BỘ CÔNG THƯƠNG

TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP TP.HCM

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO DỰ ÁN

Unity Game + Server MongoDB Atlas

MÔN: Công nghệ mới trong phát triển ứng dụng CNTT

Giảng viên hướng dẫn: Ths.Nguyễn Ngọc Lễ

Nhóm: 23

Lớp: DHKHMT18A

TP. Hồ Chí Minh, Tháng 12 Năm 2025

Lời mở đầu

Trong bối cảnh các trò chơi không chỉ dừng lại ở giải trí đơn thuần mà còn là môi trường thử nghiệm công nghệ, việc xây dựng một game 2D có kết nối backend và lưu trữ dữ liệu trên nền tảng đám mây giúp sinh viên tiếp cận thực tế với các mô hình hệ thống hiện đại. Đề tài "Unity Game + Server MongoDB Atlas" được thực hiện với mục tiêu kết hợp giữa lập trình game client (Unity) và phát triển dịch vụ web (REST API) trên Node.js/Express, đồng thời vận dụng cơ sở dữ liệu NoSQL (MongoDB Atlas) trong bài toán đồng bộ hoá tiến trình người chơi.

Báo cáo này trình bày toàn bộ quá trình xây dựng hệ thống: từ thiết kế gameplay, kiến trúc client–server, thiết kế dữ liệu, API, đến các vấn đề kiểm thử, vận hành và triển khai. Thông qua đó, nhóm mong muốn minh hoạ một quy trình phát triển phần mềm tương đối hoàn chỉnh cho một ứng dụng game có kết nối mạng, qua đó củng cố kiến thức đã học về lập trình, cơ sở dữ liệu và công nghệ web.

Nhóm xin chân thành cảm ơn quý thầy cô Khoa Công Nghệ Thông Tin, đặc biệt là giảng viên hướng dẫn, đã hỗ trợ, góp ý và tạo điều kiện để nhóm hoàn thành đồ án này. Mặc dù đã nỗ lực, báo cáo khó tránh khỏi những thiếu sót; nhóm rất mong nhận được các ý kiến đóng góp để hoàn thiện hơn trong các nghiên cứu và sản phẩm sau này.

Danh sách thành viên và phân công

STT	MSSV	Họ và tên	Nhiệm vụ chi tiết
1	22634631	Phạm Đức Tài	Thiết kế và lập trình client Unity: xây dựng gameplay platformer (di chuyển, nhảy, dash, bắn, va chạm), tối ưu animation và vật lý nhân vật; hiện thực HUD (Lives/Score/Level) gắn với GameSession; thiết kế UI cho AuthScene, MainMenu, Settings và tích hợp luồng gọi API (đăng nhập/đăng ký, start/update/end session, level progress) vào game.
2	22639301	Lai Thanh Sĩ	Phát triển backend Node.js/Express: thiết kế và hiện thực các API auth, gameProfile, sessions, levelProgress, levels; xây dựng schema và chỉ mục cho MongoDB Atlas (users, gameProfiles, gameSessions, levels, levelProgress, leaderboard, achievements); viết script seed dữ liệu demo; cấu hình và triển khai server trên Render (biến môi trường, kết nối Atlas, kiểm tra healthcheck và log).
3	22639861	Hoàng Phi Hùng	Thực hiện kiểm thử end-to-end giữa client và server (auth, lưu tiến trình, leaderboard/achievements, quay về MainMenu và chơi lại), xây dựng kịch bản test chi tiết; thu thập và phân tích log lỗi UI/UX và API; chuẩn hoá giao diện (layout, font, thông báo lỗi); biên soạn và hoàn thiện báo cáo (kiến trúc, ERD, API, UI/Gameplay, Atlas/Render) và hỗ trợ cấu hình CI/CD, theo dõi chất lượng hệ thống.

Tóm tắt

Game platformer 2D phát triển bằng Unity, kết nối backend REST, lưu trữ trên MongoDB Atlas. Báo cáo mô tả kiến trúc client–server, gameplay, đồng bộ dữ liệu người chơi, thiết kế dữ liệu, API, UI/UX, kiểm thử, vận hành và kế hoạch triển khai.

Nội dung

1 Giới thiệu

1.1 Bối cảnh và mục tiêu

Trò chơi hành động đi cảnh 2D (platformer) với nhân vật chính vượt chướng ngại, bắn kẻ địch, thu thập xu và hoàn thành màn. Hệ thống có đăng nhập/đăng ký, lưu tiến trình và thống kê trên đám mây (MongoDB Atlas), hỗ trợ âm thanh SFX/BGM và thiết lập âm lượng. Mục tiêu:

- **Trải nghiệm mượt và an toàn dữ liệu**: điều khiển, âm thanh đầy đủ; dữ liệu được lưu cloud, chịu lỗi mạng ở mức hợp lý.
- **Đồng bộ đa thiết bị**: tiến trình và hồ sơ người chơi lưu trên server; client chỉ cache ngắn hạn.
- **Dễ mở rộng**: API rõ, mô hình dữ liệu tách bạch (User, GameProfile, GameSession, Level, LevelProgress), dễ bổ sung leaderboard/achievement.
- **Vận hành được**: có log, chỉ mục, backup Atlas, CI/CD cơ bản.

1.2 Phạm vi và đối tượng

Phạm vi:

- **Client Unity**: gameplay platformer 2D, HUD, âm thanh, đăng nhập/đăng ký, cài đặt âm thanh.
- **Server REST (Node/Express)**: JWT auth, CRUD hồ sơ/phiên/level/tiến trình, swagger docs, Jest test.
- **Dữ liệu**: MongoDB Atlas với các bộ sưu tập users, gameProfiles, gameSessions, levels, levelProgress; tuỳ chọn leaderboard/achievements.

Đối tượng: người chơi (end-user), QA (kiểm thử), dev (client/server), vận hành (DevOps/DBA).

1.3 Tài liệu tham khảo liên quan

Thiết kế hệ thống nội bộ, đặc tả API (swagger), tài liệu MongoDB Atlas (kết nối, index, backup), hướng dẫn triển khai CI/CD và Jest test.

2 Kiến trúc hệ thống

2.1 Tổng quan

Kiến trúc client–server tách biệt:

- **Client Unity**: xử lý gameplay, input, render, âm thanh; gọi API HTTP(S) để đăng nhập, start/end session, đồng bộ điểm/mạng/level.
- **API server (Express)**: định tuyến REST, xác thực JWT, validate (Joi), controller cho auth, profile, session, level, levelProgress.
- **Cơ sở dữ liệu**: MongoDB Atlas; model Mongoose; chỉ mục; backup và bảo mật qua Atlas.
- **Bảo mật**: JWT, authorizeUser để tránh truy cập chéo userId; CORS; biến môi trường cho secret/URI.

2.2 Client (Unity)

- Input: di chuyển, nhảy, bắn, leo thang, dash.
- Gameplay manager: `GameSession` quản lý mạng, lives, score, levelId, đồng bộ định kỳ; `LevelExit` điều phối chuyển màn.
- UI: AuthScene (Login/Register/MainMenu), HUD (Lives, Score, Level), Settings (âm thanh).
- Âm thanh: `AudioManager` singleton, SFX (jump, shoot, coin, death, game over, level exit, click), BGM; thiết lập âm lượng (MusicVolume, SFXVolume, IsMuted) lưu PlayerPrefs; tự pause BGM khi volume = 0.
- Khôi phục HUD: khi text bị xoá (ResetAll), `GameSession` tìm lại (EnsureTextReference) sau khi scene load.

2.3 Server / API

- Auth: `/api/auth/register`, `/api/auth/login`; JWT; cập nhật `lastLoginAt`.
- GameProfile: GET/PUT profile; add score/coins; increment death; update lives; add playtime.
- Sessions: start (`POST /api/sessions`), update (`PUT /:sessionId`), end (`POST /:sessionId/end`), history.
- LevelProgress: get/update/complete per level; lưu bestScore, bestTime, playCount.
- Levels: danh sách/chi tiết level (levelNumber, sceneName, requiredScoreToUnlock).
- Middleware: `authenticateToken`, `authorizeUser`; validate bằng Joi (utils/validators).

2.4 Cơ sở dữ liệu (MongoDB Atlas)

Atlas cluster; Mongoose schema:

- User: email/username unique, password hash (Bcrypt), isActive, lastLoginAt, timestamps.
- GameProfile: thống kê tổng; currentLives, currentLevel (ref Level).
- GameSession: theo level; sessionStatus (ACTIVE/COMPLETED/ABANDONED/FAILED), isCompleted.
- Level: levelNumber unique, sceneName, difficulty, maxCoins, maxEnemies, requiredScoreToUnlock.
- LevelProgress: userId + levelId unique; bestScore, bestTime, playCount, completedAt.

Chỉ mục chính: users(email, username), gameProfiles(userId), gameSessions(userId, levelId, sessionStatus), levels(levelNumber), levelProgress(userId, levelId).

2.5 Triển khai và cấu hình

- Atlas: SRV URI, user theo role, IP whitelist; backup snapshot/PITR; theo dõi index health.
- Server: env `MONGODB_URI`, `DB_NAME`, `JWT_SECRET`, `PORT`; CORS; nodemon cho dev; Jest test; swagger doc.
- Client: cấu hình base URL API, timeout, retry ngắn khi start/end session; fallback nếu API null; pause BGM khi volume 0.
- Graceful shutdown: lắng nghe SIGTERM/SIGINT, đóng server sạch.

3 Thiết kế gameplay

3.1 Cốt lõi gameplay

Gameplay là một platformer 2D dạng side–scrolling:

- Người chơi điều khiển nhân vật chính di chuyển qua các platform, né chướng ngại, nhảy qua hố, leo thang, dash và bắn kẻ địch.
- Mỗi màn (Level) có số lượng coin và kẻ địch nhất định; người chơi thu thập coin để tăng điểm, bắn/né kẻ địch để tiến tới cổng thoát (LevelExit).
- Nhân vật có số mạng (lives) giới hạn; va chạm với kẻ địch hoặc bẫy (layer `Enemies`, `Hazards`) sẽ chết, trừ mạng hoặc game over.
- Hệ thống GameSession ghi nhận số coin, kẻ địch đã hạ, số lần chết, điểm và thời gian chơi cho mỗi level.

3.2 Level và tiến trình

Mỗi level là một scene riêng trong Unity, được ánh xạ với một bản ghi `Level` trong server:

- **Cấu trúc Level**: platform, cầu thang (layer `Climbing`), bẫy (layer `Hazards`), kẻ địch (`EnemyMovement`), coin (`CoinPickup`), cổng thoát (`LevelExit`).
- **Kẻ địch**: di chuyển qua lại với tốc độ cố định, đảo chiều khi ra khỏi trigger (OnTriggerExit2D), lật sprite theo hướng di chuyển.
- **Coin**: khi nhân vật đi qua, CoinPickup cộng điểm (`pointsForCoinPickup`), tăng bộ đếm coin trong GameSession, phát âm thanh rồi phá huỷ coin.
- **Thoát màn**: khi nhân vật chạm cổng `LevelExit` và không trong trạng thái loading, LevelExit kích hoạt coroutine `LoadNextLevel`, set cờ `IsLoading`, gửi EndSession(status = COMPLETED) và chuyển sang scene tiếp theo (hoặc về main menu tuỳ logic unlock).
- **Tiến trình**: server dùng LevelProgress để ghi lại bestScore, bestTime, playCount và isCompleted cho từng level của từng user; GameProfile cập nhật currentLevel và tổng thống kê.

3.3 Điều khiển và tương tác

Điều khiển dùng Input System mới của Unity (InputValue trong `PlayerMovement`):

- **OnMove**: nhận vector 2D từ phím/analog, nhân với `moveSpeed` để tạo velocity theo trục X; animator set `isRunning` khi vận tốc ngang khác 0.
- **OnJump**: chỉ cho phép khi collider chân (`myFeetCollider`) chạm layer `Ground`; cộng thêm vector vận tốc theo trục Y (`jumpSpeed`); phát SFX nhảy.
- **ClimbLadder**: nếu collider chân chạm layer `Climbing`, tắt gravity và cho phép di chuyển theo trục Y với `climbSpeed`, đồng thời bật animation leo.
- **OnDash**: kiểm tra cooldown, trạng thái đang dash; khi kích hoạt, coroutine `Dash` đặt gravity = 0, đẩy nhân vật theo hướng đang quay (`dashSpeed`) trong một khoảng thời gian ngắn, sau đó khôi phục gravity.

- **OnAttack**: bắn đạn bằng cách `Instantiate` prefab Bullet tại vị trí súng (gun), hướng theo chiều nhân vật quay; phát SFX bắn.
- **Bullet**: xác định tốc độ theo hướng nhân vật tại thời điểm bắn, di chuyển thẳng; khi va chạm Enemy, cộng bộ đếm enemiesDefeated trong GameSession rồi phá huỷ Enemy và Bullet.
- **Chết**: nếu collider thân chạm layer `Enemies` hoặc `Hazards`, set `isAlive = false`, trigger animation `Dying`, đẩy nhân vật bằng `deathKick`, sau đó gọi `GameSession.ProcessPlayerDeath()` (retry nhiều frame nếu chưa tìm thấy GameSession).
- **HUD**: hiển thị mạng (Lives Text), điểm (Score Text), tên level (Level Text); GameSession là singleton `DontDestroyOnLoad`, mỗi lần scene gameplay load sẽ tìm lại các text này theo name nếu bị destroy.

3.4 Âm thanh và nhạc nền

- **SFX**:
  - Jump: gọi `AudioManager.PlayJump()` trong `OnJump`.
  - Shoot: `PlayShoot()` trong `OnAttack`.
  - Coin: `PlayCoinPickup()` trong `CoinPickup`.
  - Level exit: `PlayLevelExit()` khi player chạm `LevelExit`.
  - Death: `PlayDeath()` trong `ProcessPlayerDeath()`.
  - Game over: `PlayGameOver()` khi hiện GameOver modal.
  - Button click: `PlayButtonClick()` qua `UIButtonClickSound` và `SoundSettingsButton`.
- **Nhạc nền (BGM)**:
  - `AudioManager` có `musicSource` loop, `backgroundMusicClip`; `PlayBackgroundMusic()` được gọi sau khi login/register thành công.
  - *MusicVolume* là volume chính (thay thế master), SFX volume là nhân thêm cho SFX; khi MusicVolume = 0, nhạc bị pause (và unpause khi > 0).
  - Slider trong SoundSettingsPanel chỉnh MusicVolume và SFXVolume theo thời gian thực, lưu PlayerPrefs; toggle mute set `IsMuted`.

4 Mạng và đồng bộ

4.1 Luồng đăng nhập / đăng ký

Luồng AuthScene:

- **Đăng ký** `POST /api/auth/register`: gửi {username, email, password}; nhận JWT + user; lưu token, chuyển MainMenu.
- **Đăng nhập** `POST /api/auth/login`: gửi {email, password}; nhận JWT + user; cập nhật `lastLoginAt`; lưu token.
- Sau đăng nhập/đăng ký: bật nhạc nền (AudioManager.PlayBackgroundMusic), load profile hiện tại, hiển thị MainMenu.

4.2 Đồng bộ hồ sơ người chơi

- **GET /api/gameProfile/:userId**: tải lives, totalScore, currentLevel,... để thiết lập HUD và logic game.
- **Update currentLevel**: khi StartSession, GameSession gọi server cập nhật currentLevel (qua LevelProgressManager/Level) trước khi chơi.
- **Sync định kỳ**: GameSession có timer SYNC_INTERVAL để gửi stats (score, coins, enemies, deaths, lives) lên server.

4.3 Lưu trữ tiến trình (cloud save)

- **Start session** `POST /api/sessions`: tạo phiên với userId, levelId; server trả sessionId; client cache để update.
- **Update session** `PUT /api/sessions/:sessionId`: gửi delta (score, coins, enemiesDefeated, deathCount, livesRemaining).
- **End session** `POST /api/sessions/:sessionId/end`: chốt trạng thái (COMPLETED/FAILED/ABANDONED), duration.
- **Complete level** `POST /api/levelProgress/:userId/:levelId/complete`: đánh dấu hoàn thành, cập nhật bestScore/bestTime/playCount/isCompleted; đồng bộ GameProfile (totalScore,...).

4.4 Sơ đồ luồng đăng nhập / phiên chơi (mô tả)

Luồng đăng nhập/đăng ký:

- Người chơi nhập thông tin → AuthScene gọi `/api/auth/login` hoặc `/api/auth/register`.
- Server xác thực, trả JWT + user, cập nhật `lastLoginAt`.
- Client lưu token, bật nhạc nền, tải profile (currentLives, currentLevel, totalScore), chuyển MainMenu.

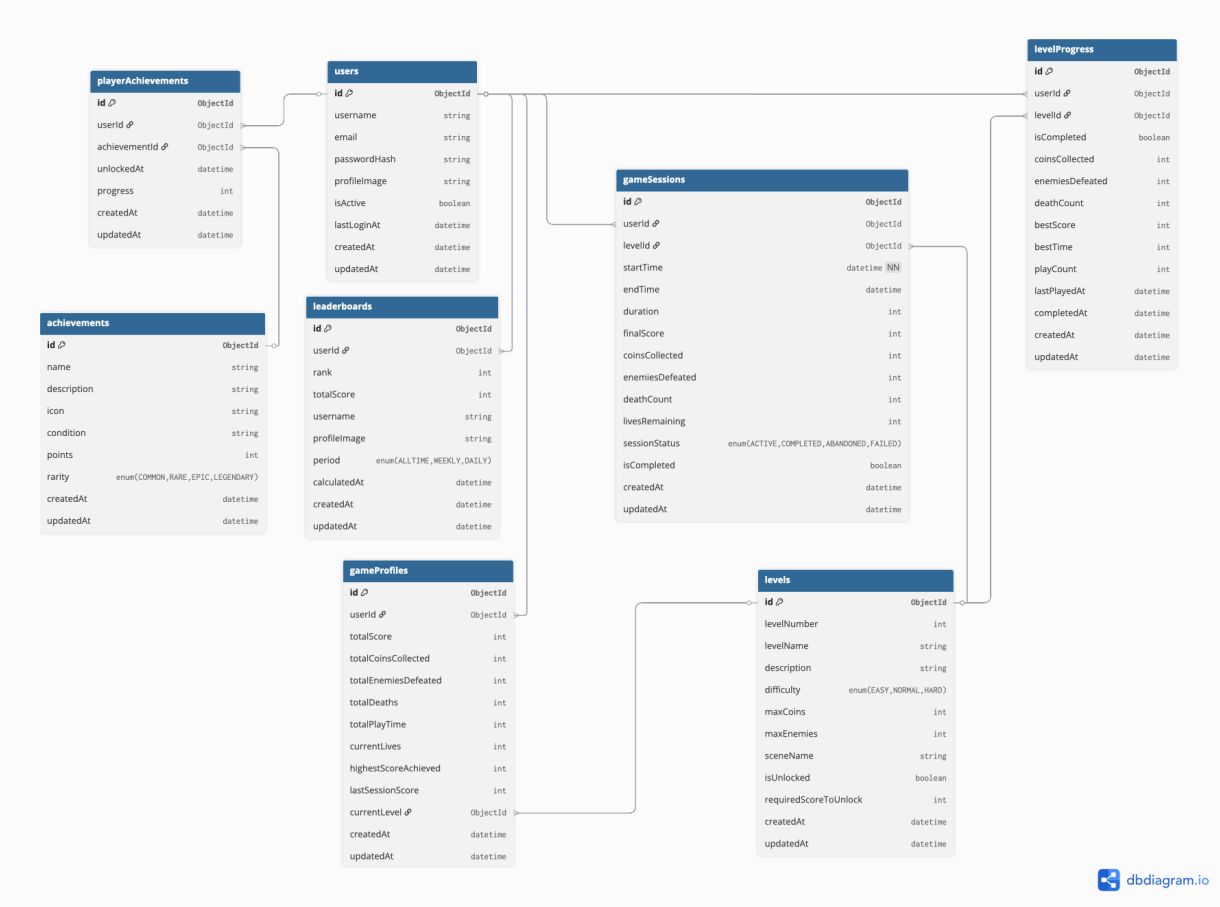
Luồng phiên chơi (start → sync → end):

- *Start*: Chọn level → `POST /api/sessions` (userId, levelId) → sessionId; tải/khởi tạo LevelProgress.
- *Sync trong phiên*: định kỳ `PUT /api/sessions/:sessionId` gửi score, coins, enemiesDefeated, deathCount, livesRemaining.
- *Complete level*: `POST /api/levelProgress/:userId/:levelId/complete` cập nhật bestScore/bestTime/playCount/isCompleted.
- *End*: `POST /api/sessions/:sessionId/end` với trạng thái (COMPLETED/FAILED/ABANDONED) và duration; nếu mất kết nối có thể bỏ qua end.

4.5 Xử lý lỗi và retry

- Client kiểm tra null API: nếu thiếu LevelProgressManager/AchievementManager, log warning và bỏ qua.
- Một số thao tác quan trọng (tìm GameSession khi chết, cập nhật level text) có cơ chế retry vài frame.
- Khi network lỗi nặng: fallback reload scene hiện tại; EndSession có thể bỏ qua nếu không có sessionId.
- BGM/SFX không phụ thuộc mạng; HUD được khôi phục local nếu UI bị destroy.

5 Sơ đồ ERD



Hình 1: Sơ đồ ERD của hệ thống

6 Thiết kế dữ liệu (MongoDB)

6.1 Lược đồ chính

- **users**: lưu thông tin tài khoản – _id, username (duy nhất, 3–20 ký tự), email (duy nhất, regex kiểm tra định dạng), passwordHash (được hash bằng Bcrypt, không trả về mặc định), profileImage, isActive, lastLoginAt, timestamps. Chỉ mục trên `email`, `username`.
- **gameProfiles**: thống kê tổng cho mỗi người chơi – userId (uniqe, ref User), totalScore, totalCoinsCollected, totalEnemiesDefeated, totalDeaths, totalPlayTime (giây), currentLives, highestScoreAchieved, lastSessionScore, currentLevel (ref Level), timestamps. Chỉ mục trên `userId`.
- **gameSessions**: log phiên chơi từng level – userId, levelId (ref Level), startTime, endTime, duration, finalScore, coinsCollected, enemiesDefeated, deathCount, livesRemaining, sessionStatus (ACTIVE/COMPLETED/ABANDONED/FAILED), isCompleted, timestamps. Chỉ mục trên `userId`, `levelId`, `sessionStatus`.
- **levels**: định nghĩa level – levelNumber (duy nhất, ≥ 1), levelName, description, difficulty (EASY/NORMAL/HARD), maxCoins, maxEnemies, sceneName (tên scene Unity), isUnlocked, requiredScoreToUnlock, timestamps. Chỉ mục trên `levelNumber`.
- **levelProgress**: tiến trình theo từng level – userId, levelId, isCompleted, coinsCollected, enemiesDefeated, deathCount,

bestScore, bestTime, playCount, lastPlayedAt, completedAt, timestamps. Chỉ mục kết hợp `userId, levelId` (duy nhất), cùng các chỉ mục đơn trên userId/levelId.

- (Tuỳ chọn) **leaderboards**, **achievements**, **playerAchievements** theo mô hình đã có trong code server để phục vụ xếp hạng và thành tích.

6.2 Ràng buộc và chỉ mục

Các ràng buộc chính:

- Unique trên `users.email`, `users.username`, `gameProfiles.userId`, `levelProgress(userId, levelId)`.
- Giá trị số không âm (min: 0) cho điểm, coin, kẻ địch, số lần chết, thời gian.
- Khoá ngoại logic qua `ref` tới `User` và `Level` (được kiểm tra ở tầng API).

7 API chính

7.1 Xác thực / phân quyền

Nhóm API `/api/auth`:

- **POST /api/auth/register**: đăng ký tài khoản mới (username, email, password); trả về JWT + thông tin user.
- **POST /api/auth/login**: đăng nhập bằng email/password; trả về JWT + thông tin user; cập nhật `lastLoginAt`.

Middleware `authenticateToken` kiểm tra JWT ở header; `authorizeUser` đảm bảo user chỉ truy cập tài nguyên của chính mình (so sánh `:userId` với payload token).

7.2 Quản lý người chơi

Nhóm API `/api/gameProfile`:

- **GET /api/gameProfile/:userId**: lấy GameProfile (tổng điểm, coin, kẻ địch, deaths, currentLives, currentLevel, v.v.).
- **PUT /api/gameProfile/:userId**: cập nhật profile (cho phép server-side hợp nhất dữ liệu).
- **POST /api/gameProfile/:userId/score**: cộng dồn điểm.
- **POST /api/gameProfile/:userId/coins**: cộng dồn coin.
- **POST /api/gameProfile/:userId/death**: tăng số lần chết.
- **PUT /api/gameProfile/:userId/lives**: cập nhật số mạng hiện tại (currentLives).
- **POST /api/gameProfile/:userId/playtime**: cộng dồn thời gian chơi.

7.3 Tiến trình / thành tích

Nhóm API `/api/sessions`:

- **POST /api/sessions**: startSession – tạo GameSession mới cho userId + levelId.
- **PUT /api/sessions/:sessionId**: updateSession – cập nhật điểm, coin, enemies, deaths, livesRemaining trong khi chơi.
- **POST /api/sessions/:sessionId/end**: endSession – chốt kết quả, set `sessionStatus` và duration.
- **GET /api/sessions/:userId/history**: trả về lịch sử session của một người chơi.

Nhóm API `/api/levelProgress`:

- **GET /api/levelProgress/:userId**: lấy toàn bộ tiến trình level của user.
- **GET /api/levelProgress/:userId/:levelId**: lấy tiến trình cụ thể một level.
- **PUT /api/levelProgress/:userId/:levelId**: cập nhật thống kê level (coins, enemies, deaths, bestScore, bestTime, playCount).
- **POST /api/levelProgress/:userId/:levelId/complete**: đánh dấu level hoàn thành, cập nhật isCompleted và completedAt.

Nhóm API `/api/levels`:

- **GET /api/levels**: trả danh sách level (phục vụ client mapping sceneName ↔ levelId).
- **GET /api/levels/:levelId**: chi tiết một level (tên, số level, độ khó, điểm yêu cầu để mở khoá).

Các nhóm `/api/leaderboard`, `/api/achievements` (nếu dùng) phục vụ xếp hạng và thành tích.

7.4 Giám sát và logging

Log lỗi API; metric tần suất start/end session, tỉ lệ lỗi; lưu client warnings nếu cần.

8 Giao diện người dùng

8.1 Luồng màn hình

AuthScene: **Login / Register** → **Main Menu** → **Play** → **Level scene**; **Settings** (âm thanh) là một modal phủ lên (ẩn các panel khác khi mở).

8.2 Màn hình đăng nhập / đăng ký

- Login: form email/password; nút Login gọi API; statusText hiển thị lỗi; loadingOverlay khóa thao tác trong lúc chờ.
- Register: form username/email/password; nút Register; sau khi đăng ký thành công vẫn ở trạng thái login (theo flow hiện tại), có thể điều hướng về Login; statusText/overlay tương tự.
- Chuyển đổi panel: nút “Go Register” và “Back to Login” ẩn/hiện các panel tương ứng.

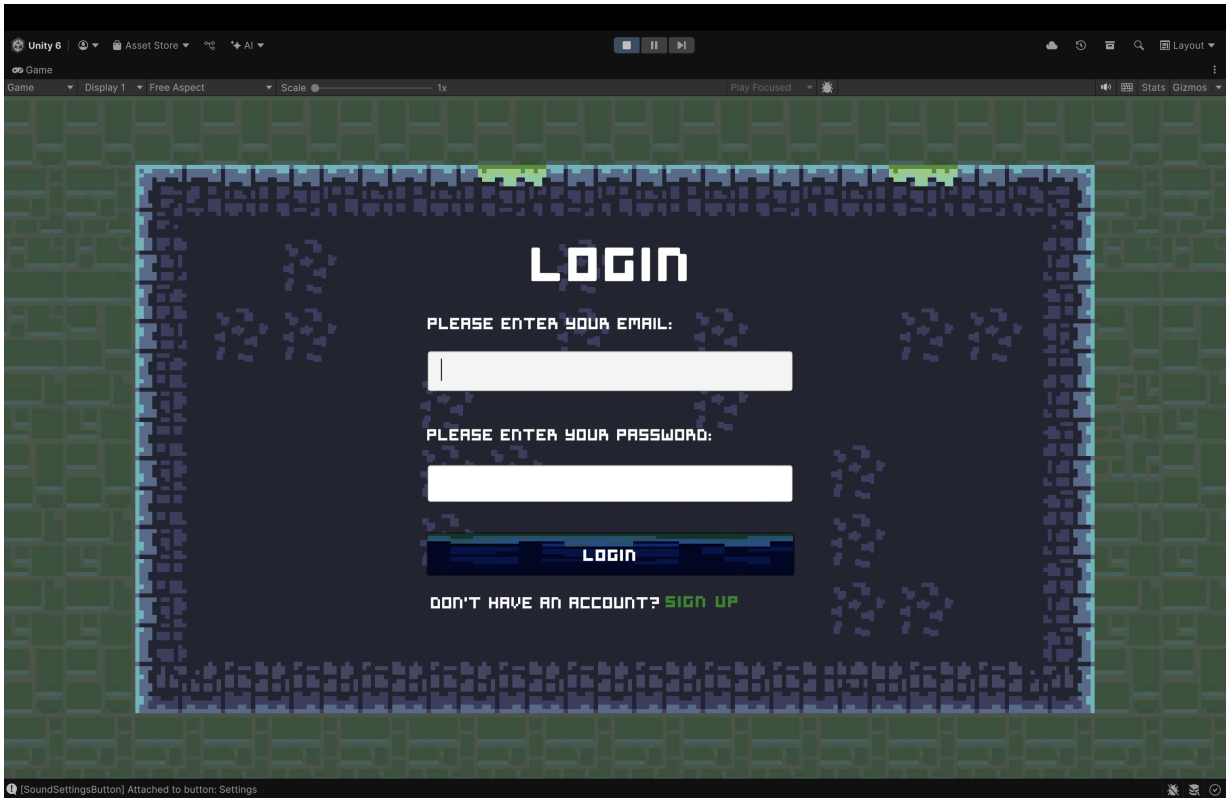
8.3 HUD trong game

- Hiển thị Lives, Score, Level (và Score Icon). Khi quay về main menu, các text này có thể bị destroy; GameSession (DontDestroyOnLoad) sẽ tìm lại theo name (“Lives Text”, “Score Text”, “Level Text”, “Score Icon”) sau khi scene gameplay load.
- Cập nhật liên tục: livesText/scoreText được cập nhật mỗi lần thay đổi; levelText được cập nhật sau khi resolve levelId từ server.

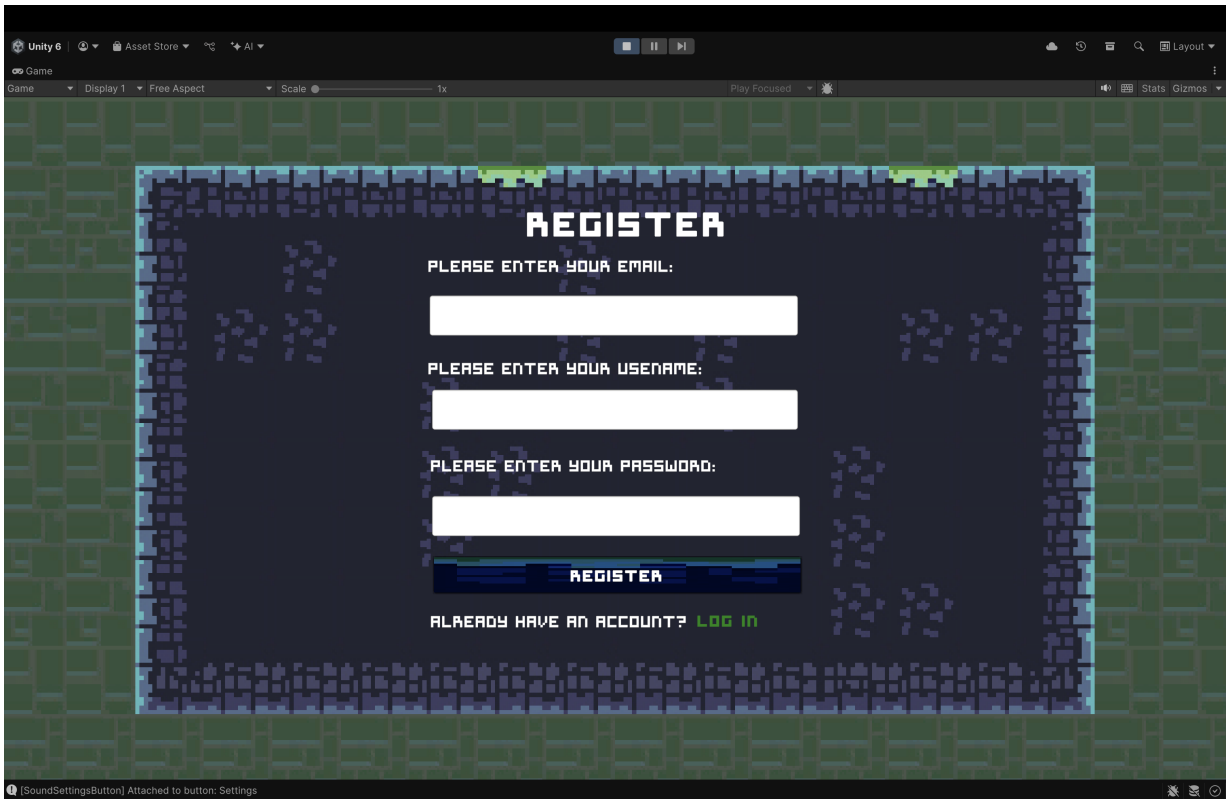
8.4 Cài đặt âm thanh

- Modal Settings: dùng SoundSettingsPanel; khi mở sẽ ẩn các panel khác (login/register/main menu), khi đóng sẽ khôi phục panel phù hợp (main menu nếu đã login, login nếu chưa).
- Slider MusicVolume: điều chỉnh âm lượng chính (ảnh hưởng BGM và nhân với SFX); SFXVolume nhân thêm cho SFX; Mute toggle đặt âm lượng về 0 logic.
- Lưu PlayerPrefs: MusicVolume, SFXVolume, IsMuted; áp dụng real-time; nếu MusicVolume = 0 thì BGM pause; >0 thì unpause.
- Button click: tất cả button có thể gắn UIButtonClickSound; riêng Settings button dùng SoundSettingsButton để mở panel.

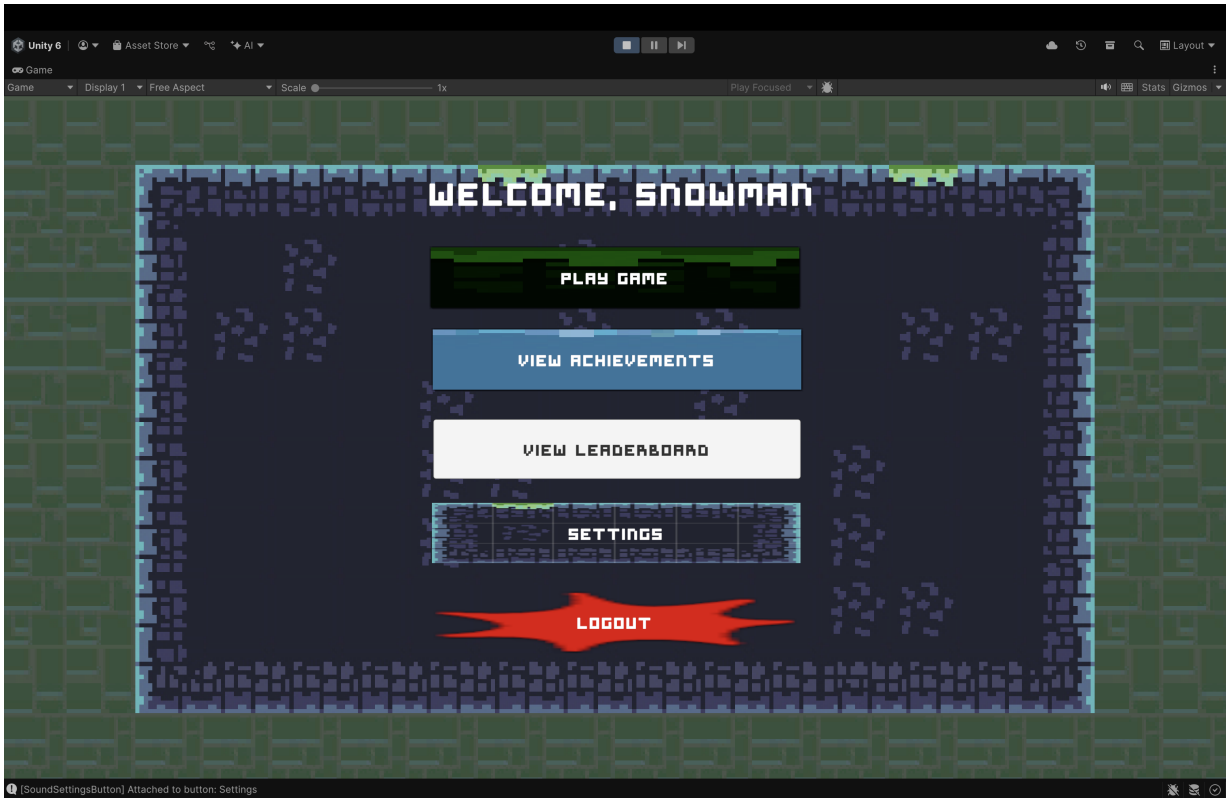
8.5 Hình minh hoạ giao diện



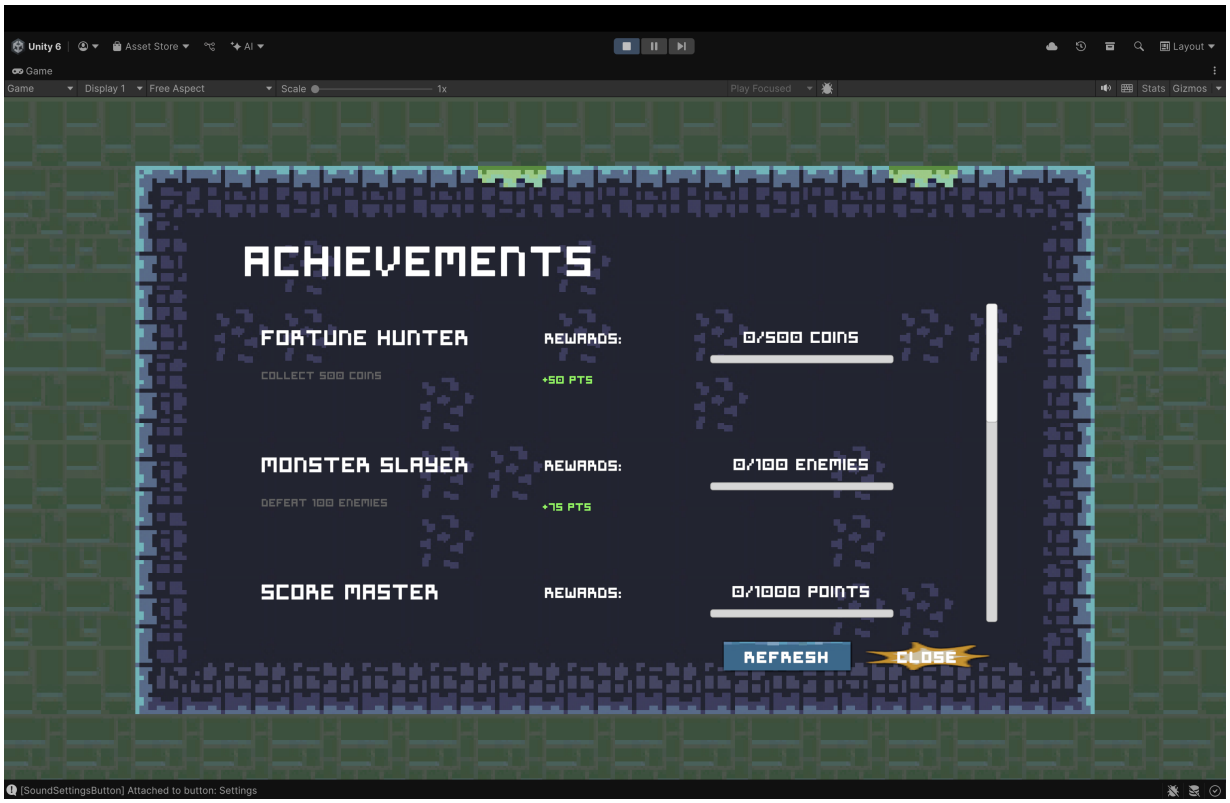
Hình 2: Màn hình Login



Hình 3: Màn hình Register

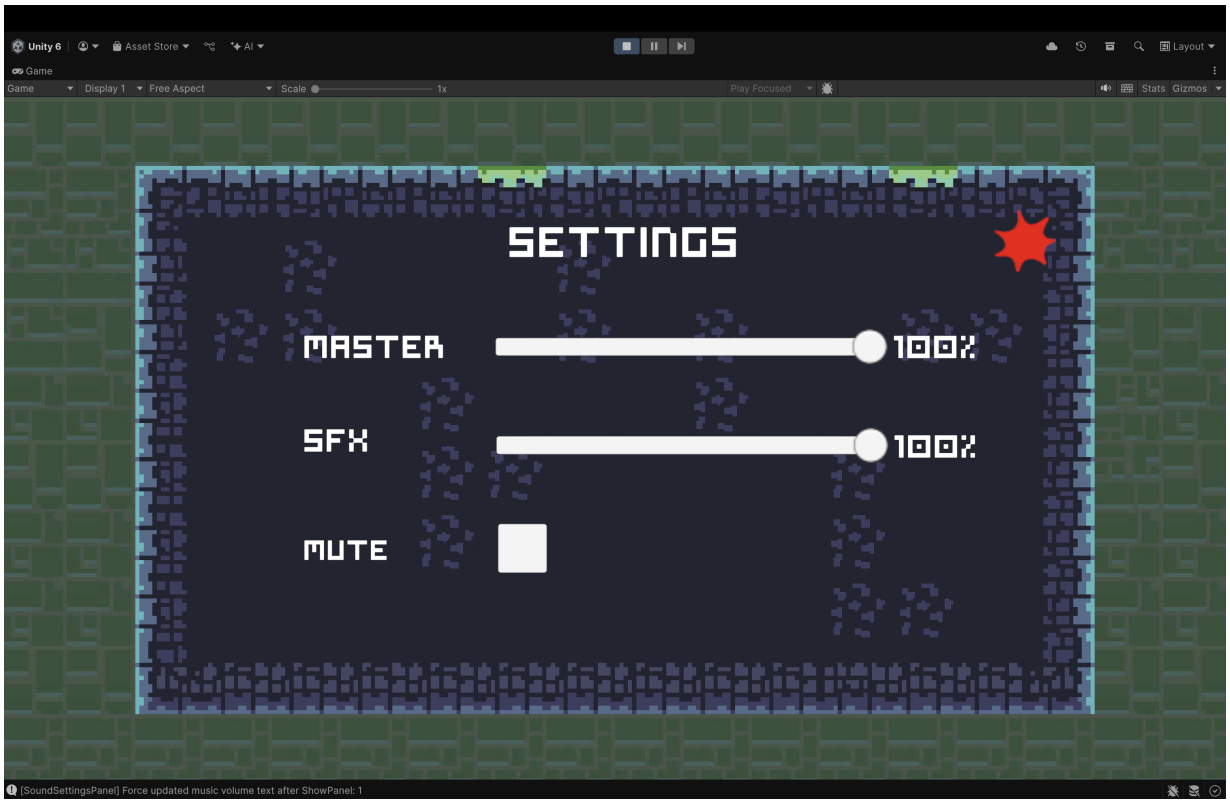


Hình 4: Main Menu

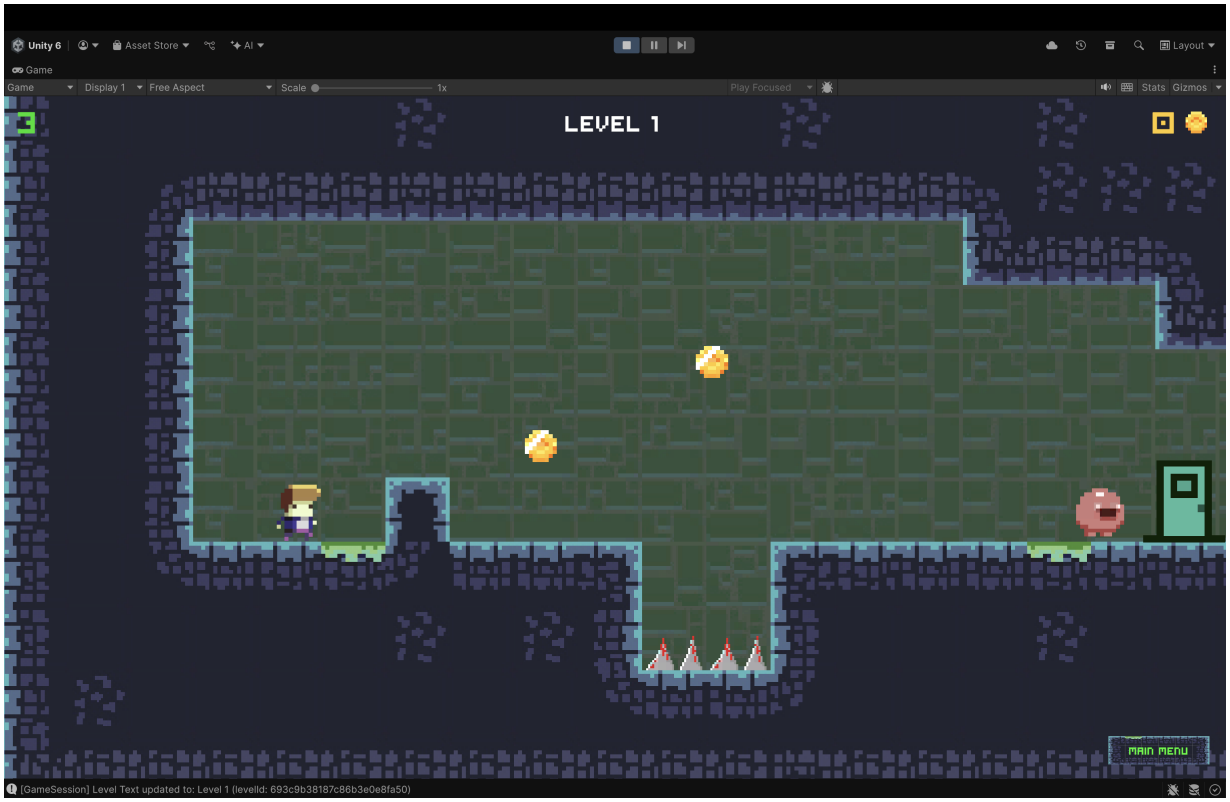


Hình 5: Màn hình Achievements

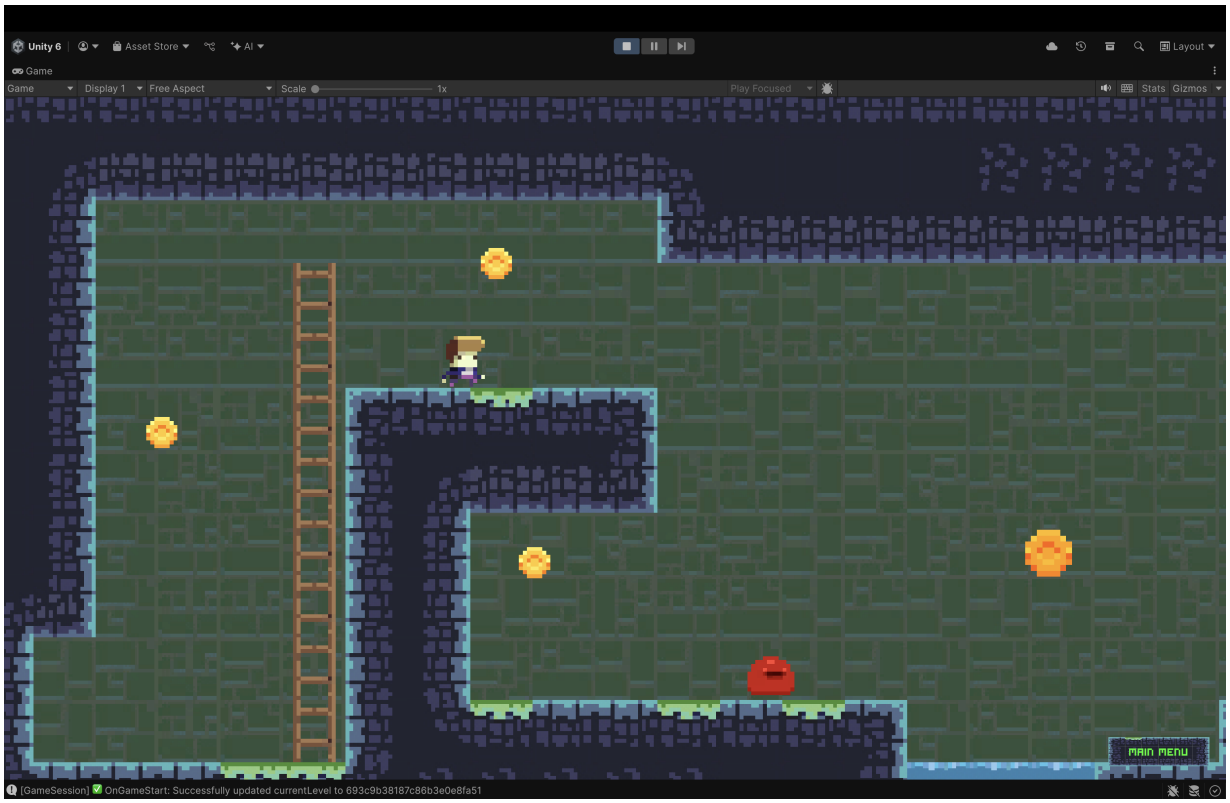
Hình 6: Màn hình Leaderboard



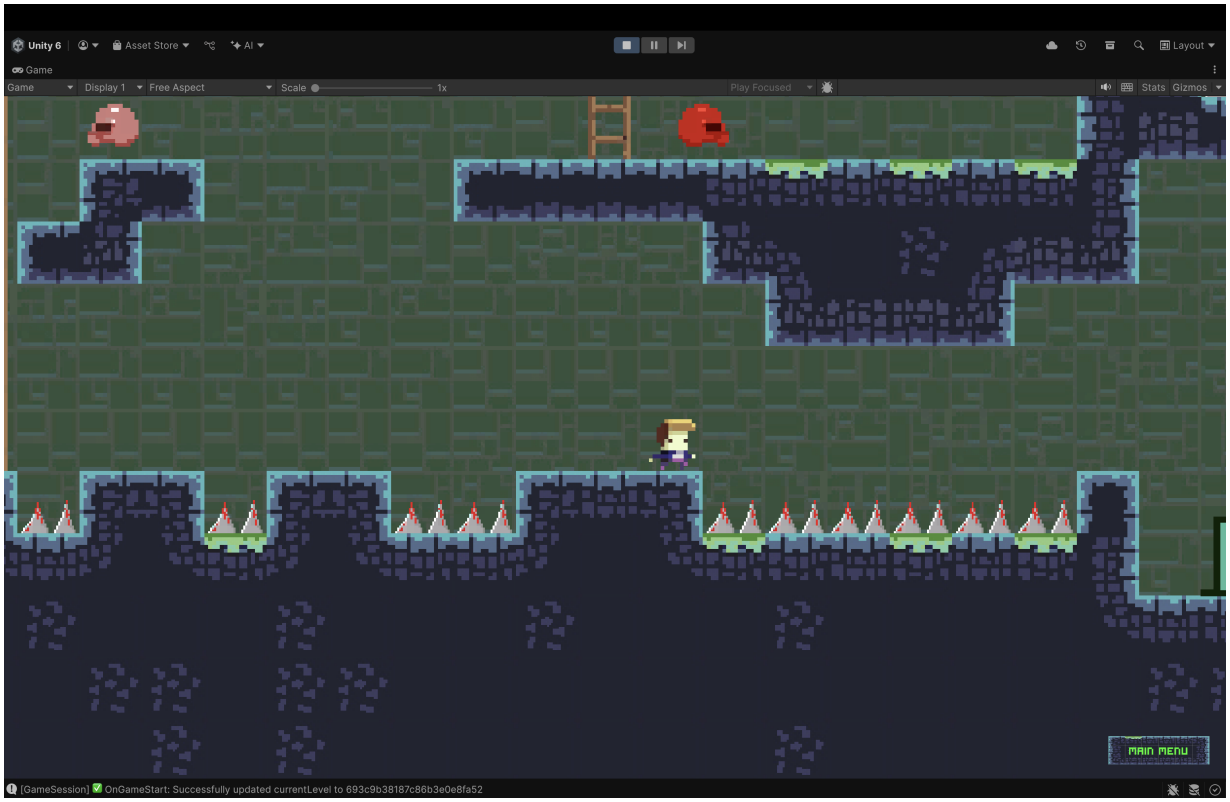
Hình 7: Màn hình Settings âm thanh



Hình 8: Gameplay Level 1



Hình 9: Gameplay Level 2



Hình 10: Gameplay Level 3

9 Vận hành / triển khai

9.1 CI/CD

- Server: pipeline build/test (Jest), lint/validate; deploy Node/Express (commonjs) lên Render (web service). Có script seed (utils/seedDB) và leaderboard (utils/calculateLeaderboard).
- Biến môi trường Render: `MONGODB_URI`, `DB_NAME`, `JWT_SECRET`, `PORT` (Render tự cấp); base URL API công khai để Unity gọi.
- Client: build Unity; (khuyến nghị) tự động hóa đóng gói build cho target platform; có thể cấu hình step gọi API healthcheck sau deploy.

9.2 Cấu hình MongoDB Atlas

- Biến môi trường: `MONGODB_URI` (SRV), `DB_NAME`, user Atlas tối thiểu quyền; IP whitelist (thêm outbound IP của Render).
- Atlas backup: snapshot/PITR; giám sát index, CPU/Memory, kết nối; dùng chỉ mục đã khai báo trong schema.
- Quy hoạch dữ liệu: userId/levelId là ObjectId; levelNumber unique; levelProgress unique (userId, levelId).

10 Kết luận

Dự án đã xây dựng được một hệ thống game Unity 2D kết nối server riêng, lưu trữ trên MongoDB Atlas với kiến trúc client–server rõ ràng. Về phía client, gameplay platformer được hiện thực đầy đủ (di chuyển, nhảy, leo, dash, bắn, thu coin, kẻ địch, HUD, âm thanh SFX/BGM, cài đặt âm lượng), đồng thời tích hợp luồng đăng nhập/đăng ký và đồng bộ tiến trình với backend. Về phía server, API REST (Node.js/Express) cùng các schema MongoDB (User, GameProfile, GameSession, Level, LevelProgress) cho phép lưu trữ hồ sơ, phiên chơi, tiến trình level một cách có tổ chức, dễ mở rộng (leaderboard, achievements).

Hệ thống đã đáp ứng các mục tiêu đề ra: dữ liệu người chơi được bảo toàn trên cloud, có khả năng đồng bộ đa thiết bị, có cơ chế xử lý lỗi và fallback, hỗ trợ kiểm thử và vận hành (Jest, swagger, backup Atlas). Tuy nhiên, vẫn còn không gian cải tiến: tối ưu thêm UI/UX (menu, feedback trong game), mở rộng phân tích thống kê (dashboard server), tăng cường bảo mật (rate limiting, refresh token), tích hợp thêm kênh phân phối build client. Trong tương lai, dự án có thể phát triển thêm nhiều chế độ chơi, hệ thống nhiệm vụ/thành tích sâu hơn, cũng như tích hợp thêm dịch vụ bên ngoài (ví dụ thanh toán, social login) để hoàn thiện trải nghiệm người dùng.