Project Documentation: 2D Game Builder Platform

1. Project Overview

This platform allows any user to describe the type of 2D(as of now it only supports 2D game creation) game they want (e.g., "A Dragon Ball Z fighting game with energy blasts"), and the system automatically generates a playable browser game for them. Users can view all their games on the home screen, and each game is stored in its own folder. Users can also modify game assets, backgrounds, and other details using simple text input.

Key Features:

- Text-to-game: Users describe a game, and the system builds it.
- Game library: All created games are visible and playable from the home screen.
- Editable: Users can update assets and game details via text.
- Modular: Each game is self-contained in its own folder.

---

2. Technical Stack

Layer	Technology
Frontend	React + TailwindCSS
Game Engine	Phaser.js
Backend	FastAPI (Python)
AI/Asset Gen	Google Gemini(Free plan)
Storage	(Local)
Deployment	(Local)

---

3. How the System Works (Step-by-Step)

A. Game Creation Flow

1. **User Input:**

User enters a game description (prompt) in the frontend (React form).

2. **Backend Processing:**

- The prompt is sent to the FastAPI backend (/generate endpoint).
- The backend uses the Gemini AI model to generate a **game blueprint** (JSON with meta, characters, scene, gameplay, UI/UX).
- The blueprint is saved in a new game folder.
- The backend then generates a **production plan** (detailed asset prompts, Phaser code templates, UI templates, loading plan).
- The production plan is saved in the same folder.

3. **Asset Generation:**

- Gemini Image Generation API is getting used for assests generation.
- Assets are organized in /assets subfolders (characters, backgrounds, effects, UI).

4. **Game Assembly:**

- Phaser game files (main.js, loadAssets.js, createScene.js, etc.) are generated per game, using the production plan.
- All files for a game are stored in its own folder under games/.

5. **Frontend Display:**

- The home screen lists all games by reading the games/ folder and their blueprints.
- Users can play, modify, or delete games.

B. Game Modification Flow

- Users can update assets or game details by submitting new text prompts.
- The backend updates the assets in the game's folder.

---

4. Project Structure & File Roles

A. Backend (game-builder-platform/BE/)

- main.py: FastAPI app, all main endpoints, game generation logic, static file serving.
- services/gemini.py: Handles AI calls to generate blueprints, production plans, and assets.

- services/storage.py: Handles saving/loading blueprints and production plans.
- games/: Each subfolder is a separate game, containing:
- blueprint.json: High-level game design (permanent for each game).
- production_plan.json: Detailed plan for assets, Phaser code, UI (permanent for each game).
- phaser/: All game engine files (dynamic, generated per game).
- assets/: All images, sprites, backgrounds, effects, UI (dynamic, can be replaced/updated).

NOTE: All the folders and files under game folder are dynamic and will be different for each game.

B. Frontend (game-builder-platform/src/)

- components/: Reusable React components (PromptForm, GenerationProgress, GameCard, etc.).
- pages/: Main app pages (Home, CreateGame, PlayGame, ModifyGame).
- engine/: Game logic modules (assetLoader, fightingEngine, etc.).
- api/: API communication (gameAPI.js).
- config/: Game-specific configs (if needed).

---

5. Permanent vs. Dynamic Code/Files

Permanent (core logic, rarely changes)	Dynamic (changes per game or user input)
Backend FastAPI code (main.py, services/)	games/<game_name>/blueprint.json
React components and pages	games/<game_name>/production_plan.json
Game engine modules (Phaser templates)	games/<game_name>/phaser/ (all JS, HTML)
API communication logic	games/<game_name>/assets/ (all images, etc.)
Asset generation logic	

---

6. Current Progress & Prototype Limitations

- I have successfully implemented the complete end-to-end flow—from generating a prompt to creating and playing the game. The game and its assets are being generated without major issues.
- Users can currently **create**, **play**, **modify**, and **delete** games.
- **Character portraits** still lack visual quality and need significant improvement.
- Only **2D games** are supported at this stage.
- Basic gameplay controls like **move left**, **move right**, **jump**, and **attack** are functional.
- Advanced features such as **character selection**, **level design**, and **game progression systems** are not yet implemented.
- Some parts of the game generation still require **manual fixes**, which should ideally be handled automatically in future versions.

---

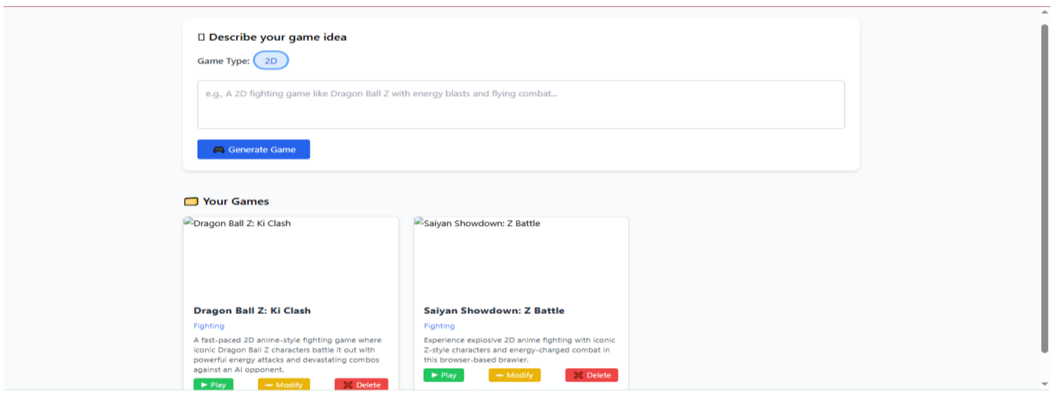
7. Future Improvements & Roadmap

- **Enhanced Asset Generation**:
    - Upgrading to more powerful models will likely improve both the **code quality** and the **visual assets** (images, animations, etc.). Currently, a free version of the Gemini model is being used.
- **Improved UI Templates**:
    - In the production version, I plan to incorporate the ui_templates code, which can significantly enhance the look and usability of the game interface.
- **Better Libraries & Tools**:
    - By integrating more robust and feature-rich libraries, we can unlock greater possibilities in terms of **game mechanics**, **interactivity**, and **visual fidelity**.

---

8. UI Screenshots

HOME SCREEN:



Describe your game idea

Game Type: 2D

e.g., A 2D fighting game like Dragon Ball Z with energy blasts and flying combat...

Generate Game

Your Games

Dragon Ball Z: Ki Clash

Dragon Ball Z: Ki Clash

Fighting

A fast-paced 2D anime-style fighting game where iconic Dragon Ball Z characters battle it out with powerful energy attacks and devastating combos against an AI opponent.

▶ Play — Modify ✖ Delete

Saiyan Showdown: Z Battle

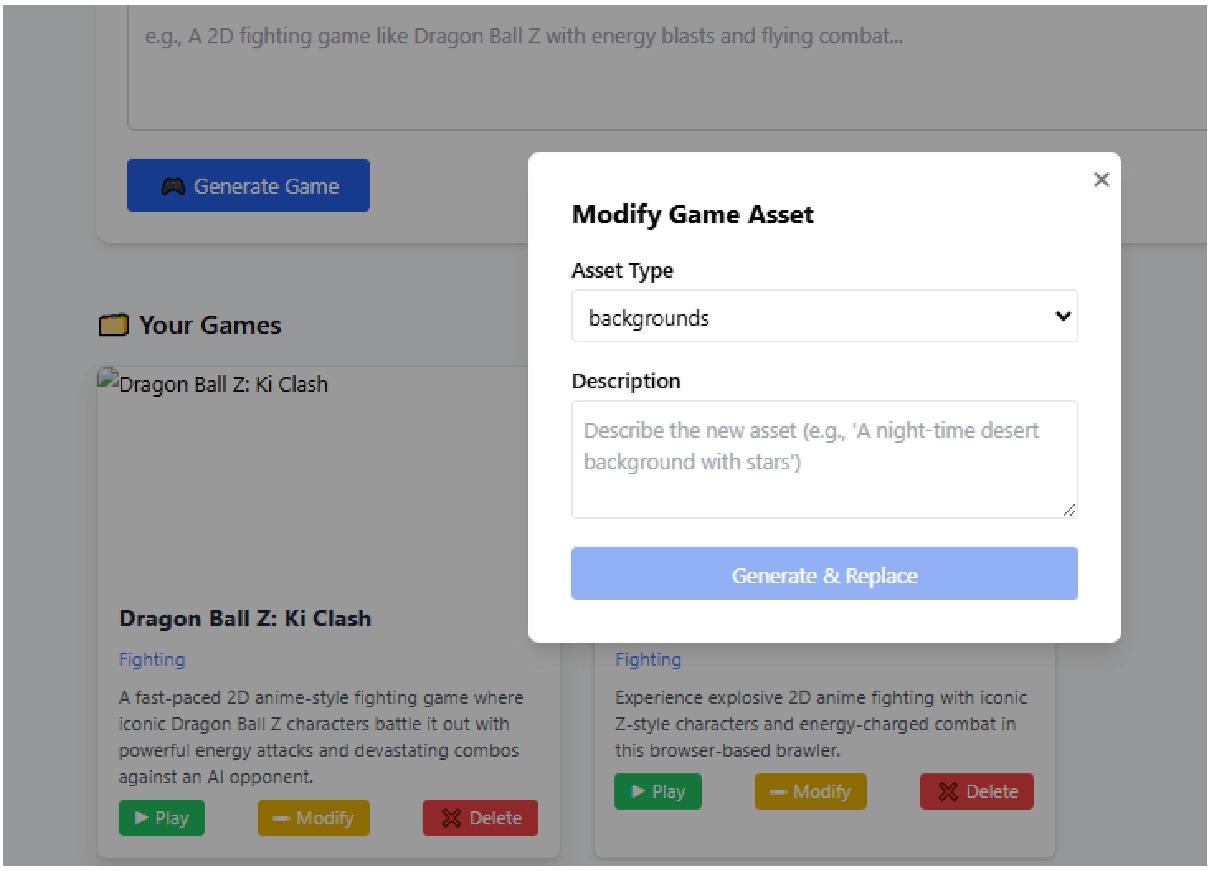
Saiyan Showdown: Z Battle

Fighting

Experience explosive 2D anime fighting with iconic Z-style characters and energy-charged combat in this browser-based brawler.

▶ Play — Modify ✖ Delete

MODIFY GAME ASSETS POP-UP



e.g., A 2D fighting game like Dragon Ball Z with energy blasts and flying combat...

Generate Game

Your Games

Dragon Ball Z: Ki Clash

Dragon Ball Z: Ki Clash

Fighting

A fast-paced 2D anime-style fighting game where iconic Dragon Ball Z characters battle it out with powerful energy attacks and devastating combos against an AI opponent.

▶ Play — Modify ✖ Delete

Fighting

Experience explosive 2D anime fighting with iconic Z-style characters and energy-charged combat in this browser-based brawler.

▶ Play — Modify ✖ Delete

Modify Game Asset

Asset Type: backgrounds

Description: Describe the new asset (e.g., 'A night-time desert background with stars')

Generate & Replace

GAMEPLAY SCENES