Product Requirements Document (PRD)

Project Title: Flappy Bird H5 Game with NFT and Blockchain on Cloud

Project Detail:

Building Flappy Bird H5 Game with NFT and Blockchain.

This project develops a cloud-based Flappy Bird H5 game integrating blockchain for NFT collection and trading. Built with Python Django (backend) and PostgreSQL, it is deployed on Heroku and Firebase Hosting.

**1. Project Overview**

**1.1 Objective**

Develop a cloud-based H5 game inspired by Flappy Bird, integrating blockchain technology to allow players to collect and trade NFTs. The project demonstrates the integration of cloud computing, blockchain, and H5 game development.

**1.2 Target Audience**

* Blockchain and NFT enthusiasts.
* Casual gamers.
* Developers and students interested in blockchain and cloud computing.

**1.3 Success Metrics**

* Fully functional H5 game hosted on the cloud.
* Smooth integration of blockchain for NFT minting and trading.
* Deployment of backend and frontend on cloud platforms.
* Positive feedback from a small group of testers.

**2. Functional Requirements**

**2.1 Gameplay Features**

1. **Basic Gameplay Mechanics:**
   * Player controls a bird that flaps to avoid obstacles.
   * Endless runner format with increasing difficulty.
   * Scoring system to track progress.
2. **NFT Collection:**
   * Players collect in-game items (e.g., skins, badges) that can be minted as NFTs.
   * Randomized drops for rare items.
3. **Leaderboard:**
   * Global leaderboard stored in the cloud.
   * Displays high scores and top players.
4. **User Authentication:**
   * Integration with blockchain wallets (e.g., MetaMask) for login.
   * Optional cloud-based login for non-blockchain users.

**2.2 Blockchain Features**

1. **NFT Minting:**
   * Use ERC-721 or ERC-1155 standard.
   * Metadata includes item name, rarity, and player details.
2. **Marketplace Integration:**
   * Players can trade NFTs via third-party platforms (e.g., OpenSea).
3. **Testnet Deployment:**
   * Utilize testnets like Rinkeby or Polygon Mumbai for development.

**2.3 Backend Features**

1. **Game Data Management:**
   * Store scores, progress, and non-NFT items in a PostgreSQL database.
2. **Blockchain Integration:**
   * Use Web3.py or Ethers.js to interact with smart contracts.
3. **API Development:**
   * Provide RESTful APIs for frontend-backend communication using Django REST Framework.

**2.4 Cloud Features**

1. **Frontend Hosting:**
   * Deploy the game on Firebase Hosting or AWS S3.
2. **Backend Deployment:**
   * Host backend server on Heroku or AWS EC2.
3. **Storage:**
   * Use IPFS for NFT metadata.
   * Store game data in PostgreSQL.

**3. Non-Functional Requirements**

1. **Performance:**
   * The game should load within 3 seconds on standard devices.
   * Handle up to 100 concurrent users in the initial phase.
2. **Scalability:**
   * Backend should scale to support additional users.
3. **Security:**
   * Secure API endpoints with authentication tokens.
   * Use HTTPS for all communications.
4. **Usability:**
   * Intuitive UI for both gamers and blockchain users.

**4. Technical Requirements**

**4.1 Frontend**

* **Framework:** PixiJS or PlayCanvas.
* **Features:**
  + Responsive design for mobile and desktop.
  + Integration with Web3.js for blockchain interaction.

**4.2 Backend**

* **Framework:** Python Django with Django REST Framework.
* **Database:** PostgreSQL.
* **Hosting:** Heroku or AWS EC2.

**4.3 Blockchain**

* **Platform:** Ethereum or Polygon.
* **API:** Alchemy or Infura for blockchain interactions.
* **Smart Contracts:** Written in Solidity.

**4.4 Cloud**

* **Frontend Hosting:** Firebase Hosting or AWS S3.
* **Backend Hosting:** Heroku or AWS.
* **Storage:** IPFS for NFT metadata.

**5. Development Milestones**

**5.1 Phase 1: Planning and Setup**

* Define project architecture.
* Set up development environment (Django, PostgreSQL, PixiJS/PlayCanvas, blockchain testnet).
* Choose cloud platforms (Firebase, AWS, Heroku).

**5.2 Phase 2: Core Gameplay Development**

* Implement Flappy Bird mechanics.
* Add scoring and obstacle generation.
* Test gameplay on multiple devices.

**5.3 Phase 3: Blockchain Integration**

* Develop and deploy smart contracts.
* Integrate Web3.js/Ethers.js for wallet connections.
* Implement NFT minting and metadata storage on IPFS.

**5.4 Phase 4: Backend Development**

* Build APIs for game data and blockchain communication.
* Set up database for user progress and leaderboard.
* Deploy backend to Heroku/AWS.

**5.5 Phase 5: Cloud Deployment**

* Deploy frontend to Firebase Hosting/AWS S3.
* Integrate frontend and backend.

**5.6 Phase 6: Testing and Optimization**

* Conduct functional and performance testing.
* Optimize loading times and gameplay.
* Ensure blockchain transactions are smooth.

**5.7 Phase 7: Final Deployment**

* Deploy to production cloud platforms.
* Share game with testers and gather feedback.

**6. Risks and Mitigation**

**6.1 Blockchain Costs**

* **Risk:** High gas fees on Ethereum mainnet.
* **Mitigation:** Use Polygon for lower costs.

**6.2 Performance Issues**

* **Risk:** Slow game loading or backend response.
* **Mitigation:** Optimize assets and use caching.

**6.3 Security Vulnerabilities**

* **Risk:** Smart contract exploits or API breaches.
* **Mitigation:** Audit smart contracts and secure API endpoints.

**7. Deliverables**

1. Fully functional H5 Flappy Bird game.
2. Smart contract deployed on testnet.
3. Cloud-hosted frontend and backend.
4. Documentation:
   * Game design.
   * Deployment guide.
   * API documentation.

**8. Timeline**

| **Phase** | **Duration** | **Deliverables** |
| --- | --- | --- |
| Planning | 1 week | Architecture and environment setup |
| Gameplay Development | 2 weeks | Flappy Bird game mechanics |
| Blockchain Integration | 2 weeks | Smart contracts and NFT features |
| Backend Development | 2 weeks | APIs and database setup |
| Cloud Deployment | 1 week | Frontend and backend hosting |
| Testing & Optimization | 1 week | Bug fixes and performance tuning |
| Final Deployment | 1 week | Production release |

**Phân tích và giải thích file PRD (Product Requirements Document) của dự án Flappy Bird H5 tích hợp NFT và Blockchain:**

**1. Tổng quan dự án**

* **Mục tiêu**:
  + Phát triển trò chơi H5 dựa trên Flappy Bird, tích hợp công nghệ blockchain để người chơi thu thập và giao dịch NFT.
  + Triển khai trên đám mây, sử dụng Python Django (backend), PostgreSQL và triển khai trên Heroku, Firebase Hosting.
* **Đối tượng người dùng**:
  + Người đam mê blockchain và NFT.
  + Người chơi game giải trí thông thường.
  + Nhà phát triển và sinh viên quan tâm đến blockchain và điện toán đám mây.
* **Tiêu chí thành công**:
  + Trò chơi hoạt động ổn định trên nền tảng đám mây.
  + Tích hợp blockchain để đúc (mint) và giao dịch NFT trơn tru.
  + Nhận được phản hồi tích cực từ một nhóm thử nghiệm nhỏ.

**2. Yêu cầu chức năng**

* **Chức năng trò chơi**:
  + **Cơ chế chơi**: Điều khiển chim tránh chướng ngại vật theo dạng endless runner, điểm số tăng theo thời gian.
  + **Thu thập NFT**: Người chơi nhận các vật phẩm trong game (skin, huy hiệu) và đúc chúng thành NFT. Một số vật phẩm hiếm sẽ xuất hiện ngẫu nhiên.
  + **Bảng xếp hạng**: Hiển thị điểm cao và danh sách người chơi hàng đầu, lưu trữ trên đám mây.
  + **Xác thực người dùng**: Hỗ trợ đăng nhập bằng ví blockchain (e.g., MetaMask) hoặc đăng nhập qua đám mây.
* **Chức năng blockchain**:
  + **Đúc NFT**: Sử dụng tiêu chuẩn ERC-721 hoặc ERC-1155, metadata bao gồm tên vật phẩm, độ hiếm, và thông tin người chơi.
  + **Giao dịch NFT**: Hỗ trợ giao dịch qua các nền tảng như OpenSea.
  + **Triển khai trên testnet**: Sử dụng mạng thử nghiệm như Rinkeby hoặc Polygon Mumbai.
* **Chức năng backend**:
  + Quản lý dữ liệu trò chơi: Lưu điểm, tiến trình, và các vật phẩm không phải NFT trên PostgreSQL.
  + Tích hợp blockchain: Sử dụng Web3.py hoặc Ethers.js để giao tiếp với hợp đồng thông minh.
  + Phát triển API: Cung cấp API RESTful để giao tiếp giữa frontend và backend.
* **Chức năng đám mây**:
  + Lưu trữ frontend: Triển khai trên Firebase Hosting hoặc AWS S3.
  + Triển khai backend: Lưu trữ server backend trên Heroku hoặc AWS EC2.
  + Lưu trữ dữ liệu: Metadata NFT lưu trên IPFS, dữ liệu game lưu trên PostgreSQL.

**3. Yêu cầu phi chức năng**

* **Hiệu năng**:
  + Tải trò chơi dưới 3 giây trên các thiết bị thông thường.
  + Hỗ trợ 100 người dùng đồng thời ở giai đoạn đầu.
* **Khả năng mở rộng**: Backend có thể mở rộng để hỗ trợ thêm người dùng.
* **Bảo mật**: Sử dụng HTTPS, bảo vệ API bằng token xác thực.
* **Trải nghiệm người dùng**: Giao diện trực quan, dễ sử dụng.

**4. Yêu cầu kỹ thuật**

* **Frontend**:
  + Framework: PixiJS hoặc PlayCanvas.
  + Tính năng: Thiết kế responsive, tích hợp Web3.js để giao tiếp blockchain.
* **Backend**:
  + Framework: Python Django + Django REST Framework.
  + Database: PostgreSQL.
  + Hosting: Heroku hoặc AWS EC2.
* **Blockchain**:
  + Nền tảng: Ethereum hoặc Polygon.
  + API: Alchemy hoặc Infura.
  + Hợp đồng thông minh: Viết bằng Solidity.
* **Đám mây**:
  + Lưu trữ frontend: Firebase Hosting hoặc AWS S3.
  + Lưu trữ backend: Heroku hoặc AWS.
  + Lưu metadata: IPFS.

**5. Mốc phát triển**

* **Giai đoạn 1**: Lập kế hoạch và thiết lập môi trường (1 tuần).
* **Giai đoạn 2**: Phát triển gameplay cốt lõi (2 tuần).
* **Giai đoạn 3**: Tích hợp blockchain (2 tuần).
* **Giai đoạn 4**: Phát triển backend (2 tuần).
* **Giai đoạn 5**: Triển khai trên đám mây (1 tuần).
* **Giai đoạn 6**: Kiểm thử và tối ưu hóa (1 tuần).
* **Giai đoạn 7**: Triển khai chính thức (1 tuần).

**6. Rủi ro và cách giảm thiểu**

* **Chi phí blockchain cao**: Sử dụng Polygon để giảm phí gas.
* **Vấn đề hiệu năng**: Tối ưu hóa tài nguyên và sử dụng caching.
* **Lỗ hổng bảo mật**: Kiểm toán hợp đồng thông minh và bảo mật API.

**7. Sản phẩm bàn giao**

* Trò chơi Flappy Bird H5 hoàn chỉnh.
* Hợp đồng thông minh trên testnet.
* Frontend và backend được lưu trữ trên đám mây.
* Tài liệu hướng dẫn triển khai và thiết kế API.