Realm War

Final Year Project

Session 2020-2024

A project submitted in partial fulfillment of the degree of

BS Gaming and Multimedia

Department of Gaming & Multimedia

The Superior University, Lahore

Spring 2024

Type (Nature of project)	[✓] **D**evelopment [ ] **R**esearch [ ] **R**&**D**
Area of specialization	
FYP ID	FYP-BGMM-F23-009

Project Group Members

Sr.#	Reg. #	Student Name	Email ID	*Signature
(i)	Bgmm-f20-022	Anas Shaikh	bgmm-f20-022@superior.edu.pk	
(ii)	Bgmm-f20-021	Muhammad Abrar	bgmm-f20-021@superior.edu.pk	
(iii)	Bgmm-f20-036	Aun Akbar	bgmm-f20-036@superior.edu.pk	

*The candidates confirm that the work submitted is their own and appropriate credit has been given where reference has been made to work of others

Plagiarism Free Certificate

This is to certify that, I Anas Shaikh S/D of Shaikh Ahsan Ullah, group leader of FYP under registration no Bgmm-f20-022 at Gaming and Multimedia Department, The Superior University, Lahore. I declare that my FYP report is checked by my supervisor.

Date: Name of Group Leader: Anas Shaikh Signature: _____

Name of Supervisor: Hafiza Maria Co-Supervisor: Ms. Amna

Designation: Lecturer Designation: Lecturer

Signature: _____ Signature: _____

HoD: Dr. Irfan Ud Din

Signature: _____

Realm War

Change Record

Author(s)	Version	Date	Notes	Supervisor's Signature
Anas Shaikh	0.5	5/9/2023	Concept for the Games core mechanics	
Anas Shaikh	1.3	17/12/2023	First Test Build	
Anas Shiakh	1.5	24/12/2023	Smoothing out the Core Mechanics and Smoothing out movement	
Aun Akbar	1.9	03/1/2024	Shooting Mechanics Part-1	
Aun Akbar/ Muhammad Abrar	2.1	11/1/2024	Shooting Mechanics Part-2	
Anas Shaikh / Muhammad Abrar	2.3	15/1/2024	UI/XI Assets and Build Part-1	
Anas Shaikh / Muhammad Abrar	2.5	02/2/2024	UI/XI Assets and Build Part-2	
Anas Shaikh / Aun Akbar	3.0	09/2/2024	Play Test Build	
Anas Shaikh	3.4	13/2/2024	3D	

APPROVAL

PROJECT SUPERVISOR

Comments: _____

_____

Name:_____

Date:_____ Signature:_____

PROJECT MANAGER

Comments: _____

_____

Date:_____ Signature:_____

HEAD OF THE DEPARTMENT

Comments: _____

_____

Date:_____ Signature:_____

Dedication

This work is dedicated to:

To the passionate minds, the tireless creators, and the unstoppable dreamers,

This game is a testament to your unwavering dedication. From the early stages of concept and design to the late nights of coding and testing, you've poured your hearts and souls into this project. Your commitment to excellence, creativity, and collaboration has transformed mere ideas into a captivating digital masterpiece.

Through challenges and triumphs alike, you've stood together, a team united by a shared vision. Each line of code, pixel placed, and idea shared reflects the collective dedication that defines this journey. Your resilience in the face of setbacks and your ability to push boundaries have set new standards in game development.

May this game stand as a testament to the countless hours, the shared laughter, the heated debates, and the genuine camaraderie that fueled its creation. As players embark on the adventure you've crafted, may they feel the heartbeat of your dedication pulsing through every pixel and line of code.

This game is not just a product; it's a manifestation of your dedication, creativity, and teamwork. Here's to the incredible journey you've undertaken, and the indomitable spirit that makes this team truly extraordinary.

Acknowledgements

We extend our heartfelt gratitude to Hafiza Maria, whose guidance and unwavering support have been instrumental in the creation of this game. Your leadership and expertise have not only steered us through challenges but have also inspired a collaborative and creative spirit within the team.

Throughout the development process, your insightful feedback, constructive criticism, and encouragement have shaped the trajectory of our project. Your dedication to fostering an environment of learning and growth has empowered each team member to excel in their respective roles.

Your belief in our capabilities and commitment to excellence motivated us to reach new heights in game development. We are immensely grateful for the time, expertise, and mentorship you generously shared with us.

This game stands as a testament to the effective collaboration between a dedicated team and a supportive supervisor. Your influence has left an indelible mark on our professional and personal development.

Thank you for being more than a supervisor – a mentor, a guide, and a driving force behind our success.

Executive Summary

Realm Wars is a multiplayer shooter game set in a Modern day setting where players engage in fast-paced combat across Different maps, utilizing unique classes with set play styles. With simi realistic visuals, team-based gameplay, and competitive modes like TDM, Domination, Free for all and Capture the flag. Realm Wars gives an immersive gameplay experience for both casual and competitive players, by tapping into this growing market which demands engaging multiplayer experiences.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Chapter 1: Introduction

Realm Wars, an immersive multiplayer shooter where players are transported into a vibrant world with different settings and urban environments. From Rich forests to a crash sight in the desert, where every corner leads to a fast and engaging gun fight. Pick from 4 different Load-outs (Assault, Recon, Support, & Engineer), each with a new and distinct playstyle. Where Assault offers more fast and upfront gameplay , Recon offers a long range and slow gameplay, Support is for Mid to long range gun fights, and Engineer dominates in close range gunfights since its a shotgun class type.

1.1. Background

Realm Wars originated from a collective aspiration of multiplayer shooters like Halo and Call of duty. Conceived by a dedicated team of designers and Developers, the project aims to create an experience that would transport players into a world with rich Gameplay, stunning visuals, and intense combat.

Drawing inspiration from various hero shooters like Team Fortress 2 and Arcade shooters, the development journey of Realm Wars was marked with many brainstorming sessions, design processes, and attention to detail. Each aspect of this game, forms a diverse Load-outs, Art style, gunplay, and Map design.

1.2. Motivations and Challenges

The decision to develop Realm Wars as a final year project stemmed from a combination of personal passion, academic interests, and industry relevance. As avid gamers and enthusiasts of both fantasy literature and multiplayer shooters, our team was driven by a shared desire to create a game that would combine these elements in a unique and engaging way.

1.3. Goals and Objectives

Goals:

To create a multiplayer shooter experience set in the modern day that captivates players with dynamic gameplay, cool maps and fun game modes.

Objectives:

1. **Engage Players**: Design gameplay mechanics, environments, and narratives that immerse players in the world of Realm Wars, encouraging them to explore, strategize, and collaborate with others.

2. **Technical Proficiency:** Utilize industry-standard game development tools and programming languages to implement efficient and stable game mechanics, networking systems, and graphical assets.

3. **Player Retention:** Implement progression systems, rewards, and social features that incentivize continued engagement and foster a strong and active player community.

4. **Feedback Integration:** Gather feedback from playtesting sessions and community forums to identify areas for improvement and implement iterative updates that enhance the overall gameplay experience.

5. **Accessibility:** Ensure that Realm Wars is accessible to a wide audience of players by optimizing performance across different hardware configurations and providing intuitive controls and user interfaces.

6. **Launch and Marketing:** Plan and execute a comprehensive marketing strategy to generate awareness and anticipation for the game leading up to its launch, leveraging social media, and gaming conventions

7. **Post-Launch Support:** Provide ongoing support and updates post-launch to address any technical issues, balance concerns, or additional content requests from the player community, ensuring the longevity and success of Realm Wars in the competitive gaming market.

1.4. Literature Review/Existing Solutions

1. **Engage Players:** By developing an immersive game world and lore that intertwines with the gameplay, making a captivating world for players to explore. Incorporate dynamic events, and challenges that encourage player interaction and a sense of community within the game.

2.**Balance and Variety:** Conduct extensive playtesting and balance iterations to ensure that each class, weapons, and equipments offers unique strengths and weaknesses.

3.**Technical Proficiency:** Utilize established game development engines such as Unity Engine, along with industry-standard programming languages such as C#, to build a stable and optimized game experience. Prioritizing performance, optimization and network stability to ensure smooth gameplay for players across various hardware configs and internet connections.

4.**Accessibility:** Design easy to understand user interfaces and control schemes that targets a diverse range of players, including those with varying levels of gaming experience and physical abilities.

1.5. Project Plan

1.**Pre-Production Phase:**

- Conduct initial concept development and market research
- Define project scope, goals, and objectives
- Formulate development team and allocate roles and responsibilities

2.**Conceptualization:**

- Develop game design document outlining core gameplay mechanics, features, and game art.
- Create concept art, character designs, and environment concepts
- Prototype gameplay mechanics and conduct playtesting to gather feedback.

3.**Testing and QA:**

- Conduct comprehensive testing to identify and resolve bugs, glitches, and performance issues.
- Implement anti-cheat measures and security protocols to ensure fair gameplay.
- Optimize game performance across different hardware configurations and internet connections

4.**Launch Preparation:**

- Finalize game build and prepare for official launch across targeted platforms.
- Coordinate with distribution platforms for launch promotions and visibility.

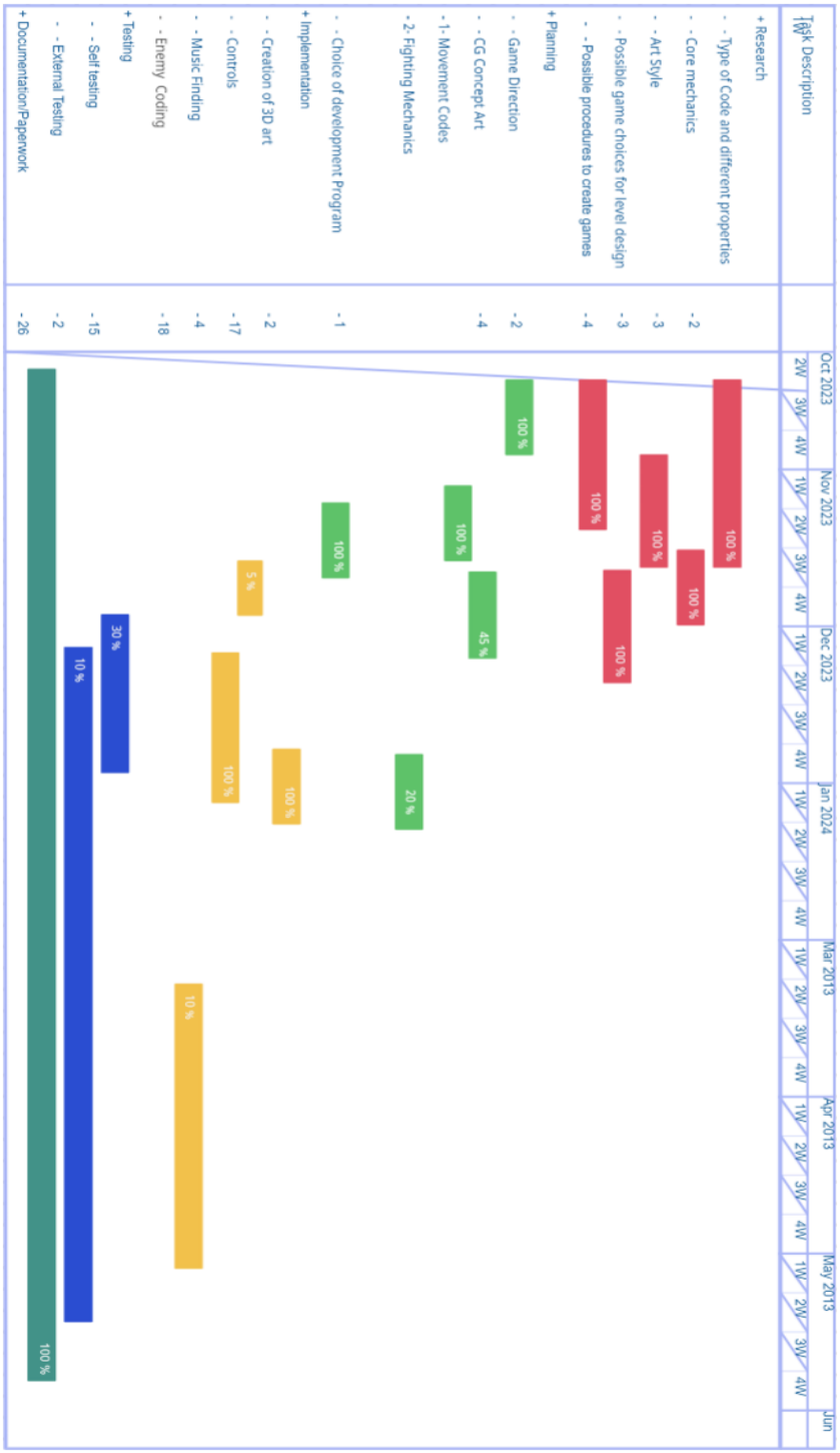
5.**Launch and Post-Launch Support:**

- Officially launch the game across targeted platforms with marketing campaigns and community events.
- Monitor game performance, player feedback, and analytics post-launch.
- Provide ongoing support, updates, and content expansions to maintain player engagement and satisfaction.

1.5.1. Work Breakdown Structure

WBS #	WBS Deliverable	Activity #	Activity to Complete the Deliverable	Duration (# of Days)	Responsible Team Member(s) & Role(s)
1	Art work / key visuals	1	Hand painted	10	Anas Shaikh
1	In-Game assets	2	Hand painted	2 weeks	Anas Shaikh
1	2D Character and environment Sketches	3	Hand painted and sculpted	2 weeks	Anas Shaikh
1	Play test Environment build	4	Environment testing	2 weeks	Anas Shaikh
1	Movement scripts	5	Using C#	2 weeks	Abrar & Aun
1	Attack scripts	6	Using C#	2 weeks	Abrar & Aun
1	UI/UX	7	Hand painted	2 weeks	Anas Shaikh
1	Alpha build	8	Using the Unity engine to compile the game	2 weeks	Anas Shaikh,Abrar & Aun
1	Beta build	9	Using the Unity engine to compile the game	2 weeks	Anas Shaikh,Abrar & Aun
1	Final build	10	Using the Unity engine to compile the game	2 weeks	Anas Shaikh,Abrar & Aun

1.5.2. Gantt Chart



1.6. Report Outline

Creating a comprehensive report outline for the game "Realm Wars" involves organizing key information in a structured format. Here's a general outline that you can adapt and expand upon based on the specific requirements and content you want to include in your report:

1. Executive Summary:

- Brief overview of the game project.
- Key highlights of development and achievements.
- Current status of the game.

2. Introduction:

- Background of "Realm Wars".
- Objectives and goals of the game.
- Target audience and market positioning.

3. Concept and Design:

- Overview of the game concept and storyline.
- Detailed description of gameplay mechanics.
- Design choices and rationale.
- Concept art and visual style.

4. Development Process:

- Project initiation and planning.
- Phases of development (pre-production, production, post-production).
- Challenges encountered and solutions implemented.
- Collaboration and communication within the development team.

5. Technical Features:

- Overview of the game engine and technology used.

- Discussion on key technical aspects e.g.,(momentum-based movement system, dynamic level design).
- Challenges and solutions related to technical implementation.

6. Art and Visuals:

- Description of the art direction and visual style.
- Character design, animations, and environmental assets.
- Visual innovations and unique aspects of the game's aesthetics.

7. Gameplay Mechanics:

- Detailed breakdown of core gameplay elements.
- Exploration of linear map design and replayability features.
- Integration of player choices and impact on the narrative.

8. Marketing and Community Engagement:

- Marketing strategy and promotional activities.
- Community building efforts and engagement initiatives.
- Pre-launch and post-launch marketing plans.

9. Testing and Feedback:

- Beta testing process and outcomes.
- Player feedback and adjustments made.
- Quality assurance and testing methodologies.

10. Launch and Post-Launch:

- Game launch strategy and execution.
- Post-launch support and updates.
- Community response and player metrics.
- Plans for ongoing support and future expansions.

Chapter 2

Software Requirement Specifications

Chapter 2: Software Requirement Specifications

2.1. Introduction

2.1.1. Purpose

The purpose of “Realm Wars” is to provide players with an immersive and exciting multiplayer shooter experience set in a near future war torn world. By blending rich lore, dynamic gameplay mechanics, and realistic visuals, the game aims to take players into a world where they can engage in fast paste gun fights and team coordination.

2.1.2. Document Conventions

The following are the “Document Conventions” for Realm wars:

- Clear file naming, version control, and consistent formatting ensure organization and readability.
- Headers and footers with relevant information aid in document navigation and tracking.
- References and citations follow a standardized format for credibility and attribution.

- Structured documents with clear sections and headings facilitate understanding and comprehension.
- Review and approval processes involve stakeholders to ensure accuracy and quality.

2.1.3. Intended Audience and Reading Suggestions

Realm Wars is tailored for gamers who crave fast-paced gunplay, and strategic multiplayer . Whether you're a casual or a competitive gamer hungry for intense gun fights, Realm Wars offers something for everybody. Dive into the near-future world of “Realm Wars”.

2.1.4. Product Scope

Gameplay Features:

- Multiplayer Modes
- Diverse Classes and Abilities sets
- Progression System

Visuals and Audio:

- Realistic Graphics
- Gun sounds and effects

Community and Social Features:

- Clans
- Leaderboards and Rankings
- Customization Options

Post-Launch Support and Updates:

- Ongoing Development
- Community Engagement

Monetization Strategy:

- Free-to-Play Model
- Season Passes and DLCs

2.1.5. References

Game references:

1. https://store.steampowered.com/app/202970/Call_of_Duty_Black_Ops_II/

2. https://store.steampowered.com/app/976730/Halo_The_Master_Chief_Collection/

2.2. Overall Description

2.2.1. Product Perspective

Realm Wars is a new, self-contained product developed as its own independent multiplayer shooter set in a near-future setting. It is not a follow-on member of any existing product. The game is designed to stand by itself as a complete game, offering unique game mechanics, , and engaging multiplayer. While Realm Wars is a standalone product, it can potentially serve as a component for E-sports, like hosting competitive tournaments or community events in public places like malls and stadiums.

Interfaces between Realm Wars and external systems may include:

• Network interfaces for online multiplayer functionality and connecting players to game servers.

• Payment interfaces for in-game purchases**User Classes and Characteristics**

The Following are the user classes and characteristics:

1. Casual Players (Daily everyday players)
2. Competitive Player (Professional player playing for trophies and cash prises)
3. Social gamers (Players playing to meet new people)
4. Content Creator (Players making content on the game)
5. New players (Players new to the game)

2.2.2. Operating Environment

The game will run via L.A.N or online servers, and players will on their respective PCs (Personal Computers).

2.2.3. Design and implementation constrains

Hardware Limitations

- Compatibility with various hardware configurations, including minimum requirements for CPU, GPU, and memory, may limit graphical fidelity and performance optimizations.

- Timing requirements for real-time multiplayer interactions and synchronization.

Specific Technologies, Tools, and Databases:

- Use of specific game engines (unity and unreal), programming languages (C++ and C#), licensing agreements, and “compatibility/optimization” requirements.

Design Conventions and Programming Standards:

- Adherence to design conventions and programming standards established by the development team or the customer's may influence the code readability, maintainability, and scalability.

Maintenance Responsibilities:

- If the customer's organization will be responsible for maintaining the delivered software, developers may need to provide documentation, training, and ongoing support to ensure smooth transition and continued operation of the game post-launch.

2.3. External Interface Requirements

2.3.1. User Interface

Game Client Interface:

- Sample Screen Images: Include login screen, main menu, in-game HUD, and settings menu

- Screen Layout Constraints: Ensure intuitive placement of essential elements such as health Counter, minimap, and objective indicators.
- Standard Buttons and Functions: Include options for gameplay settings, audio controls, and accessibility features.

- Keyboard Shortcuts: Provide customizable key bindings for common actions such as movement, shooting, and abilities

Social Features Interface:

- **Sample Screen Images:** Include friend list, and clan management.
- **GUI Standards:** Utilize Icons, and Navigation menus.
- **Standard Buttons:** Options for adding friends, and creating/joining clans.

Matchmaking and Lobby Interface:

- **Sample Screen Images:** Include matchmaking queue, and lobby chat.
- **GUI Standards:** Display player matchmaking status, and match settings.
- **Standard Buttons and Functions:** Include options for queueing.

2.3.2. Hardware Interface

Supported Device Types:

- **Desktop Computers:** Support for Windows, and Linux operating systems.
- **Mobile Devices:** Availability on iOS and Android smartphones and tablets.

Nature of Data and Control Interactions:

- **Input Devices:** Interaction with hardware peripherals such as keyboards, mice, controllers, and touchscreens for player control.
- **Output Devices:** Rendering of graphics, audio, and haptic feedback on displays, speaker feedback to the player.
- **Network Communication:** Data transmission between the game client and remote servers for multiplayer gameplay, matchmaking, and social features

Device-Specific Optimization:

- **Hardware Acceleration:** Utilizing graphics processing units (GPUs) for rendering high-quality visuals and physics simulations, optimized for specific GPU architectures and APIs (e.g., DirectX).

- **Platform-Specific APIs:** Integration with platform-specific APIs and SDKs for features such as achievements, leaderboards, and in-app purchases on gaming consoles and mobile devices.

Peripheral Support:

- Controller Input: Support for various controller types, including gamepads, joysticks, and motion controllers, with customizable button mappings and input configurations.
- Touchscreen Interaction: User interface elements optimized for touch input on mobile devices, with intuitive gestures and controls for navigation and gameplay.

Hardware Requirements:

- Minimum and Recommended Specifications: Defined hardware requirements for CPU, GPU, RAM, and storage space to ensure optimal performance and stability across supported devices.
- Optimization for Low-End Devices: Scalable graphics settings and performance optimizations to accommodate lower-end hardware configurations without sacrificing gameplay quality.

2.3.3. Software Interfaces

Operating Systems:

- Windows 10 and lter

Game Enginen:

- Unity

Networking Middleware:

- Photon Unity Networking (PUN) (version 2.26 or later)e.

Database Integration:

- SQLite (version 3.35 or later)
  Purpose: Provides local storage for player profiles, settings, and progress data on client devices.

Integrated Commercial Components:

- Steamworks SDK (version 1.50 or later)

Messaging Protocols:

- TCP/IP and UDP

  Purpose: Facilitates communication between the game client and remote servers for multiplayer gameplay, matchmaking, and social features.
- HTTP/HTTPS

  Purpose: Supports web-based interactions such as authentication, updates, and in-game purchases.

Data Items and Messages:

- Incoming Data:
  - Player inputs (e.g., keyboard, mouse, controller)
  - Network messages for player movement, actions, and interactions
- Outgoing Data:
  - Game state updates for synchronization with remote servers
  - Achievement and progress updates for integration with platform-specific features

Shared Data:

- Player profiles and progress data shared between the game client and remote servers
- Matchmaking preferences and lobby configurations shared between players during matchmaking and lobby interactions

2.4. System Features

Gameplay Features:

- Multiplayer Modes: Provide options for team-based battles, free-for-all skirmishes, and objective-based gameplay.

- Classes and Abilities: Implement a diverse roster of classes with unique abilities, weapons, and playstyles.
- Progression System: Offer a progression system that allows players to unlock new weapons, abilities, and cosmetic items as they level up.
- Dynamic Environments: Feature diverse maps and landscapes with interactive elements and strategic opportunities.

Visuals and Audio:

- Stunning Graphics: Deliver visually immersive environments, character designs, and special effects.
- Atmospheric Soundtrack: Complement gameplay with dynamic music and sound effects that enhance the player experience.

- **Social and Community Features:**
- Clans and Guilds: Enable players to form alliances, compete in tournaments, and earn rewards as a team.
- Leaderboards and Rankings: Track player progress and achievements, promoting competition and recognition within the community.
- Customization Options: Provide extensive character customization features for personalization and self-expression**.**

Matchmaking and Lobby Services:

- Matchmaking System: Implement a robust matchmaking system that ensures fair and balanced matches based on player skill levels and preferences.
- Lobby Interface: Design intuitive lobby interfaces for players to create, join, and customize game sessions before starting matches.

Progression and Customization Services:

Player Progression: Track player progress and rewards through a comprehensive progression system that offers meaningful incentives for gameplay.

- Loadout Customization: Allow players to customize their loadouts with weapons, abilities, and cosmetic items to suit their playstyle.

Networking and Communication Services:

- Real-Time Multiplayer: Support real-time multiplayer gameplay with seamless synchronization of game state and actions between players.
- Voice Chat: Integrate voice chat functionality for communication between players during matches and social interactions.

Platform Integration Services:

- Cross-Platform Compatibility: Ensure compatibility across multiple platforms including PC, consoles, and mobile devices for broad accessibility.
- Platform-Specific Features: Integrate platform-specific features such as achievements, leaderboards, and in-app purchases on gaming consoles and mobile devices.

Post-Launch Support and Updates:

- Ongoing Development: Commit to regular updates, patches, and expansions to introduce new content and address player feedback.
- Community Engagement: Establish channels for player feedback, support, and interaction to foster a vibrant and engaged player community.

2.4.1. Feature: Multiplayer Modes

Description and Priority: This feature allows players to engage in various multiplayer modes, including team-based battles, free-for-all skirmishes, and objective-based gameplay. It is of High priority as it forms the core of the multiplayer experience.

Stimulus/Response Sequences:

- User selects desired multiplayer mode from the main menu.
- System displays available modes and allows the user to choose one.
- Upon selection, the system initiates matchmaking or creates/joins a lobby based on the chosen mode.
- Players are matched or grouped together, and the game session begins according to the selected mode.

Functional Requirements:

- The game client must provide options for selecting different multiplayer modes, including team-based, free-for-all, and objective-based modes.
- The system must matchmake players into appropriate lobbies or game sessions based on their selected mode.
- Players must be able to create and customize private lobbies for custom game modes and matches.
- The system must ensure fair and balanced matchmaking to provide an enjoyable multiplayer experience for all players.
- The game must support seamless transition between different multiplayer modes without interruption to the player experience.
- In the event of matchmaking errors or connection issues, the system must provide informative error messages and allow players to retry or exit gracefully.

2.4.2. Player Progression

Description and Priority: This feature enables player progression, tracking their achievements, experience points, and rewards earned throughout the game. It is of High priority as it contributes to player engagement and long-term retention.

Stimulus/Response Sequences:

- User completes in-game objectives, wins matches, or earns achievements.
- System updates player profile and progression data accordingly.
- Player receives visual and auditory feedback indicating progression, such as level-ups, unlocked items, or earned rewards.

Functional Requirements:

- The system must track player experience points (XP) earned through gameplay actions such as completing objectives, winning matches, and performing well in-game.
- Players must progress through levels based on accumulated experience points, with each level providing rewards and unlocking new gameplay features.
- The game client must display player progression data, including current level, experience points, and progress towards the next level.
- The system must reward players with in-game currency, items, and cosmetic unlocks as they progress through levels and achieve milestones.
- Players must have the ability to view their progression history, including achievements unlocked, rewards earned, and progression milestones reached.
- In the event of progression-related errors or discrepancies, the system must provide clear error messages and allow players to report issues for resolution.

2.5. Nonfunctional Requirements

2.5.1. Performance Requirements

The following are the performance requirements for Realm Wars:

- 60 FPS (Minimum)
- 720P-1080 resolution (Minimum)

2.5.2. Safety Requirements

- The system must provide clear error messages and a reporting mechanism for progression-related issues to prevent loss or damage to player data.
- Safeguards must be implemented to detect and resolve progression-related errors proactively, minimizing potential harm to players.

2.5.3. Security Requirements

- **Data protection:** Implement secure data transmission protocols for online mutiplayer

- **Authentication:** Utilize strong authentication mechanisms to secure user accounts and prevent unauthorized access.

2.5.4. Usability Requirements

- **User Interface Clarity:** Ensure that the user interface is intuitive and user-friendly, with clear indications of health, ammunition, and other relevant information.
- **Accessibility:** Support accessibility features such as color-blind modes and customizable control mappings to cater to a diverse player base.

2.5.5. Reliability Requirements

- **Server Uptime:** Aim for 99.9% server uptime to ensure continuous availability for players.

- **Stability:**Minimize crashes and system failures to provide a stable gaming experience. The game client should not crash more than once per 20 hours of gameplay.

2.5.6. Maintainability/Supportability Requirements

- **Modularity:** Design the game codebase with modularity to facilitate easier updates and the addition of new content.

- **Documentation:** Maintain comprehensive documentation for developers, including code documentation, design documents, and update/release notes.

Chapter 3

Use Case Analysis

Chapter 3: Use Case Analysis

1. Character Selection:

Primary Actor: Player

Description: Enables players to choose or create their in-game character.

Use Case Steps:

1. Player accesses the character selection screen.
2. The system presents existing characters or allows the creation of a new one.
3. Player customizes the character's appearance and selects a class.
4. The system saves the selected character for gameplay.

2. In-Game Movement and Combat:

Primary Actor: Player

Description: Covers player actions during a match, including movement and combat.

Use Case Steps:

1. Player navigates the environment using momentum-based movement.
2. Combat actions include aiming, shooting, using special abilities, and reloading.
3. The system calculates damage, hit detection, and triggers appropriate animations.
4. Players strategize and adapt their movements based on the linear map design.

3. Environmental Interaction:

Primary Actor: Player

Description: Involves interactions with interactive elements within the game environment.

Use Case Steps:

1. Players approach interactive elements (e.g., switches, destructible objects).
2. The system responds with dynamic changes in the environment.
3. Environmental interactions impact gameplay and may provide strategic advantages.

4. End Rewards:

Primary Actor: System

Description: Concludes a multiplayer match, determining winners and providing rewards.

Use Case Steps:

1. The system evaluates level clear results based on predefined victory conditions.
2. Rewards, such as experience points and in-game currency, are distributed.
3. Players receive updates on their performance and progression.

5. Game Update and Patching:

Primary Actor: System

Description: Involves the process of updating the game with new content, fixes, or improvements.

Use Case Steps:

1. System notifies players of available updates.
2. Players initiate the update process or it occurs automatically.
3. The system downloads and installs the update.
4. Players access the latest version of the game.

This Use Case Analysis provides an overview of the key interactions and functionalities within "Realm Wars," covering player registration, character selection, multiplayer matchmaking, in-game actions, environmental interaction, end-of-match processes, game updates, and spectator features. Each use case contributes to the overall player experience in the momentum-based movement shooter with a linear map design.

Chapter 4

System Design

Chapter 4: System Design

1. **Game Overview:**

Genre: FPS shooter.

Gameplay: Semi Linear/ open map design with a level-based progression system.

Mode: Multiplayer.

2. **Architecture:**

Client-Server Model: As it is a single-player and multiplayer game with a simple client architecture with a Photon server interaction that can be employed.

Components: Client-side game engine, rendering system, input system, and level manager.

3. **Game Engine:**

Selection: Unity.

Features: Physics engine for momentum-based movement, rendering system for graphics, and sound engine for audio.

4. **Game Loop:**

Input Processing: Capture user inputs for movement, shooting, and any other relevant actions.

Physics Update: Apply physics to handle player movement, collisions, and interactions.

Game Logic Update: Implement game-specific logic such as enemy AI, level progression, and scoring.

Rendering: Display the updated game state to the player.

5. **Player Movement:**

Momentum-Based: Implement physics for realistic momentum-based movement to enhance gameplay dynamics.

Controls: Support keyboard/mouse or controller inputs for a seamless gaming experience.

6. **Weapons and Power-ups:**

Arsenal: Include a variety of weapons with different attributes.

Pick-ups: Enhance player abilities temporarily, adding strategic depth.

7. **Level Design:**

Linear Maps: Design levels in a linear fashion with progressive difficulty.

Obstacles and Challenges: Introduce obstacles, enemies, and challenges to maintain engagement.

8. **Enemy AI:**

Behavior: Implement diverse enemy behaviors based on the Map layout and weapon in hand.

Difficulty Scaling: Increase enemy difficulty according to the player's desire.

9. **Level Progression:**

Objective-Based: Set clear objectives for each level.

Scoring System: Reward players based on performance, encouraging replayability.

10. **UI/UX Design:**

HUD Elements: Display essential information like health, ammo, and objectives.

Menus: Intuitive menus for Game-mode selection, settings, and player statistics.

4.1. Domain Model

A Domain Model provides a visual representation of the conceptual classes and relationships within the game:

1. **Player Class:**

   Properties: Position, Health, Ammo, Score

   Methods: Move(), Shoot(), TakeDamage()

2. **Weapon Class:**

   Properties: Damage, Ammo, FireRate

   Methods: Fire(), Reload()

3. **Enemy Class:**

   Properties: Position, Health, Behavior

   Methods: Move(), Attack(), TakeDamage()

4. **Level Class:**

   Properties: MapLayout, Objectives, Pick-ups

   Methods: Update(), Complete()

5. **PickUp Class:**

   Properties: Type, Duration, Effect

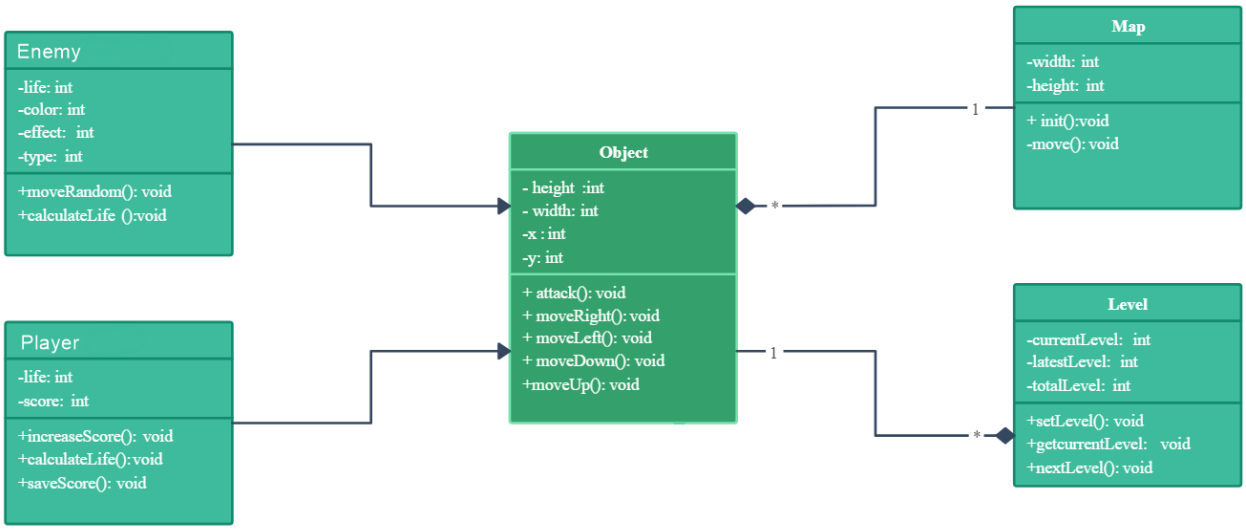
   Methods: ApplyEffect(), Expire()

6. **GameController Class:**

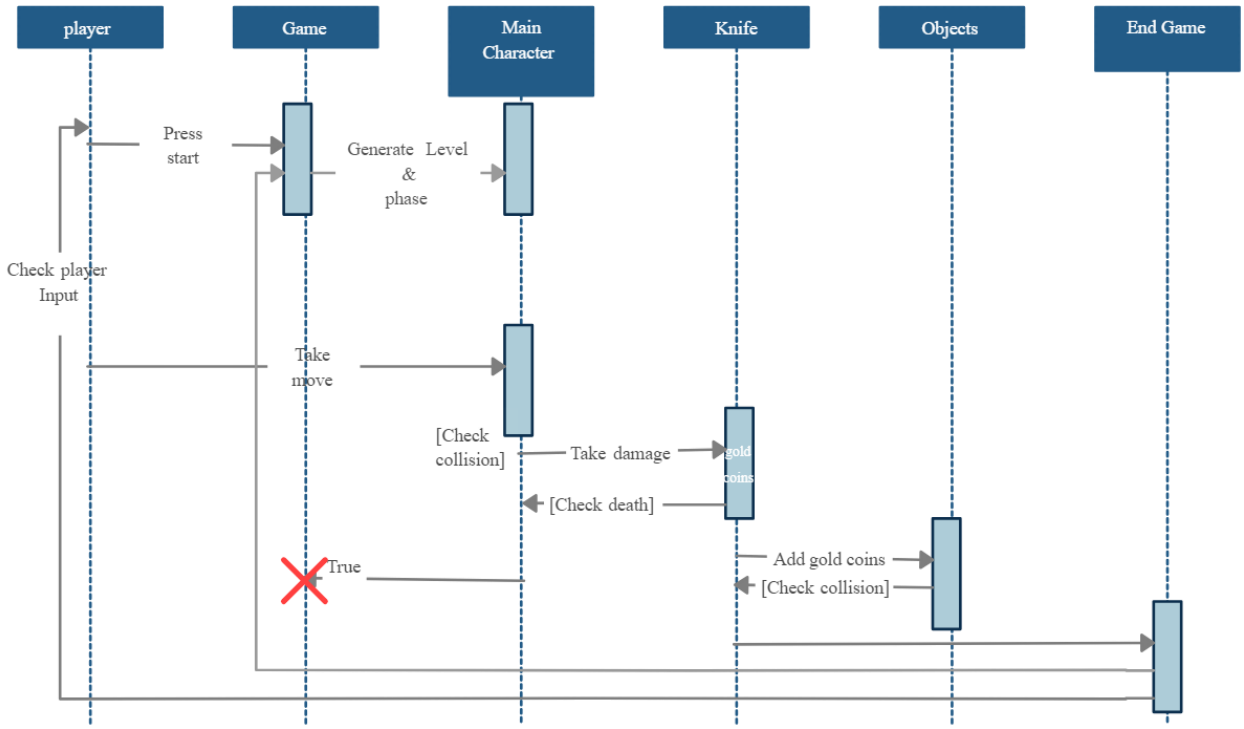
   Properties: CurrentLevel, PlayerInstance, EnemyList

   Methods: StartGame(), EndGame(), LoadLevel()

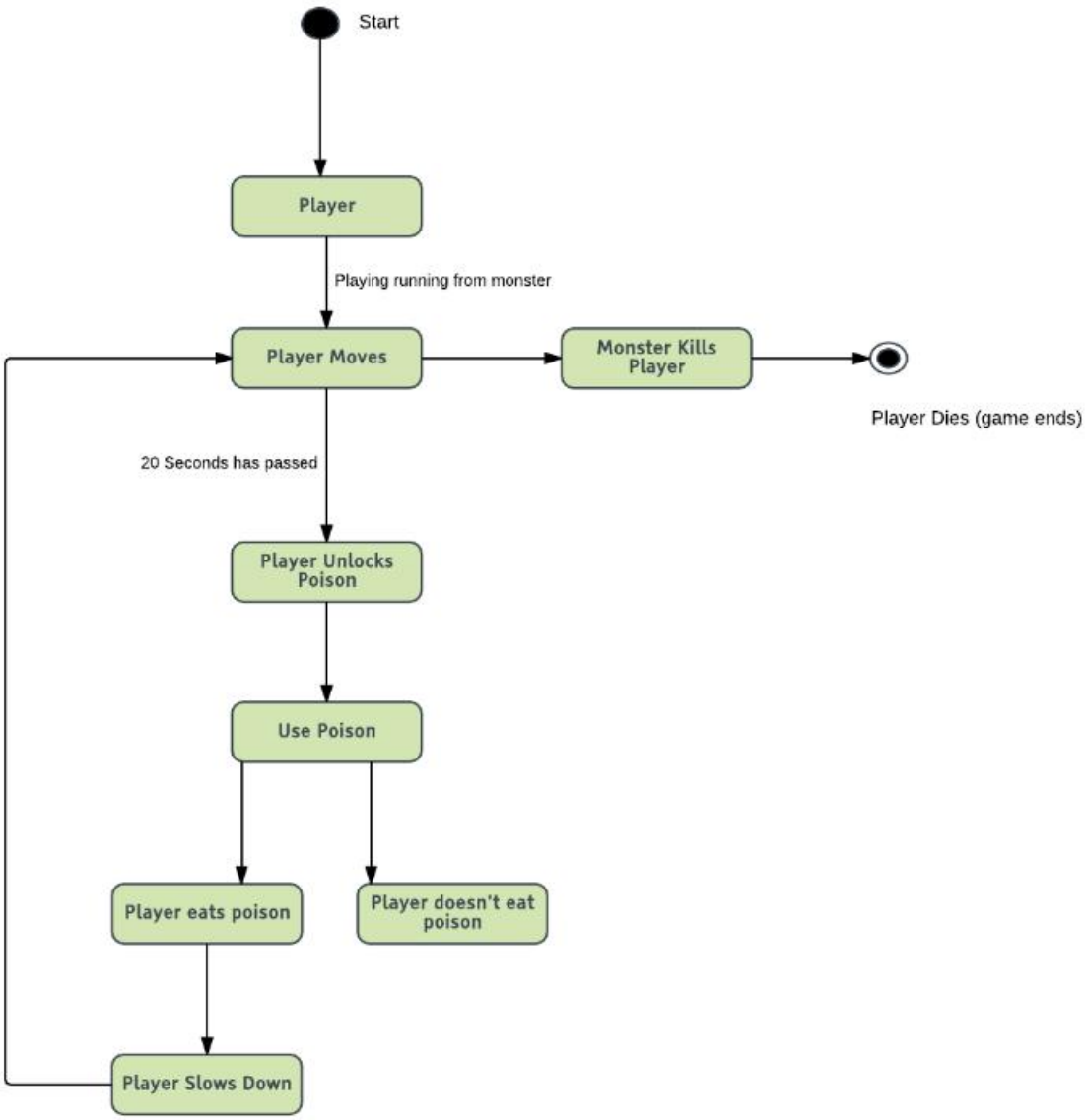
4.2. Class Diagram



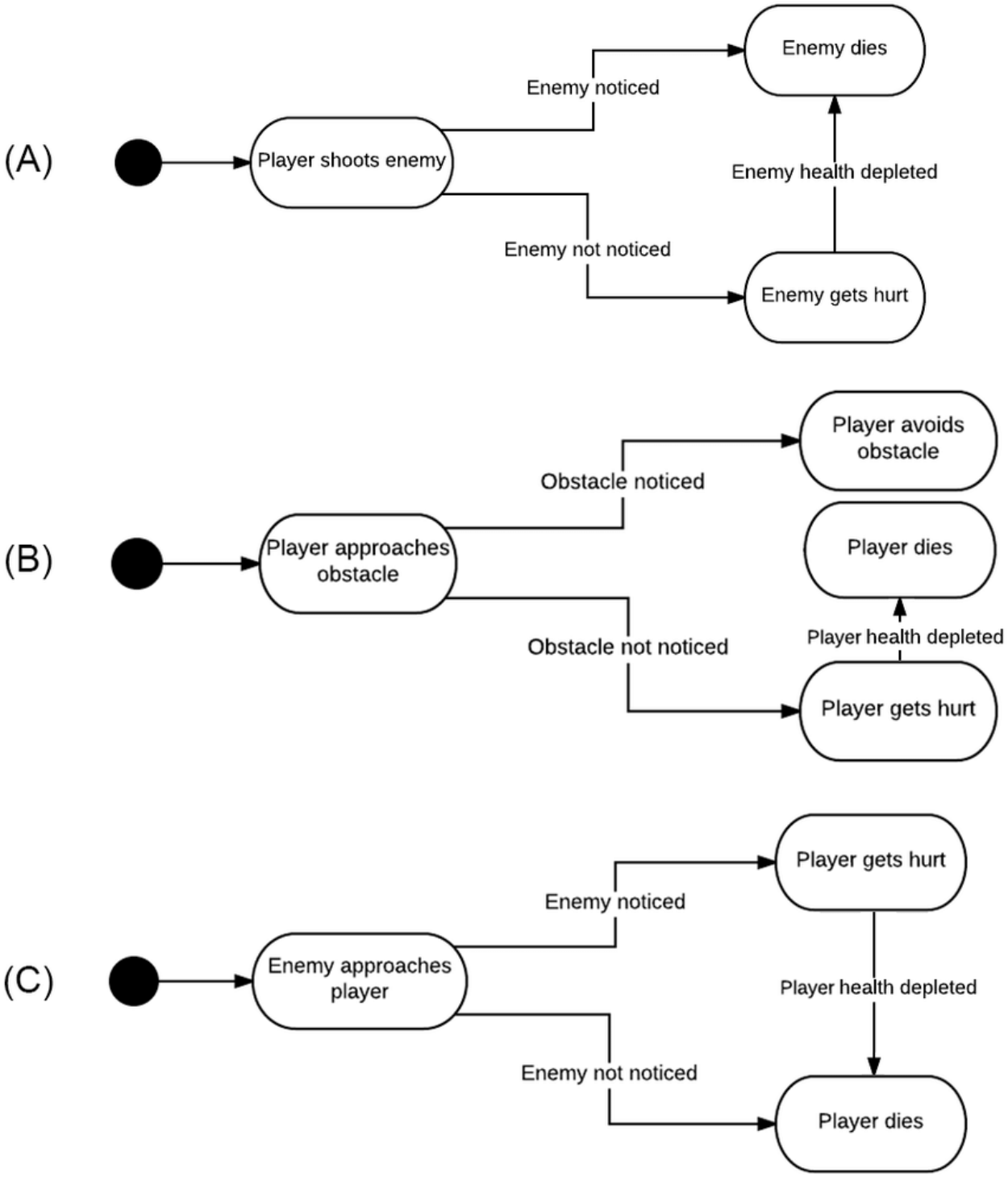
4.3. Sequence / Collaboration Diagram



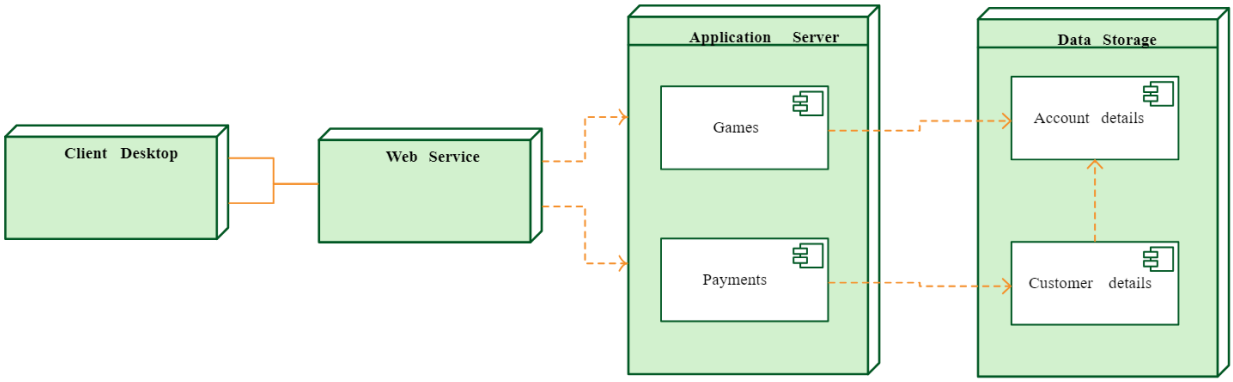
4.4. Activity Diagram



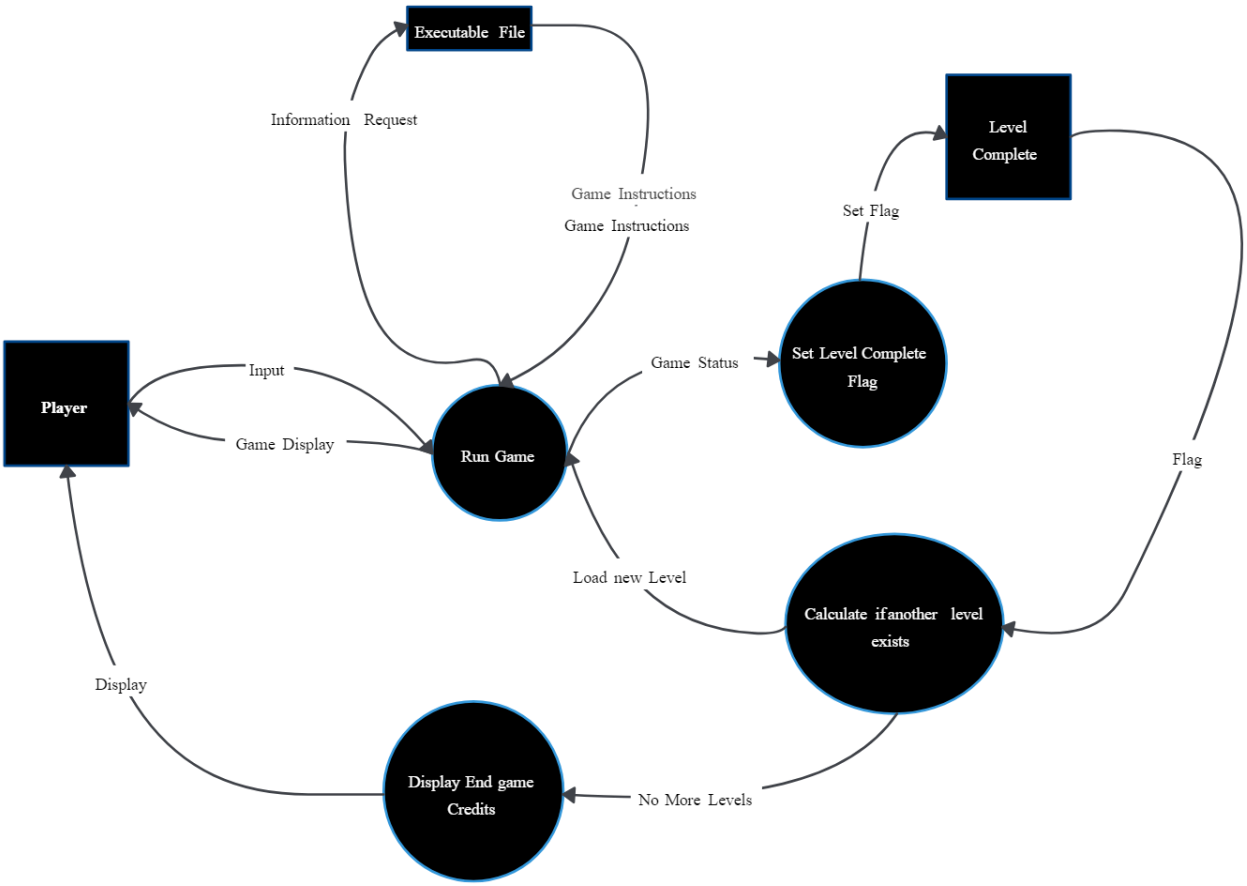
4.5. State Transition Diagram



4.6. Deployment Diagram



4.7. Data Flow diagram



Chapter 5
Implementation

Chapter 5: Implementation

Creating a full game implementation involves a substantial amount of code, but I'll provide you with a basic outline and pseudocode for key components. This implementation will be based on Unity and Blender 3D for game development, and I'll touch on best practices, coding standards, and version control as well.

5.1. Important Flow Control/Pseudo codes

1. **PlayerController:**
   - Handles player input and movement.
   - Applies momentum-based physics.

```csharp
public class PlayerController : MonoBehaviour
{
    public float speed = 10.0f;
    public float jumpForce = 5.0f;

    void Update()
    {
        // Handle player movement
        float horizontal = Input.GetAxis("Horizontal");
        float vertical = Input.GetAxis("Vertical");
        Vector3 movement = new Vector3(horizontal, 0.0f, vertical);
        GetComponent<Rigidbody>().AddForce(movement * speed);
```

```
        // Jumping
        if (Input.GetButtonDown("Jump"))
        {
            GetComponent<Rigidbody>().AddForce(Vector3.up * jumpForce, ForceMode.Impulse);
        }
    }
}
```

2. **LevelManager:**

- Manages level progression and spawning of enemies.

```csharp
public class LevelManager : MonoBehaviour
{
    public Transform playerSpawnPoint;
    public GameObject enemyPrefab;
    public int numberOfEnemies = 5;

    void Start()
    {
        SpawnPlayer();
        SpawnEnemies();
    }
```

```
    void SpawnPlayer()
    {
        Instantiate(playerPrefab, playerSpawnPoint.position, playerSpawnPoint.rotation);
    }


    void SpawnEnemies()
    {
        for (int i = 0; i < numberOfEnemies; i++)
        {
            Vector3 enemySpawnPosition = // Calculate spawn position;
            Instantiate(enemyPrefab, enemySpawnPosition, Quaternion.identity);
        }
    }
}
```

Important Flow Control/Pseudocode:

Game Loop Pseudocode:

```csharp
while (gameRunning)
{
    ProcessInput();
    UpdatePhysics();
    UpdateGameLogic();
    Render();
}
```

Update Physics Pseudocode:

```csharp
void UpdatePhysics()
{
   // Apply physics calculations
}
```

Update Game Logic Pseudocode:

```csharp
void UpdateGameLogic()
{
   // Handle game-specific logic (enemy AI, level progression, etc.)
}
```

5.2. Components, Libraries, Web Services and stubs

Libraries:

1. **Unity Engine**: Game development engine.
2. **Blender 3D:** 3D modeling and animation.
3. **Photon:** For online services and server connections.

5.3. Deployment Environment

Deployment Environment:

Platform: Unity supports various platforms (Windows, macOS, Linux, Android, iOS, etc.).

5.4. Tools and Techniques

Tools and Techniques:

1. **Unity Editor:** For game development and testing.
2. **Blender 3D:** For creating 3D models and animations.
3. **Git:** Version control for source code.

5.5. Best Practices / Coding Standards

Best Practices / Coding Standards:

1. Follow Unity's coding conventions.
2. Use meaningful variable and function names.
3. Organize code into classes and functions for modularity.
4. Comment code for clarity.
5. Optimize performance where necessary.

5.6. Version Control

Version Control:

Use Git for version control. Initialize a Git repository for your Unity project and commit changes regularly. Create branches for features or experiments, and use pull requests for code review. Remember, this is a simplified overview. In a real-world scenario, you'd need to consider many additional factors, such as asset management, sound, UI, and more.

Anas Shaikh

ORIGINALITY REPORT

9%	7%	1%	8%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Higher Education Commission Pakistan Student Paper	8%
2	www.coursehero.com Internet Source	1%

Exclude quotes	On	Exclude matches	< 3 words
Exclude bibliography	On		