо)

Твоя роль: Frontend Developer (на минималках) / BI Specialist Инструменты: Streamlit, Plotly.

🗓️ Вторник, 20 января (18:00) — Прототип приложения

- Задача 3.1: Hello Streamlit
    - Создать файл app.py.
    - Написать базовый код:

Python

import streamlit as st

import pandas as pd

import sqlite3

st.title('Deadlock Meta Analyzer')

conn = sqlite3.connect('deadlock.db')

df = pd.read_sql('SELECT * FROM matches', conn)

st.write(df.head()) # Проверка, что данные выводятся

- Задача 3.2: Перенос графиков
    - Взять код графиков из понедельника (Plotly) и вставить их в app.py через st.plotly_chart().
    - Сделать выпадающий список (Dropdown) для выбора Героя.
- Definition of Done (DoD): Ты запускаешь в терминале streamlit run app.py, и у тебя в браузере открывается локальный сайт с графиками.

🗓️ Четверг, 22 января (18:00) — Улучшение UX и Фильтры

- Задача 3.3: Интерактивность
    - Добавить слайдер "Длительность матча" (чтобы отсечь турбо-игры по 10 минут).
    - Добавить вкладки (Tabs): General Meta, Hero Analysis, Raw Data.

- Задача 3.4: "Секретная фича"
    - Добавь блок "My Take" (Мое мнение). Текстовое поле, где ты как аналитик пишешь выводы текстом под графиками. Например: *"Как мы видим, Seven переоценен на низком рейтинге..."*. Это покажет работодателю, что ты умеешь делать выводы.
- Definition of Done (DoD): Приложение выглядит законченным, кнопки работают, ничего не падает.

Модуль 4: Deployment & Polish (Финиш)

Твоя роль: DevOps / Product Manager Инструменты: GitHub, Streamlit Cloud.

🗓️ Суббота, 24 января (10:30) — Деплой и README

- Задача 4.1: GitHub Repo
    - Создать файл requirements.txt (список библиотек: streamlit, pandas, plotly).
    - Залить app.py, database.py и саму базу deadlock.db (если она небольшая) на GitHub.
- Задача 4.2: Streamlit Cloud
    - Зайти на share.streamlit.io -> Войти через GitHub -> Выбрать свой репозиторий.
    - Нажать кнопку "Deploy".
- Задача 4.3: Продающее описание
    - В README.md написать: "Это инструмент для анализа меты Deadlock. Он позволяет игрокам понять, кого пикать. Стек: Python, SQL, Streamlit".
- Definition of Done (DoD): Ты кидаешь ссылку мне (в чат) или другу, и она открывается с телефона.