Project Report: DeathRoom Online

Project description

Project name: DeathRoom Online
Code repository: `https://github.com/IU-Capstone-Project-2025/DeathRoom`

DeathRoom Online is a fast-paced arena shooter inspired by Quake and Call of Duty, designed for competitive multiplayer gaming. The project addresses the growing demand for skill-based, responsive FPS games that prioritize low-latency gameplay over complex progression systems.
Problem Statement: Modern arena shooters often sacrifice network responsiveness for graphical fidelity. Our solution combines precise movement mechanics with robust server validation, ensuring fair and responsive gameplay in 12-player matches.

Team Members and Assigned Roles

- **Project Lead, Backend - Frontend Developer**: [Vsevolod Nazmudinov] – Oversees project direction.
- **Backend Developer, Network Engineer**: [Igor Kuzmenkov] - handles UDP implementation and network optimization.
- **Backend Developer**: [Slava Molchanov] - implements core game logic.
- **Client Part**: [Ilyas Khatipov] – Responsible for the client parts.
- **Game Designer, UI/UX**: [Gleb Lobov] – Creates models in Blender and manages animations.
- **Backend Developer, DevOps Engineer**: [Rodion Krainov] – Manages Docker deployment and server infrastructure.
- **Backend Developer, Reporter**: [Almas Bagishaev] - Responsible for reports.

Chosen Project Idea

We are developing a fast-paced arena shooter inspired by **Quake** and **Call of Duty**, featuring:

- 12-player multiplayer matches
- Competitive leaderboards and match statistics

Basic requirements

Target users

- Competitive players: Need sub-50ms latency
- Esports enthusiasts: Require spectator tools
- Casual players: Want quick matchmaking

Target Users

- Competitive FPS players who value responsiveness
- Esports enthusiasts looking for pure skill-based gameplay
- Casual players who enjoy fast-paced action

High-Level User Stories & Initial Scope

User Stories

- As a player, I want to join matches quickly so I can play without long waiting times.
- As a competitor, I need responsive controls so that my actions match what I see on screen.
- As a spectator, I want to watch ongoing matches so that I can learn from better players.
- As an admin, I need server metrics so I can monitor performance.

Initial Scope

- Core movement (strafe jumping, crouch sliding)
- weapon types (long range, projectile)
- arena maps
- Basic matchmaking
- Leaderboard system

Chosen Tech Stack with Justification

Backend

- **.NET + LiteNetLib/ENet** – Provides high-performance networking suitable for FPS games.
- **Photon Fusion (Server-authority)** - Ensures fair game by preventing client-side cheating.
- **EF Core + PostgreSQL** – Reliable data storage for player stats and leaderboards.
- **Redis** – Enables horizontal scaling when adding more game servers.

Frontend

- **Unity 6** – Latest engine version with improved networking and performance.
- **Blender + Mixamo** – Fast pipeline for creating and animating 3D models.

Infrastructure

- **Docker** – Containerization for consistent deployment across environments.
- **UDP Protocol** – Essential for low-latency gameplay in fast-paced shooters.