**1. THE MAZE: MVP SPECIFICATION**

Minimum Viable Product (MVP) refers to the basic version of the game that includes core features and functionalities.

This MVP specification covers the essential elements necessary to create a playable and enjoyable maze game. Additional features and refinements can be implemented in subsequent iterations based on feedback and further development.

Here is the MVP specification for "The Maze" game:

**(a). Game Mechanics:**

- Player Movement: The player can navigate through the maze using arrow keys or touch controls (for mobile).

- Collision Detection: The player collides with walls and obstacles, preventing them from passing through.

- Objective: The objective is to reach the exit point of each level.

**(b). Level Design**:

- Maze Generation: Randomly generate mazes with walls, pathways, and obstacles using algorithmic approaches like recursive backtracking or randomized Prim's algorithm.

- Multiple Levels: Include a series of increasingly challenging levels with varying maze sizes and layouts.

**(c). Puzzles and Challenges:**

- Key and Door Mechanics: Some areas may be locked, requiring the player to find keys scattered throughout the maze to unlock doors and progress.

- Traps and Obstacles: Introduce traps, such as moving platforms or spikes, which the player must avoid to survive.

- Environmental Puzzles: Incorporate puzzles that require interacting with elements in the environment, such as pushing objects, activating switches, or leveraging physics-based mechanics.

**(d). Visuals and Audio:**

- 2D Graphics: Use visually appealing and cohesive art assets for the maze, player character, keys, doors, and other elements.

- Atmospheric Audio: Include background music, sound effects for interactions, and ambient sounds to enhance immersion.

**(e). User Interface (UI):**

- Main Menu: Provide options to start the game, access settings, view instructions, and exit.

- Level Selection: Allow players to choose levels they have unlocked or previously completed.

- In-game HUD: Display relevant information, such as the player's current level, collected keys, and remaining lives.

**(f). Scoring and Progression:**

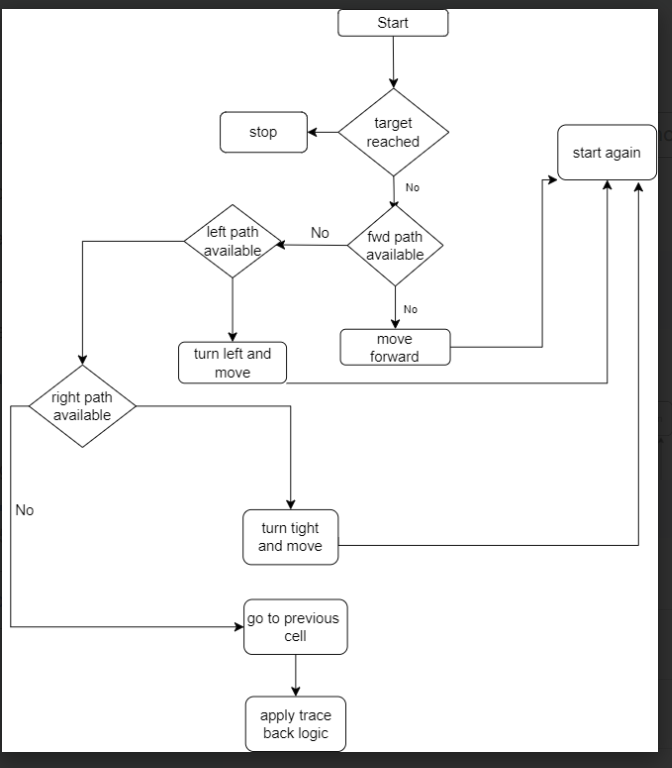
- Scoring System: Track the player's performance based on completion time, collected keys, and lives remaining.

- Level Unlocking: Unlock subsequent levels as the player completes previous ones.

- Save and Load: Implement the ability to save game progress and load it later.

**2. ARCHITECTURE**

**Illustration of The Maze MVP**



**3. APIS AND METHODS**

**- List and describe the API routes that you will be creating for your web client to communicate with your web server**

(a). User Registration

- POST /api/register: Allows the web client to send user registration data to the server for creating a new user account.

(b). User Login and Authentication:

- POST /api/login: Accepts user login credentials and returns an authentication token to the web client for subsequent authenticated requests.

- POST /api/logout: Logs out the user and invalidates the authentication token.

(c). User Profile

- GET /api/profile/:userId: Retrieves the profile information of a specific user identified by the userId.

- PATCH /api/profile/:userId: Allows the web client to update specific fields of the user's profile, such as contact information or professional background.

(d). User Directory

- GET /api/user: Retrieves a list of all registered user profiles.

- GET /api/userId: Retrieves the profile information of a specific user identified by the userId.

(e). Levels

- GET /api/levels: Retrieves a list of available levels.

- GET /api/level/:userId: Retrieves the specific level of a user identified by the userId.

**- List and describe any API endpoints or function/methods that you will be creating to allow any other clients to use.**

(a). Public API Endpoints

- GET /api/public/levels: Retrieves a list of public levels that are accessible to all clients, including non-authenticated users.

(b). Authentication and Authorization

- POST /api/authenticate: Accepts user credentials (username/password) and returns an authentication token for subsequent authenticated requests made by the client.

- GET /api/verify-token: Verifies the validity of an authentication token sent by the client and returns the associated user information.

(c). Analytics and Insights

- GET /api/analytics/user-activity/:userId: Retrieves analytics and activity data for a specific user, helping clients analyze user engagement and behavior within the game network.

**4. DATA MODEL**



**5. USER STORIES**

**Define 3-5 detailed user stories that will be satisfied when your MVP is complete.**

(a). As a developer, I can make the game movements simple , so that the player can navigate through the maze using arrow keys or touch controls.

(b). As a developer, I can make multiple levels of the game, so that the player can have multiple challenges per level.

(c). As a developer, I can make some areas of the maze locked, so that the player may be required to find keys scattered throughout the maze to unlock doors so as to progress to the next level.

(d). As a developer, I can make the user interface easy, so that the player can have options to start the game, access settings, view instructions, and exit.

**6. MOCKUPS**

