Airplane Citizen Rescue Game

Authors

Syed Arsalan Shah
Wajahat Ullah
Irfan Rahmani

Supervisor: Dr. Afsheen Khalid

Final Year Project Report submitted in partial fulfillment of the requirements for the Degree of BSSE (Hons.)

INSTITUTE OF MANAGEMENT SCIENCES, PESHAWAR
PAKISTAN

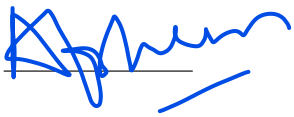
Session: 2020-2024

Certificate of Approval

I, certify that I have read the report titled: **Airplane Citizen Rescue Game**, by **Syed Arsalan Shah, Wajahat Ullah and Irfan Rahmani**, and in my opinion, this work meets the criteria for approving the report submitted in partial fulfillment of the requirements for BSSE (Hons.) at Institute of Management Sciences, Peshawar.

Supervisor: Dr. Afsheen Khalid

Assistant Professor

Signature: 

Coordinator BSSE (Hons.): Mr. Hamood Ur Rehman Durrani

Assistant Professor

Signature: _____

Coordinator FYP: Mr. Omar Bin Samin

Lecturer

Signature: _____

Declaration

We, **Syed Arsalan Shah, Wajahat Ullah and Irfan Rahmani**, hereby declare that the Final Year Project Report titled: **Airplane Citizen Rescue Game** submitted to FYP Coordinator and R&DD by us is our own original work. We are aware of the fact that in case, our work is found to be plagiarized or not genuine, FYP Coordinator and R&DD has the full authority to cancel our Final Year Project and We will be liable to penal action.

Syed Arsalan Shah

BSSE (Hons.)

Session: 2020-2024

Wajahat Ullah

BSSE (Hons.)

Session: 2020-2024

Irfan Rahmani

BSSE (Hons.)

Session: 2020-2024

Dedication

We dedicate this Final Year Project to our parents and teachers, who have always supported and helped us in every aspect of life.

Acknowledgement

All the praise to Allah that induced the man with intelligence, knowledge, and wisdom. Peace and blessing of Allah be upon the Holy Prophet who exhort his followers to seek for knowledge from cradle to grave.

Foremost, We would like to express our sincere gratitude to our supervisor Mr. Omar Bin Samin for his continuous support, patience, motivation, enthusiasm, and immense knowledge. His guidance helped us throughout the project. Last, but not the least, We would like to thank our parents for supporting us morally and spiritually throughout our life.

Abstract

This thesis addresses the development of "Airplane Citizen Rescue," a pioneering mobile game which integrates aerial battle with citizen rescue operations. Addressing a gap in the industry of gaming, the research focuses on developing a thrilling and purpose-driven gaming experience for users. Utilizing Unity 3D and programmed in C sharp, the project leverages the Scrum approach to manage its iterative development process.

The research commences by identifying the necessity for a game that mixes action-packed gameplay with humanitarian objectives. Methodologically, the project applies rigorous requirement engineering methodologies, using existing game evaluations and interviews to build its objectives and technical specifications. Design efforts, including architectural diagrams and system sequence diagrams, constitute the blueprint for implementation, stressing modular architecture and scalability.

Testing plays a crucial role in developing gaming mechanics, with user feedback and stakeholder reviews guiding iterative changes. The documentation process, from Software Requirements Specification (SRS) to thorough test plans, documents the project's history and acts as a roadmap for future development.

Key findings emphasise the successful integration of gaming features and the achievement of project objectives. The conclusion highlights the significance of "Airplane Citizen Rescue" as a pioneering success in mobile gaming, offering players a riveting blend of fun and purpose. Moving forward, the research sets the foundation for continued innovation and advancement in the changing field of mobile gaming.

Contents

List of Figures

Chapter 1

Introduction

With the rise of the digital era where people crave for entertainment, gaming has emerged victor among many entertainment sources. In the last decade, gaming industry has seen massive growth and is now twice as big than the film and music industry combined. The gaming industry attracts tremendous number of players from variety of age groups that spend a great part of their time playing video games. Such a passionate gaming community demands games that that fulfill their expectation in almost every regard be it performance, graphics, gameplay or storyline. With a vast number and variety of gamers, there arises a demands in variation and diversity of gaming genres including action, adventure, mystery, story mode, online multiplayer, single player, first person shooter (FPS), simulation etc, where players are always looking for a game that offers new experiences and a unique feel. Keeping these factors in mind, the demand for game development and game developers has and is increasing worldwide, and a successful game can bare fruitful in many aspects.

With our project "Airplane Citizen Rescue Game", we want to dive into the domain of game development and develop an immersive gaming experience for the gaming community. In this documentation, we present the journey of our project exploring its technical and non-technical aspects.

1.1 Overview

The "Airplane Citizen Rescue Game" is a thrilling and action-packed 3D video game project developed as a part of a final year undergraduate program. The video game provides an immersive gaming experience by placing the player into the role of a heroic pilot, that possesses exceptional talent and abilities as a pilot and is equipped with variety of advanced jets and helicopters capable of performing stunning maneuvers. The game environment is set in a world where the cities are at the threats of variety of rampaging monsters and natural disasters that put the public at risk. The player, being a heroic pilot has to act as the strongest

line of defense and protect the cities from any external threat by defeating the monsters with his fighter jet. The player also must protect the citizens in case of a natural disaster and rescue the troubled citizens from the dangers while carefully maneuvering their helicopter, ensuring the safety of every citizen. The video game is mission based, the missions evolves with the skill of our protagonist, i.e. as he gets experience with every passing mission, the level of difficulty, the intensity of monster attacks, the scale of natural disasters and machinery upgrades. With each successful mission the game advances to a new adventure unlocking more exciting elements of the game. In this high-stakes adventure, player must engage in an action packed aerial combat with the monsters, navigate the skies of the beautifully crafted world, perform daring rescue missions, ensure the safety of the city and its inhabitants, all while being involved in a compelling gameplay and experiences a heroic feeling.

1.2 Project Motivation

- Motivated by a deep passion for gaming and game development, the inception of "Airplane Rescue Game" was incited to craft a unique game of our own.
- Recognizing the lack of games being produced in our regional gaming market, especially aerial combat games, we aim to fill this gap by developing a fresher experience for the gaming community.
- Aerial combat provides an unparalleled and distinct immersive experience that many other vehicle centric games often fail to provide.
- Inspired by our childhood dreams of being a pilot, interests in aviation, military jets, and helping public, we aim to integrate these elements under one experience.
- We aspire to develop a game that fuses the elements of aerial action, rescue mission, heroism, story mode single player games for a fresher and thrilling experience.
- We intend to promote positivity with our game, with our hero setting an example of pragmatic and brave individual inspiring and encouraging the players in areas of aviation, rescuing and be righteous.
- Observation led us to the conclusion that there is a scarcity in development of gaming project in our university. By showcasing our 3D gaming project, we want to motivate fellows and juniors specially to opt for creative gaming projects, including 3D games within our university.

- Thus the "Airplane Citizen Rescue Game" serves as a showcase for our enthusiasm, a reaction to market demands, and a motivator for people to explore career options in the exciting world of game production.

1.3 Project Vision, Scope, and Glossary

1.3.1 Project Vision

We envision our game to provide an exciting, thrilling and an engaging experience to the player that combines aerial combat, action, heroism and sense of fulfilling responsibility. Our vision is to create a game that fulfills the gap we saw in the market. i.e. there were fewer to no aerial games that added the element of rescuing citizens. Therefore, with our game, we envision to fill this gap. We envision to spread positivity with our game by promoting a courageous hero while providing peak entertainment. beyond entertainment, we extend our vision to inspire our players and promote passion for rescue work, aviation and positivity. In future, our vision includes expanding game support to multiple platforms like Windows making it available for more players to enjoy.

1.3.2 Project Scope

The game "Airplane Citizen Rescue Game" is divided into two modes of missions, one being the aerial combat missions and the other missions focused on rescuing citizens from disasters e.g. natural disasters, fire disaster. There are total of 7 missions in the game, with 4 being aerial combat missions and 3 being the rescue missions. The development of the game is done for Android platforms. The game for now does not work for Windows. The game is a mixture of 3D and 2D. All gameplay aspects of the game are to be built in 3D while majority of the user interface (UI) is to be built in 2D. For the development, unity 3d game engine is being utilized based on C sharp. The major game assets including (jets, helicopters, city landscapes) are sourced from unity's official asset store. While low level game assets are to be built inhouse in Blender 3D The game does not include multiplayer support and is based on a single player style. The primary demographic for our game caters mostly to younger audience and teenagers, providing a captivating and immersive experience that suits their tastes. However, the game's inclusive design makes sure that players of all ages may have fun.

1.3.3 Glossary

- Aerial Combat: Situations in games where players' planes fight and battle in the air, usually against monsters.
- Asset Store: A digital marketplace that facilitates the discovery and acquisition of digital assets by developers, such as textures and 3D models.
- C sharp Scripting: The act of writing code to automate tasks, manipulate objects, and augment the functionality of systems and applications is referred to as "C# scripting".
- Heads-up Display (HUD): A transparent display onto which information is displayed in the user's line of sight.
- High level Architecture: The organization and structure of a complex system as a whole, with an emphasis on the interactions between its main components and the guiding principles that govern their behavior.
- Joystick: A 2D HUD in gameplay that helps to guide the plane moveability,
- Load Time: Load time implies the duration required for an entire set of essential game data to be loaded prior to gameplay.
- Missions: Certain tasks or goals in the game that the player needs to finish.
- Modularity: The extent to which the individual modules or components of a system can be disassembled and reassembled.
- Game Asset: Any digital element that assists in the development and administration of a video game is considered an asset.
- In-house Asset: Assets or components that are internally developed by the development team.
- Natural Disasters: Catastrophe that happen in the game, like earthquakes, floods, fires.
- Poly Count Optimization: Management of the quantity of polygons (poly count) within a three-dimensional model in order to strike a balance between operational efficiency and visual precision.
- Texturing: Texture modification and application on three-dimensional surfaces in order to improve the visual appeal of assets.
- Rigging: This is the process of giving characters or items a skeleton or structure so that they can be animated.

- User Interface (UI): The user interface (UI) of a video game comprises the visual components that facilitate player interaction with the game. It comprises the entirety of the visual elements that appear on the screen.
- Unity 3D Game Engine: An extensively adopted and robust game development engine that facilitates the creation of augmented reality, virtual reality, 2D, and 3D games.

1.4 Objectives

- To optimize game performance for Android devices.
- To achieve realistic and immersive aerial mechanics for the gameplay
- To maintain a GitHub repository that includes all necessary information regarding the project.
- To develop an engaging and intuitive UI for better user experience.
- To maintain and complete the necessary documentation for the project.
- To implement a comprehensive quality assurance process to ensure a bug free release of the project.

1.5 Tools

A range of tools will be utilized in the development cycle of our project.

1.5.1 Vs Code

Vs code is a lightweight and versatile code editor with a language support of variety of programming languages. We will use Vs code for code editing and script development

1.5.2 Unity

Unity is a versatile and simple game engine software for 2D and 3D game development. We wil be using unity for our game development, design and testing

1.5.3 Blender

Blender is a popular 3D modelling tool for making models and animations. We will be using it for unique asset creation and manipulating the existing assets.

1.5.4 Adobe Photoshop

Adobe photoshop will be used for creating UI elements of the game.

1.5.5 Adobe Illustrator

Adobe illustrator will be used for creating UI elements like buttons, logos and icons.

1.5.6 Trello

Trello is a project management tool that we will be using to collaborate and tracking progress.

1.5.7 Github

Github will be used a version control system for our project. Code base and related documentation will be included in maintained Github repositories.

1.5.8 Latex

LaTex ensures development of structured, consistent and professional document formatting. We will be using LaTeX for our project documentation.

1.5.9 Google Meet

Google Meet will be utilized for our team and stakeholder meetings and collaboration.

1.5.10 Microsoft Visio

MS Visio will be used for creating diagrams required in documentation like processes, workflows, architecture and design diagrams.

Chapter 2

Background Study

This chapter discusses relevant previous research works/ projects. After detailed discussion, compare all discussed research works and state limitations of them. This section explores the fundamental components that form the framework of our game development endeavor. We hope to provide a comprehensive background that supports the creation and development of our game by examining important technology. Mainly we will be overviewing Unity and Blender and how it contributes in out project. We will be encompassing historical viewpoint of these technologies, and relevant concepts that are pivotal to understand in the project.

This section examines the fundamental components that constitute the structure of our game development project. We hope to provide a comprehensive background that supports the creation and development of our game by examining important technology. Our focus will primarily be on examining the roles of Unity and Blender in our project. We will be examining the essential principles that are crucial to comprehend in the project.

2.0.1 Unity

Unity is a vital part for present-day game development, encapsulating an age of innovation and efficiency in the industry. Unity, a game engine that is both adaptable and resilient, has transcended its status as fundamental software and has become a driving force behind an endless number of innovative titles and interactive experiences. This section investigates the fundamental components of Unity, such as its function, characteristics, and substantial impact on the ever-evolving field of game development. Unity is a comprehensive game development platform that offers a unified set of tools and capabilities that empower developers to put into action their inventive ideas. Unity's adaptability enables developers of every level, from independent creators to major corporations, to create captivating 3D environments and engaging 2D experiences, thereby catering to a wide variety of game genres.

Key Features of Unity

Unity offers an extensive array of features. Below, we will describe the features that we will use during our game development journey.

Cross-Platform Capabilities: The distinguishing characteristic of Unity is its exceptional cross-platform capabilities. By utilizing a solitary codebase, creators have the ability to distribute their games across several platforms, such as personal computers, gaming consoles, mobile devices, and new technologies like virtual reality (VR) and augmented reality (AR).

Scripting in C sharp Unity utilizes C# as its principal programming language, providing a robust and user-friendly syntax. Developers can utilize the full capabilities of C# to create gameplay logic, AI, and other vital components, guaranteeing both efficiency and user-friendliness.

Animation Unity's animation system allows developers to layer multiple animations dynamically, creating complex character behaviors. It also offers animation baking, which improves performance and reduces real-time processing. Custom C# scripts extend the system, providing control over playback and interaction. Unity offers various animation packages and plugins for specific needs. Unity's animation system possesses considerable power, adaptability, and user-friendliness, making it a favored option for game developers across all skill sets and backgrounds.

2D and 3D development Unity demonstrates its versatility by providing strong support for both 2D and 3D game creation. Developers have the ability to effortlessly switch between dimensions, allowing them to create complex 3D environments or nostalgic 2D experiences, and explore a wide range of creative opportunities.

Unity's Official Asset Store The Unity Official Asset Store serves as a repository of pre-constructed assets, plugins, and tools. This connection enables developers to enhance their projects using a wide range of pre-existing materials, accelerating development and promoting a collaborative ecosystem within the Unity community.

2.0.2 Blender

Blender is a free and open-source 3D creation suite that includes a comprehensive array of features for each stage of the 3D development process. It gives artists and developers with the capacity to mold, animate, and bring their digital creations to life. Blender is a critical tool in our game production toolkit. It is used to produce unique assets and alter old ones to satisfy the changing requirements of our project. This part covers the main characteristics of Blender, defining its value and providing

extensive knowledge on crucial components that are vital for our efforts in producing and manipulating assets.

Key Features of Blender

Blender offers an extensive array of features and tools specifically designed for modeling purposes. These are the features that we will gain advantages from during the development of our product.

Mesh Manipulation and Poly count Optimization Blender demonstrates exceptional proficiency in mesh manipulation, offering thorough command over vertices, edges, and faces. This feature is extremely important as we shape and improve in-house unique assets , guaranteeing their smooth integration into our game environment. Moreover, Blender's features for reducing poly count allow us to achieve a equilibrium between visual accuracy and operational efficiency.

Texturing in Blender Texture mapping is a crucial element in improving the appearance of assets, and Blender's robust texturing features enable us to infuse life into our projects. Blender's user-friendly features allow us to effortlessly apply and edit textures on complex 3D surfaces, hence improving the overall visual appeal of our game assets.

Rigging and Animation The animation and rigging capabilities of Blender enable us to animate and manipulate figures and objects, giving them a lifelike quality. Blender offers a wide range of capabilities for the animation process, allowing users to create dynamic gameplay characters and add subtle movements to ambient items.

2.1 Related Works

In the following sections, we will explore games that incorporate aircraft combat and missions involving the rescue of individuals. There exists an extensive range of mobile games within the category of aerial combat and rescue games. We will be discussing a select handful that accurately reflect our intended vision.

2.1.1 War Dogs : Air Combat Flight S

[1] "War Dogs" is an immersive air combat flying simulator that takes place during World War II. The game features a wide range of 24 warplanes from prominent nations such as the USA, Germany, UK, Japan, and Russia. These include various types of aircraft such as dogfighters, dive-bombers, torpedo-bombers, and heavy bombers.

Key Features

- Campaign Mode: Campaign Mode offers immersive single-player experiences in five significant conflict zones, spanning from the arid landscapes of North Africa to the coastal regions of Japanese islands.
- Strategic Warfare: The game include mission that engage you in operations such as landing on aircraft carriers, launching torpedoes at enemy warships, and executing precise airstrikes.
- Multiplayer System: The game also involves online multiplayer system. There is an Arena style Team battles where you can invite other players, choose planes and engage in aerial combat.

2.1.2 Fighter Pilot: HeavyFire

[2] "Fighter Pilot: HeavyFire" is a mobile game that immerses players in a spectacular 3D aviation simulation and fast-paced action-adventure. The player assumes the position of a fighter pilot, commanding renowned contemporary jet combat aircraft equipped with authentic and powerful weaponry. The game immerses the player in the intense air combat, providing assistance to army and naval units involved in warfare on the ground and at sea.

Key Features

- Aircraft upgrade System: As player make progress in game, they can upgrade your plane. Advance through aircraft tiers, gaining access to sophisticated features such as the A10 Warthog. Player can optimize aircraft performance by modifying engine, armor, and gunnery variables. Player can equip the aircraft with weaponry that draws inspiration from real-world technology, like rockets and missiles, to enhance players readiness for combat.
- Progressive Levels: The game entails Players will encounter increasingly difficult levels that assess their flying abilities and capacity to adapt as they become more proficient in managing various aircraft.
- Mission based Game: The game comprises a diverse range of missions that become available when you successfully finish the current mission at hand.

2.1.3 City Rescue: Fire Engine Games

[3] "City Rescue: Fire Engine Games" provides players with a unique emergency encounter through exhilarating fireman missions. The game features a diverse range

of captivating missions, where players must tackle difficult tasks across all modes of this exhilarating city rescue game.

Key Features

- Rescue Mode: Rescue Mode involves the player assuming the position of a firefighter in a 3D Truck Driving game. The player is tasked with completing difficult tasks to rescue individuals and animals from burning structures, automobiles, or other urgent situations. The player must employ their abilities and utilize equipment such as hoses, ladders, axes, and fire extinguishers in order to extinguish fires and rescue casualties. The player passes obstacles and hazards, successfully accomplishing every duty within a specified time frame, and accumulating points according to their performance.

- Aerial Mode: Certain missions include the player assuming the role of a helicopter pilot, ascending into the atmosphere and deploying water or fire-retardant substances to combat extensive fires. The player must effectively manage height, velocity, and trajectory while precisely targeting the objective.

- Strategic Missions: During missions, the player must negotiate various environments, such as areas filled with smoke and darkness, while remaining cautious of concealed hazards like gas leaks or booby traps. The player must successfully accomplish objectives while adhering to a specified battery limit, demonstrating stealth and earning points based on their performance. The player must successfully do missions within a specified distance constraint, demonstrating speed, and accrue points based on their level of effectiveness. The player must carefully replenish the water or fire-retardant tank, while avoiding collisions and completing missions within a limited fuel capacity in order to collect points.

2.1.4 Emergency Firefighter Police

[4] "Emergency Firefighter Police" offers players the opportunity to join the police, EMS, or fireman department and participate in distinctive rescue missions in a vibrant metropolis. The player assumes the role of an ambulance driver, participating in high-stakes pursuits with police vehicles, and fulfilling the duty of transporting injured individuals to emergency locations in a realistic ambulance driving simulator.

Key Features

- Variety of Rescue Missions: The player joins the police, Emergency Medical Services (EMS), or firefighting department, each with distinct vehicles and equipment. The player must respond quickly to problems, including criminal pursuits, accidents, and fires, while assuming various roles to ensure the safety of the city.

- Three Rescue Department options: Police Department: Operate a law enforcement vehicle, pursuing individuals engaged in criminal activity and illicit street racing on urban expressways. The player has the ability to equip themselves with a police car and a range of weaponry in order to combat criminal behavior in the city.

- City Ambulance Simulator: The player assumes the position of an emergency medical specialist, responsible for treating patients' injuries, making diagnosis, and quickly responding to accident sites in a city ambulance.

- Firefighter Truck: The player has the opportunity to join the rapid emergency response service, fully armed and prepared to combat fires within the city. The players can equip themselves with firefighter rescue equipment, and remain vigilant at the fire station dedicated to rescue operations.

2.2 Limitations

Each game stated above exemplifies and represents its respective genre, fully living up to its title. In the limitations section, we will be highlighting limitations from the perspective of our project. Our game combines the fundamental elements of aerial combat games and rescue games, featuring a compelling narrative centred around a lone heroic pilot and their missions. However, these games have certain constraints. The game "War Dogs: Air Combat Flight S" does not have a narrative plot; instead, it is structured as a sequential mission-based game. Additionally, our game strives to incorporate rescue missions, a feature that is absent in this game. The second game, "Fighter Pilot: HeavyFire," shares the same constraint. The game lacks a coherent narrative structure. The game features missions that gradually increase in difficulty. Our objective is to address this deficiency and introduce rescue missions in our "Airplane Citizen Rescue Game". The game "City Rescue: Fire Engine Games" combines on-field and vehicular rescue missions with airborne rescue missions. The game is mission-oriented and lacks a storyline. The game "Emergency Firefighter Police" offers many modes such as police rescue, ambulance rescue, and firefighter rescue, but it does not include aerial rescue missions. This game is mission based; hence it lacks a storyline. Our objective in the game "Airplane Citizen Rescue

Game" is to enhance the significance of our rescue missions by linking them into the main storyline, strengthening their worth to our protagonist character.

Chapter 3

System Requirements, Architecture, and Design

This chapter elaborates functional requirements, non-functional requirements, architecture, and design. (Required Word Count: 1,000 to 1,500 Words).

3.1 System Requirements

3.1.1 Functional Requirements

Two modes gameplay:

- The game should incorporate an Aerial Rescue Mode, in which the player engages in missions focused on rescuing individuals.
- The game must feature an Aerial Combat Mode that transitions into a combat-focused experience, where players engage in violent engagements against extraterrestrial invaders in their aircraft.

Upgradable Aircrafts:

The game must include aircraft and helicopters that are upgradable in speed, attack power, armor and maneuverability.

Garage System:

- The game must include a garage system that enables players to look at their owned aircraft. The user has the ability to navigate through the many aircraft options in the garage by swiping.
- The improvement of weapon attack, armour, and the maneuverability of aircraft should be executed at the garage.

Interactive Environment:

The game must include an interactive environment that reacts to player actions.

Player Progression:

The game must include a progression system that enables players to advance through story and unlock new missions, types of aircraft, and challenges. This advancement should be based on the successful completion of missions in both gameplay modes.

Sound and visual effects:

The game must incorporate sound effects and visual effects.

Saving Progress:

The game must save the progress of the player as they advance through the game and unlock missions and aircrafts.

2D Heads-Up Display (HUD):

During gameplay, it is essential to have a HUD that prominently displays key features such as a map, ammunition details, and the health status of the aircraft.

3.1.2 Non Functional Requirements

Performance Optimization: The game should be optimised to guarantee seamless performance, while minimising any possibility of lag or frame rate problems.

Scalability: The game's architecture should be planned to incorporate possible future updates and modules, thus eliminating the necessity for major reengineering.

Responsive User Interface: The HUD and other UI elements should demonstrate responsiveness, delivering real-time feedback to the player's actions and guaranteeing a smooth interactive experience.

Load Time: The loading times before starting and after ending missions must be minimum of 3 seconds and maximum of 5 seconds.

3.2 Flowchart

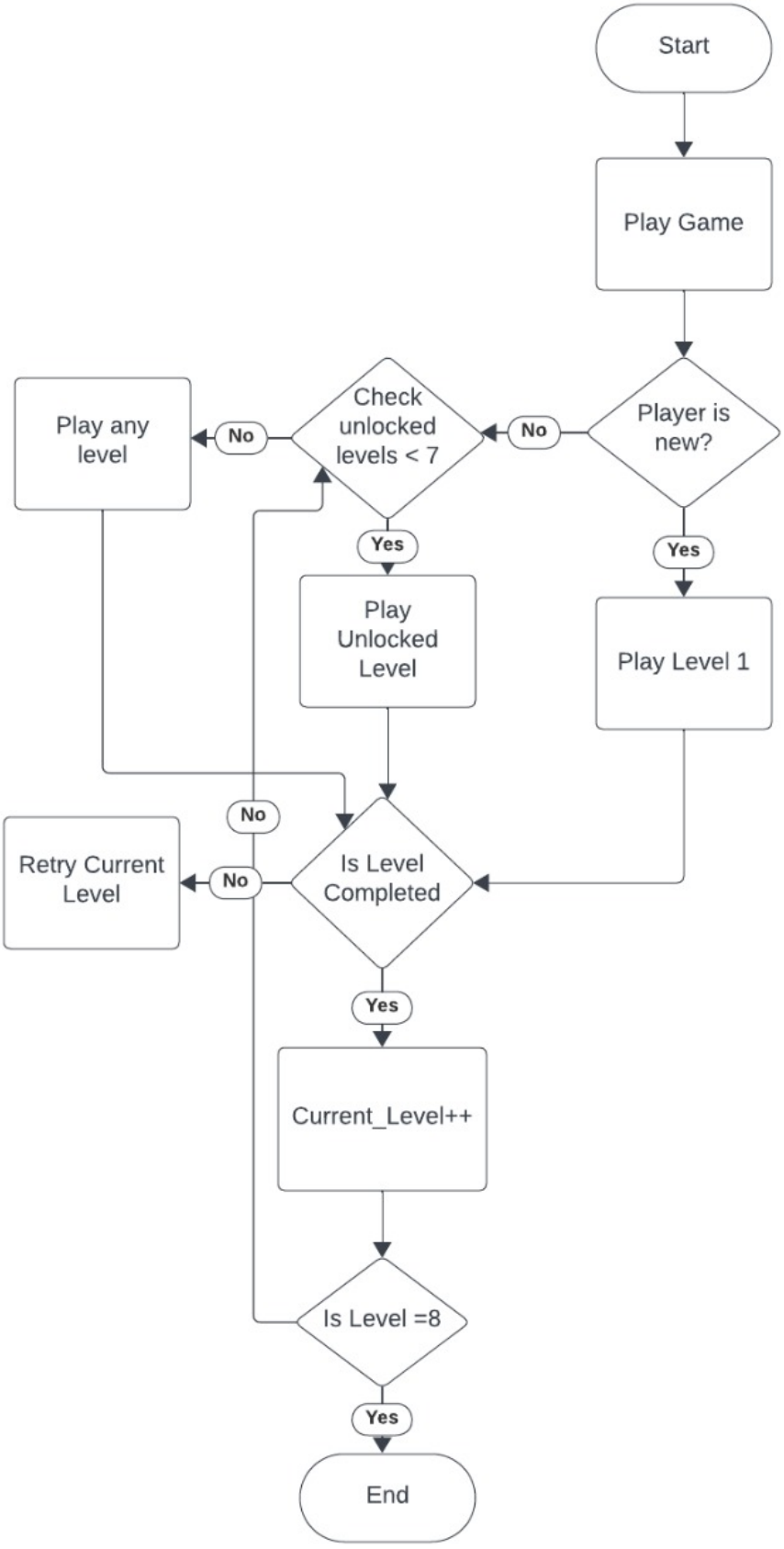


Figure 3.1: Game Flowchart

3.3 Use Case Diagram

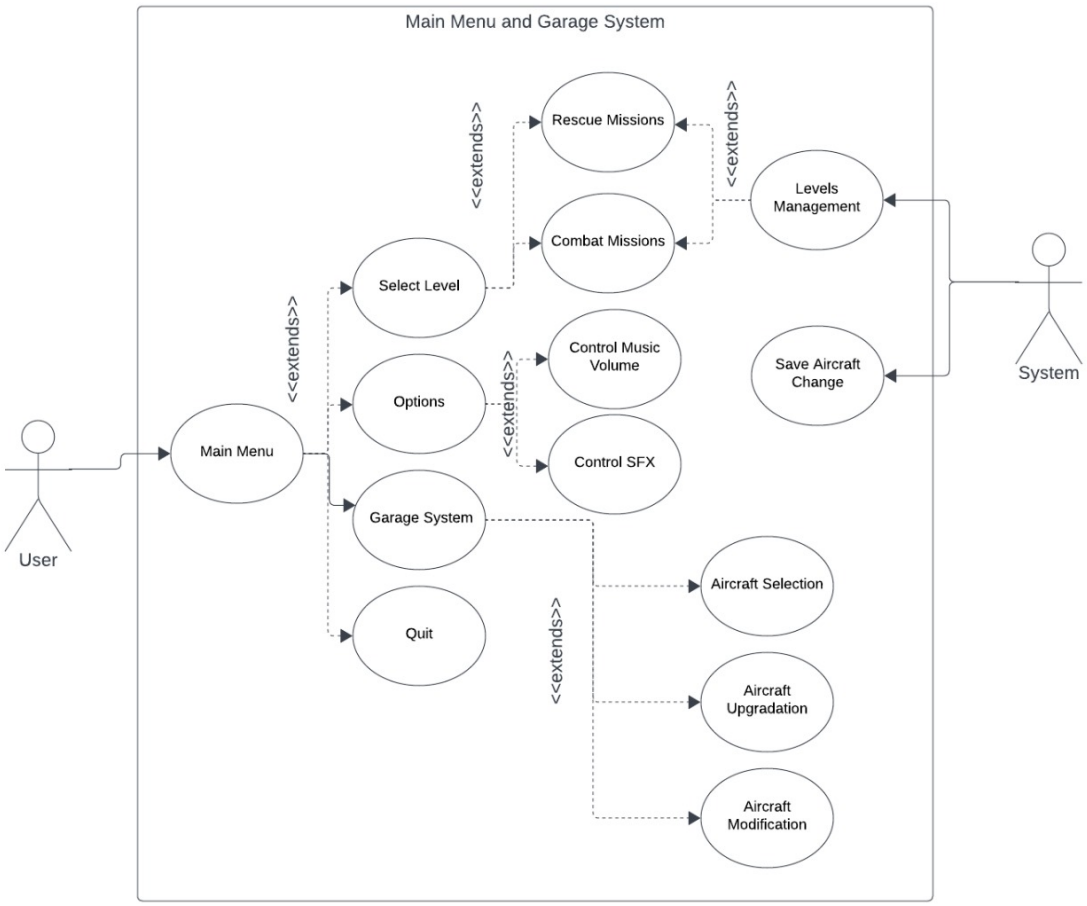


Figure 3.2: Menu and Garage system UCD

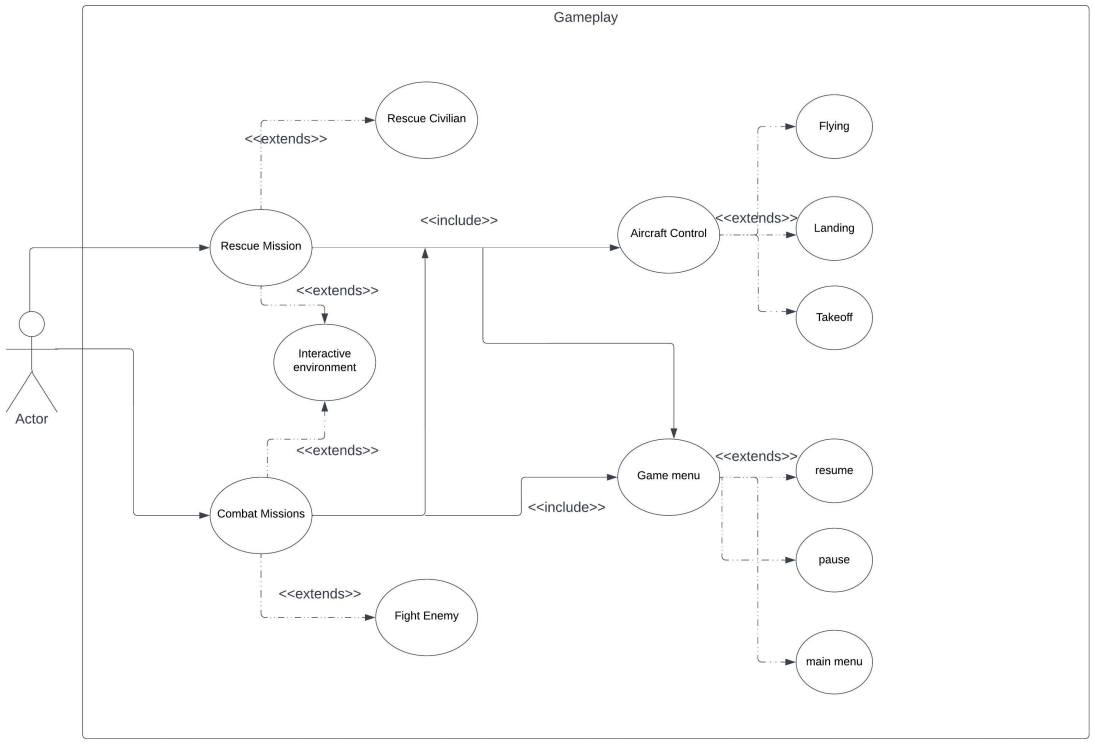


Figure 3.3: Gameplay system UCD

3.4 Software Development Plan

3.4.1 Software Architecture

High-Level Overview

Our game applies a Component-Based Architecture as its software architecture, which serves as the fundamental behind our game development. This architecture plays a crucial role in accomplishing our game's goals and fulfilling the set criteria.

Importance of Component-Based Architecture

The Component-Based Architecture is chosen for its inherent modularity and flexibility. Through the process of breaking down the system into distinct, reusable elements, we provide a modular framework that promotes simplicity in construction, maintenance, and scalability. Every component encompasses distinct functionalities, which enhances the ability to reuse code and reduces its dependence on other components. The Component-Based Architecture facilitates future expansions and changes, ensuring scalability without causing any disruption to the current structure. Moreover, the architecture promotes collaboration among development teams. Specialised teams can simultaneously work on multiple components, facilitating parallel workstreams and an efficient development process.

System Components

The following are the primary game components that will be essential in determining and influencing the fundamental behavior of the game.

1. User Interface:

   The primary menu System: manages the main interface, allowing navigation and enabling access to different game features.

2. Garage System:

   Enables the modification and upgrading of player aircraft, providing a dynamic environment for increasing the gaming experience.

3. Input Manager:

   Manages user inputs to ensure timely and precise engagement with the game.

4. Environment:

   Manages the in-game environment and general gameplay configurations, facilitating a dynamic and captivating gaming experience.

5. Visual Effects:

   This component is responsible of generating and managing visual effects throughout gameplay, including as collisions, damage indicators, and other visually striking elements.

6. Aircrafts mechanics:

   Oversees the mechanical aspects of each aircraft throughout gaming, guaranteeing authentic and timely performance during flight and combat scenarios.

7. Enemy Control:

   Controls the actions and behavior of enemies during gaming, ensuring that the player faces dynamic and demanding fights.

8. Game Physics:

   manages the mechanics of game physics, including collision mechanisms and object interactions like bouncing, which enhance the overall authenticity of the gaming experience.

9. Audio System: The audio system includes sound effects and music into the game, enriching the auditory experience and complementing in-game events.

10. Gameplay Involves crucial elements such as the evolution of the plot and the advancement of levels, which direct the player through the developing storyline and missions.

11. Database: The database component stores and organizes all progress and data created during gameplay, guaranteeing a smooth and uninterrupted experience, and enabling players to continue from their most recent saved point.

High Level Architecture Diagram

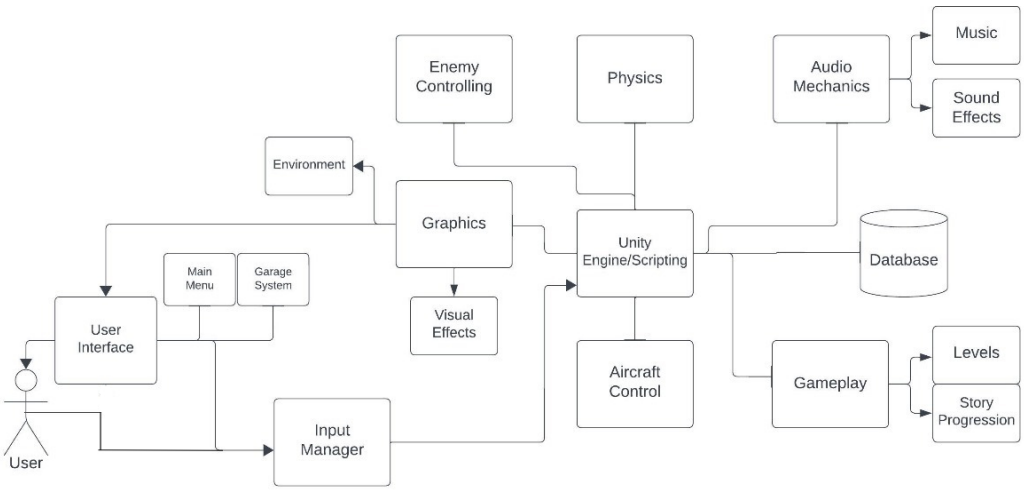


Figure 3.4: High Level architecture Diagram

3.4.2 Number Of Iterations

Iteration 1

Tasks Completed

- Implemented navigation controls for the helicopter and plane, utilizing a 2D joystick interface.
- The helicopter can rotate, move forward, and go backward. These movements are controlled by a two-dimensional circular joystick.
- The helicopter now has buttons to adjust its altitude and to increase or decrease the speed of the rotor.
- Adopted a two-dimensional circular joystick to execute rotation and elevation commands for maneuvering aircraft.
- Installed a nitro button on the aircraft to enhance its speed.

Obstacles Faced and Solutions

Applying a Joystick:

- A 2D circular joystick was studied and implemented to ensure precise and simple control over the aircraft and helicopter's movements.

Nitrous Performance:

- Implemented a nitro button to enhance the speed of the aircraft.

Achievements of the iteration

- Successful integration of joystick controls for manoeuvring both helicopters and planes.
- The aircraft is equipped with functional height adjustment and fan speed controls.
- The plane has been equipped with nitro capability to enable dynamic speed shifts.

Progress Towards Overall Project Goal

- Plane and helicopter movement: Made substantial advancements by effectively incorporating the fundamental mechanics of motion for both aircraft.
- User Controls: Implemented intuitive controls utilising a joystick for seamless manipulation of rotation and elevation, hence boosting the overall player experience.
- Game Dynamics: Implemented the nitro feature, enhancing the action with an exciting factor.

Iteration 2

Tasks Completed

- Developed a Garage System that enables users to select between helicopters and planes.
- Implemented a feature that allows users to access several models for each aircraft type, enabling them to navigate between them using "Previous" and "Next" buttons.
- Implemented a "Equip" button to enable and save the chosen aircraft in the database for future gameplay sessions.
- Implemented database integration to store the user's selected aircraft for future sessions.

Obstacles Faced and Solutions

Model Navigation:

- Implemented a user-friendly navigation system featuring "Previous" and "Next" buttons to facilitate browsing of available aircraft models.

Database Integration:

- Implemented a connection with the database to save and retrieve user-provided aircraft, ensuring a smooth experience across game sessions.

Achievements of the iteration

- Successful implementation of the Garage System, enabling users to select between helicopters and planes.
- Seamless navigating between several aircraft models within each category, improving user customizing possibilities.
- The "Equip" button allows users to activate and save their selected aircraft for future usage.

Progress Towards Overall Project Goal

- User Customization: Significantly improved user customization options with the introduction of the Garage System.
- Persistent User Preferences: Developed database integration to facilitate the storage and retrieval of user-configured aircraft, hence improving the user experience across multiple sessions.
- Preparation for Future Features: Established the groundwork for future improvements regarding player personalization and aircraft choice.

Iteration 3

Goals And Objectives:

Objective 1: Integrate the Garage System into the gameplay, guaranteeing that the selected aircraft is used in every level.

Objective 2: Implement collision physics for aircraft, resulting in destruction when they come into contact with solid surfaces.

Objective 3: Implement landing capabilities for both aircraft and helicopters, with related auditory and visual effects.

Objective 4: Address any adjustments or enhancements based on user feedback from previous iterations.

Iteration 4

Goals And Objectives:

Objective 1: Develop a captivating gaming environment featuring urban landscapes, road networks on varied terrain, and flowing rivers to heighten the total player engagement.

Objective 2: Develop interactions among aircraft, helicopters, and the environment, guaranteeing an authentic and dynamic gaming experience.

Objective 3: Develop a initial storyline to introduce player progression and context for the game world.

Objective 4: Initiate development on the initial four levels to provide the groundwork for player progression.

3.5 Fully Dressed Use Cases

3.5.1 USE CASE: Play Game

Use Case Name: Play Game
UC ID: 01
Actor: User/Player
Pre-condition: The application should have launched correctly.
Normal flow:

1. On the title screen the main menu populates the screen.
2. User is displayed with 3 buttons (play, options, quit)
3. Upon pressing the play game button, the application should take the user to the gameplay screen.

Alternate flow:
2a Player chooses options button instead:

- User is taken to option menu.
- User manipulates settings and presses back.
- Use case resumes at step 2.

2b Player chooses the quit button:

- The game is terminated upon pressing quit.

Post Condition: The game transitions from main menu to active state.

3.5.2 USE CASE: Pause Game

Use Case Name: Pause Game
UC ID: 02
Actor: User/Player
Pre-condition:

- The player is actively engaged in the gameplay.
- The game is not already paused.

Normal Flow:

1. The player presses the pause button during gameplay.
2. All movement and gameplay animation stops.
3. Game timer ceases to progress.
4. Mission sounds and music stops playing.

Post Condition: Game is paused and displays the pause menu.

3.5.3 USE CASE: Resume Game

Use Case Name: Resume Game
UC ID: 03
Actor: User/Player
Pre-condition:

- The game is currently paused.
- The player has not exited or returned to the main menu of the game.

Normal Flow:

1. The player selects “Resume Game” from the pause menu.
2. The pause menu fades away.
3. All movement and gameplay animation resumes.
4. Game timer resumes to progress.
5. Mission sounds and music resumes playing.
6. Player regains the control of their character.

Post Condition: Gameplay resumes from the exact point from where the player had paused.

3.5.4 USE CASE: Exit Game

Use Case Name: Exit Game
UC ID: 04
Actor: User/Player
Pre-condition: The game should be on the title screen or the game should be paused.
Normal Flow:

1. From Main Menu:
   (a) Player selects “Exit game” from the main menu.
   (b) The game closes.
2. From Pause Game:
   (a) Player has paused the game during gameplay.
   (b) Player selects “Exit game” from the pause menu.
   (c) Confirmation prompt appears. Yes/No.
   (d) Player selects “Yes”.
   (e) The game closes.

Alternate Flow:

- The player quits the game by pressing back or home button of their device.

2c. Player selects "No":

- Player is taken back to the pause menu.
- The use case resumes at 2a.

Post Condition:

- Game process terminates.
- Game resources are released from memory.
- Game is no longer running.
- The player is returned to their device's interface.

3.5.5 USE CASE: Change SFX/ Music

Use Case Name: Change SFX/ Music
UC ID: 05
Actor: User/Player
Pre-condition:

- The game is launched.
- The player is either at the main menu or in active gameplay.

Normal Flow:

1. From the Main Menu or Pause Menu:
   (a) The player selects "Options" button from the Main Menu or Pause Menu.
   (b) The option menu appears.
   (c) Player is given 2 options (SFX, Music)
   (d) Player can manipulate the sound as per need.
   (e) Player confirms the changes and exits the option menu.

Alternate Flow:

- After 1d, the player does not confirm changes and exits the option menu without saving sound changes. The sound remains unchanged.

Post Condition: The sound of the game is changed successfully as per user selection.

3.5.6 USE CASE: Aircraft upgrade

Use Case Name: Aircraft upgrade
UC ID: 06
Actor: User/Player
Pre-condition:

- The game is launched.
- An aircraft (either a jet or a helicopter) is equipped.
- The game mission has not been started.

Normal Flow:

1. The player selects the "Garage" option.
2. The player is taken to the Garage menu.
3. The player selects "Upgrade" option.
4. The player is taken to the Upgrade menu.
5. The player tweaks the aircraft specification.
6. Credit is deducted according to level of upgradation.
7. The player saves the changes.
8. The player exits garage.

Alternate Flow:
5a. Aircraft specifications upgrade:

- AF1 The player will have the option to upgrade the armor, speed, attack, and handling of the aircraft. After that, the use case resumes.

6a. Credit deduction:

- AF2 If the player has enough credit to afford upgrades, credit will be deducted accordingly and the use case continues.
- AF3 If the player lacks credits, upgrade will not be performed and the use case resumes from step 4.

Post Condition:

- The specifications of the aircraft are updated and saved.
- Credit is deducted from player's credit score.

3.5.7 USE CASE: Combat Missions

Use Case Name: Combat Missions
UC ID: 08
Actor: User/Player
Pre-condition:

- The desired mission in unlocked.
- The player has selected the mission from mission list.

Normal Flow:

1. Player maneuvers the aircraft towards the mission area.
2. Upon reaching the designated area, the enemy monster appears.
3. The player and the monster engage in combat.
4. The mission ends in either a win/loss for the player.

Alternate Flow:
4a. Damage taken:

1. During combat, player might take damage from monster attack, which will reduce the jet health.

5a. Mission Complete:

1. The player defeats the monster before their jet gets destroyed.
2. Mission is complete.

5b. Mission Failed:

1. The player's jet gets destroyed first by the monster.
2. Mission Failed

Post Condition:
PC 5a. Mission Complete:

1. Mission completion screen is displayed with rewards (credits, new mission unlock, etc.).
2. Player is returned to the mission select screen.

3. Player's progress in the game's overall story or campaign is saved.

PC 5b. Mission Failed:

1. Mission failure screen is displayed with options to retry, return to mission select, or exit to main menu.
2. Player's progress in the mission is not saved.
3. New mission is not unlocked.

3.5.8 USE CASE: Rescue Missions

Use Case Name: Rescue Missions
UC ID: 09
Actor: User/Player
Pre-condition:

- The desired mission in unlocked.
- The player has selected the mission from mission list.

Normal Flow:

1. A text briefing details the objective and anomaly information.
2. Player maneuvers the aircraft towards the rescue area.
3. Upon reaching the designated area, the mission timer starts.
4. The player starts rescuing troubled citizens, competing against time.
5. The mission ends in either a success or failure for the player.

Alternate flow:
5a. Mission Complete:

1. The player completes all the objectives within given time.
2. Mission is complete.

5b. Mission Failed:

1. The player fails to complete the objectives within given time.
2. Mission Failed

Post Conditions:
PC 5a. Mission Complete:

1. Mission completion screen is displayed with rewards (credits, new mission unlocks, etc.).
2. Player is returned to the mission select screen.
3. Player's progress in the game's overall story or campaign is saved.

PC 5b. Mission Failed:

1. Mission failure screen is displayed with options to retry, return to mission select, or exit to main menu.
2. Player's progress in the mission is not saved.
3. New mission is not unlocked.

3.5.9 USE CASE: Unlock new Aircraft

Use Case Name: Unlock new Aircraft
UC ID: 10
Actor: User/Player
Pre-condition:

- The player has not started the mission.
- The player is in the garage.
- The player has enough credit to unlock the desired aircraft.

Normal Flow:

1. In the garage menu, player selects "View Aircrafts" button.
2. The player browses from variety of aircrafts (jets and helicopters)
3. The player clicks "Unlock aircraft" button.
4. Credit is deducted for purchasing new aircraft.
5. The player exits the garage.

Alternate flow:
3a. Confirmation prompt:

- The player is prompted Yes/No to confirm or reject the purchase

Post Condition:

- Defined credit is deducted for that aircraft.
- The unlocked aircraft is added to the players collection to equip.

3.5.10 USE CASE: Equip a different Aircraft

Use Case Name: Equip a different Aircraft
UC ID: 11
Actor: User/Player
Pre-condition:

- The player has not started the mission.
- The player is in the garage.
- The desired aircraft is already unlocked. desired aircraft.

Normal Flow:

1. In the garage menu, player selects "View Aircrafts" button.
2. The player browses from variety of aircrafts (jets and helicopters)
3. The player selects the desired aircraft and clicks on the "Equip" button.
4. The player exits the garage.

Post Condition:

- The selected aircraft is equipped for the next mission.

3.5.11 USE CASE: Replay A Previous Mission

Use Case Name: Replay A Previous Mission
UC ID: 12
Actor: User/Player
Pre-condition:

- The player is completed some missions.
- The player is in the mission select menu.

Normal Flow:

1. In the mission select menu, player browses through missions.
2. The player selects the desired mission.
3. The mission is selected.

Alternate Flow:
2a. Locked mission:

- The player selects a locked mission.
- The player is prompted that the mission is locked.
- Use case resumes at step 1.

Post Condition:

- The selected mission starts playing.
- The screen transitions to the mission gameplay.

3.5.12 USE CASE: Controlling Aircraft

Use Case Name: Controlling Aircraft
UC ID: 13
Actor: User/Player
Pre-condition:

- The player is currently in active gameplay.
- The player is in control of the aircraft.

Normal FLow:

1. The player interacts with the on-screen circular joystick.
2. The aircrafts respond to the movement of joystick is real time.
3. The player adjusts the aircraft direction by moving the joystick in different directions.

Alternate flow:

- If the player releases joystick, the aircraft remains in its current direction.

Post Condition: The aircraft's movement is adjusted according to the player's input.

3.5.13 USE CASE: Weapon Interaction

Use Case Name: Weapon Interaction
UC ID: 14
Actor: User/Player
Pre-condition:

- The player is currently in active gameplay.
- The player is in control of the aircraft.

Normal Flow:

1. The player interacts with the on-screen artillery buttons (e.g., fire bullets, fire missiles).
2. The selected artillery action is executed by the aircraft.

Alternate flow:

- If the player doesn't interact with the artillery buttons, no artillery action is taken.

Post Condition: The selected artillery action is performed, affecting the game environment.

3.5.14 USE CASE: Boosting Mechanics

Use Case Name: Boosting Mechanics
UC ID: 15
Actor: User/Player
Pre-condition:

- The player is currently in active gameplay.
- The player is in control of the aircraft.

Normal Flow:

1. The player interacts with the on-screen boost button.
2. The aircraft receives a speed boost, enhancing its movement temporarily.

Alternate Flow:

- If the player doesn't interact with the boost button, the aircraft maintains its regular speed.

Post Condition: The aircraft experiences a boost in speed for a specific duration.

3.6 System Sequence Diagram (SSD)

3.6.1 Combat Mission SSD

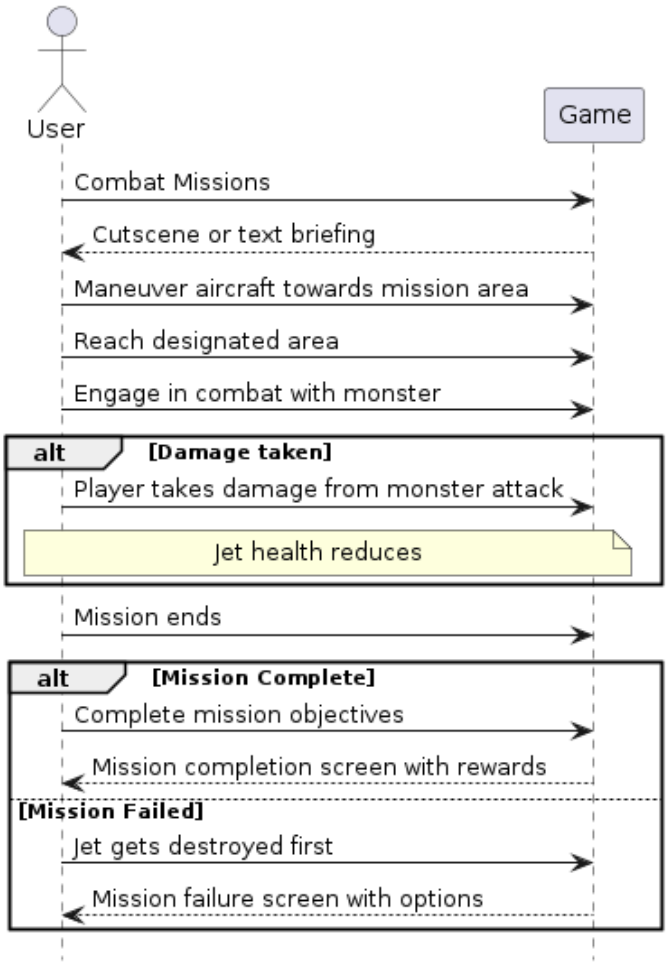


Figure 3.5: Combat Mission SSD

3.6.2 Rescue Mission SSD

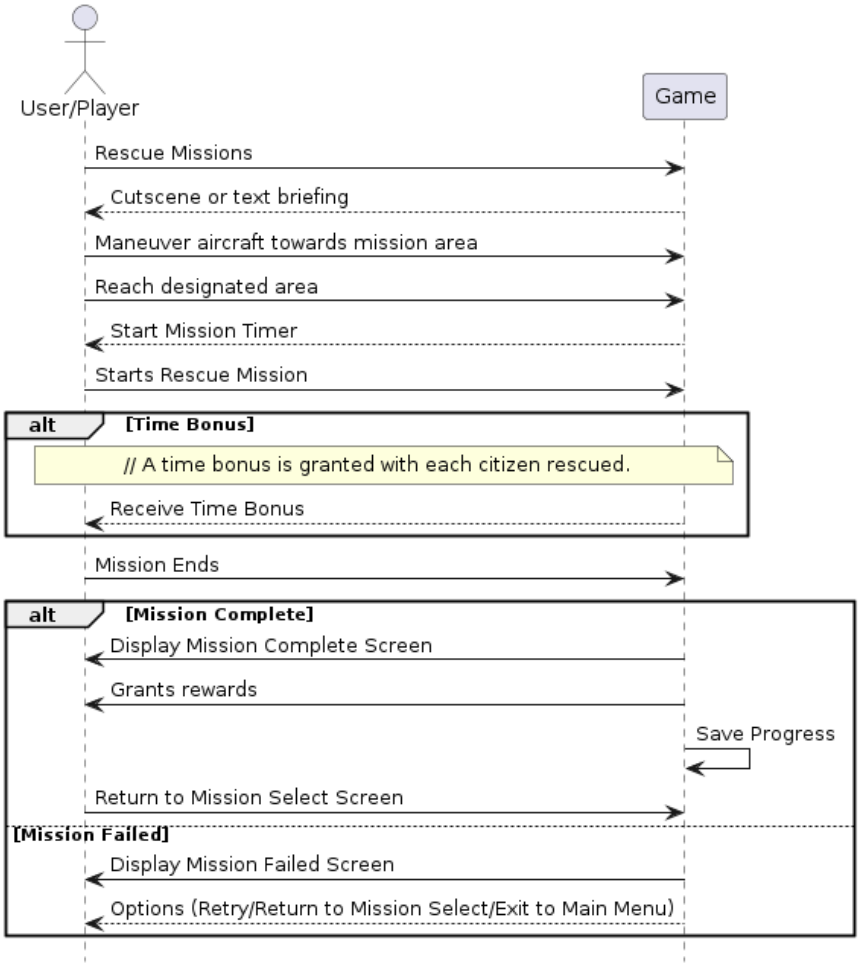


Figure 3.6: Rescue Mission SSD

3.6.3 Unlock New Aircraft SSD

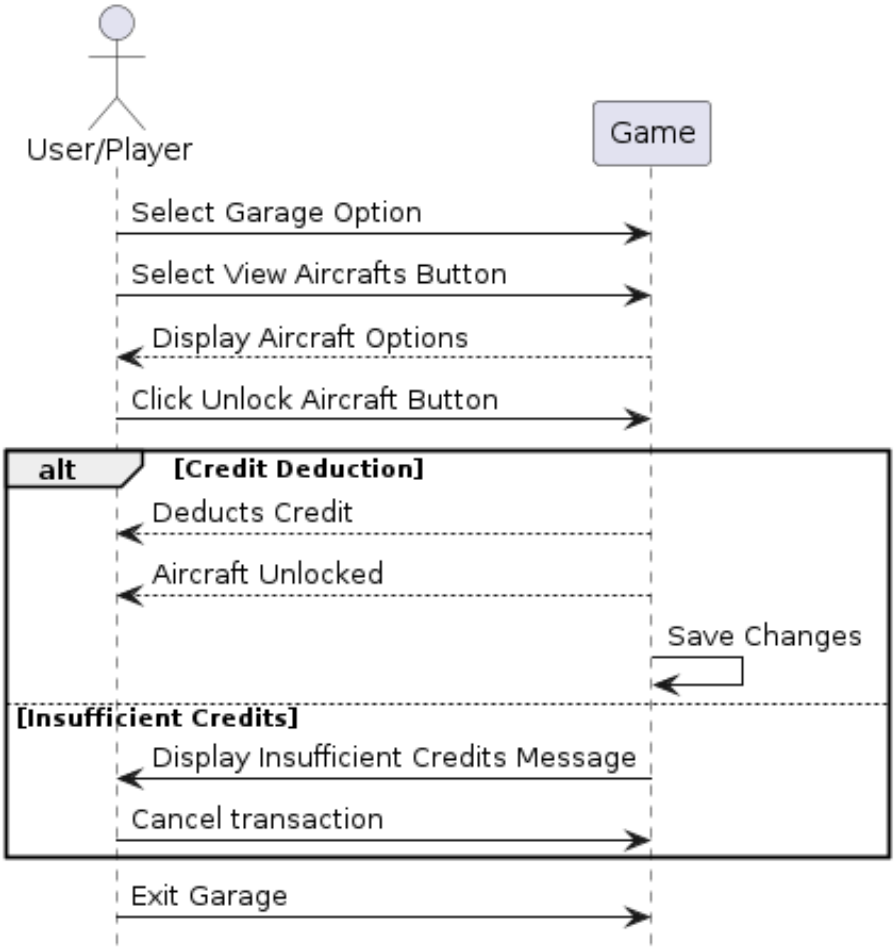


Figure 3.7: Unlock Aircraft SSD

3.6.4 Upgrade Aircraft SSD

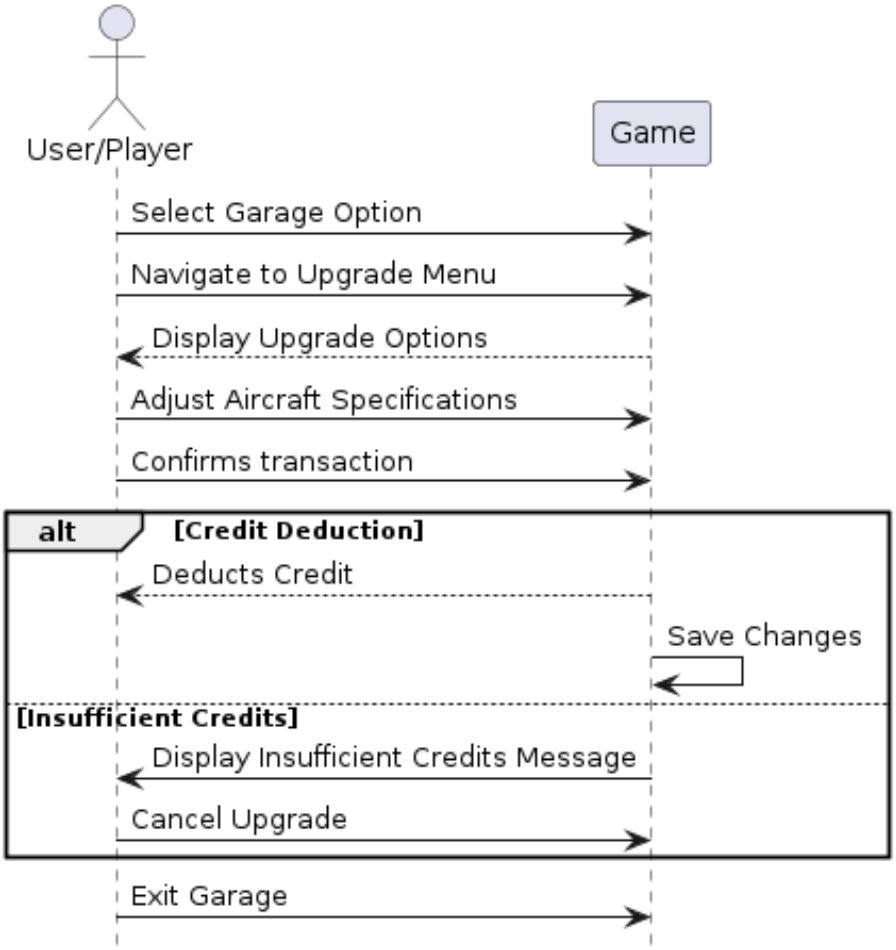


Figure 3.8: Upgrade Aircraft SSD

3.7 Software Requirements Specification (SRS) Document

3.7.1 Introduction

The Software Requirements Specification (SRS) document defines the fundamental elements and capabilities of the "Airplane Citizen Rescue Game," a dynamic and immersive 3D video gaming project. This game was created as part of an undergraduate program. It immerses players in the role of a skillful pilot, participating in aerial combat missions and high-stakes rescue operations. The purpose of the SRS document is to offer a thorough comprehension of the game's goals, characteristics, and technical requirements. It serves as a guiding resource during the development and testing stages.

3.7.2 Purpose of Document

The purpose of this document is to provide a clear overview of the Airplane Citizen Rescue Game Project. The purpose is to outline the functional and non-functional requirements of Airplane Citizen Rescue Game. This document facilitates successful communication and collaboration among developers, testers, and project stakeholders by providing a comprehensive and well-organized overview. This helps to establish a shared understanding throughout the software development lifecycle.

3.7.3 Project Scope

The game "Airplane Citizen Rescue Game" is divided into two modes of missions, one being the aerial combat missions and the other missions focused on rescuing citizens from disasters e.g. natural disasters, fire disaster. There are total of 7 missions in the game, with 2 being tutorial missions 2 aerial combat missions and 2 being the rescue missions and 1 being the final mission. The development of the game is done for Android platforms. The game for now does not work for Windows. The game is a mixture of 3D and 2D. All gameplay aspects of the game are to be built in 3D while majority of the user interface (UI) is to be built in 2D. For the development, unity 3d game engine is being utilized based on C sharp. The major game assets including (jets, helicopters, city landscapes) are sourced from unity's official asset store. While low level game assets are to be built inhouse in Blender 3D The game does not include multiplayer support and is based on a single player style. The primary demographic for our game caters mostly to younger audience and teenagers, providing a captivating and immersive experience that suits their tastes. However, the game's inclusive design makes sure that players of all ages may have fun.

3.7.4 Game Requirements

Functional Requirements

1. **Campaign Mode:**

   [**FR1.01**] The game must be based on a campaign mode. i.e. it should follow a defined storyline based on the hero.
   [**FR1.02**] The campaign must be divided into 7 missions.
   [**FR1.03**] At the start of each mission, there should be a cutscene that defines the mission objectives, enemy details or rescue details.

2. **Mission Modes:**

   [**FR2.01**] The missions of the game shall be including 2 types. i.e. combat missions and rescue missions.
   [**FR2.02**] there should be 4 combat missions and 3 rescue missions.

3. **Combat missions:**

   [**FR3.01**] Combat missions are to be played with fighter jet rather than a helicopter.
   [**FR3.02**] Combat mission must include an enemy that engages in combat with the player.
   [**FR3.03**] The mission is completed once all the defined objectives are accomplished.
   [**FR3.04**] The mission is failed if the aircraft is destroyed before completing the objectives.

4. **Rescue Missions:**

   [**FR4.01**] Rescue missions are to be played with a rescue helicopter rather than a fighter jet.
   [**FR4.02**] The rescue missions are time boxed. i.e given objectives must be completed within given time frame.
   [**FR4.03**] With each citizen rescued, player will be given a time bonus of +5 seconds which will be added to the timer.
   [**FR4.04**] The mission is completed once all objectives are completed within time.
   [**FR4.05**] The mission is failed if the player does not complete objectives within given time.

5. **Unlocking Missions:**

   [**FR5.01**] A mission can only be unlocked if the mission prior to that has been completed.

[**FR5.02**] Once a mission is unlocked, it can be replayed from the mission select menu.

6. **Reward system:**

   [**FR6.01**] Upon completing a mission, player will be given reward in form of "Credits".
   [**FR6.02**] After a mission is completed, the next mission will be automictically unlocked as a reward.

7. **In-game Credits:**

   [**FR7.01**] In-game currency shall be called as "Credits".
   [**FR7.02**] Credits can be used to unlock new aircrafts.
   [**FR7.03**] Credits can be used from aircraft customization that includes aircraft control and specifications.

8. **Mission Failed:**

   [**FR8.01**] If the player fails a mission (reference [**FR3.04**] [**FR4.05**]) no reward shall be granted to the player.
   [**FR8.02**] The player must be prompted with options to retry, return to mission select, or exit to main menu.

9. **Garage System:**

   The game must include a garage system that includes following functionality:
   A. Aircraft Customization:
   [**FR9.A1**] The player can customize an aircraft from the garage menu. Customization options include:
   I. Specification Upgrade:
   [**FR9.A2**] The player can enhance the aircraft performance in following aspects:
   a. Armor for increased aircraft health.
   b. handling for better maneuverability.
   c. Attack power for increased damage.
   e. Speed for fastening the aircraft.

   II. Unlocking new aircraft:
   [**FR9.A3**] Player can browse through different aircraft in the garage.
   [**FR9.A4**] Player can purchase any aircraft if they have enough credit.

   III. Equip an aircraft:
   [**FR9.A5**] The user can browse through the unlocked aircrafts and equip the aircraft of their choice.

10. **Credit deduction:**
    [**FR10.01**] Upon customizing the aircraft, respective amount of credits will

be deducted from player.

[**FR10.02**] If player does not have enough credit, the player must be prompted saying insufficient credits and the changes in applied to the aircraft will not be saved.

[**FR10.03**] If the player does not have enough credits to unclock an aircraft, the player must be prompted saying insufficient credits.

11. **Game Menus:**

The game will be consisting of following menus each with different functionalities:

A. Main menu:

[**FR11.A1**] The game must consist of a main menu that is displayed when the game is launched.

[**FR11.A2**] The main menu consists of following options/buttons:

a. Play

b. Options

c. Exit

B. Option Menu:

[**FR11.B1**] the option menu must include sound setting to change sound.

[**FR11.B2**] Separate option for SFX and music must be provided to the user.

C. Play Menu:

[**FR11.B3**] The play button will lead to the play menu that include following options:

a. Start new game: starts a new game from mission 1.

b. Continue game: Takes player to mission select of their current campaign.

c. Garage: takes player to the garage.

d. Back: takes player to the main menu.

D. Mission Select Menu:

[**FR11.D1**] The mission select menu showcases the missions in the game.

[**FR11.D2**] The player can select to play any unlocked mission.

E. Pause Menu:

[**FR11.D1**] The player will be capable of pausing the game in between the mission, opening the pause menu.

a. The pause menu must include following buttons:

b. Resume: resumes the game from the point it was paused.

c. Option: Opens the sound changing option.

d. Exit to main menu: Takes the player back to the main menu.

e. Quit: Terminates the game process.

12. **Jet capabilities:**

[**FR12.01**] The jet movements will be controlled via on-screen joystick.

[**FR12.02**] The jet must be capable of shooting bullets and missiles.
[**FR12.03**] The jet must be capable of using a temporary speed boost using boost option.
[**FR12.04**] Dedicated buttons for shooting bullets, missile and boost must be present on screen.

Non-Functional Requirements

1. **Performance Optimization:**
   [**NFR1.01**] The game should be optimized to guarantee seamless performance, ensuring a frame rate ranging from 30-60 frames per second (FPS) and a response time of minimum 100 milliseconds (ms) thus minimizing any possibility of lag or frame rate problems.

2. **Scalability:**
   [**NFR2.01**] The game's architecture should be planned to incorporate possible future updates and modules, thus eliminating the necessity for major re-engineering.

3. **Responsive User Interface:**
   [**NFR3.01**] The HUD and other UI elements should demonstrate responsiveness, delivering real-time feedback to the player's actions and guaranteeing a smooth interactive experience.

4. **Load Time:**
   [**NFR4.01**] The loading times before starting and after ending missions must be minimum of 3 seconds and maximum of 5 seconds.

5. **2D HUD:**
   [**NFR5.01**] The game HUD during gameplay must include elements like map radius, buttons necessary for controlling aircraft (joystick, bullet, missile, boost)

System Requirements (Part of NFRs)

1. **Hardware Requirements:**
   [**SHR1.01**] The mobile device must be equipped with minimum dual-core 1.2 GHz processor.
   [**SHR1.02**] The mobile device must be equipped with minimum 3 GB RAM.
   [**SHR1.03**] The minimum screen resolution shall be 720x1280 pixels.
   [**SHR1.04**] The operating system required shall be Android 8 or later.
   [**SHR1.05**] There is no internet connectivity required.

2. **Implementation Requirements:**

   [**SIR1.01**] The game engine to be used for development must be Unity 3D.

   [**SIR1.02**] The game assets are to be acquired from official Unity Asset store.

   [**SIR1.03**] Development of in-house assets and asset manipulation shall be done in Blender.

   [**SIR1.04**] The game is developed for Android only.

   [**SIR1.05**] The game logic and functionalities along with scripting is to be done in C sharp programming language.

3.8 Test Plan

3.8.1 Introduction

Purpose

The purpose of this test plan is to ensure the quality and reliability of the Airplane Citizen rescue game on the Android platform by outlining the testing approach, methodologies, and responsibilities.

Scope

This test plan encompasses testing activities for gameplay mechanics, integration of game elements, performance on Android devices, preliminary UI usability, and security testing to prevent user exploitation.

Objectives

- To quantify objectives with SMART metrics:
- To achieve zero critical gameplay bugs reported during beta testing.
- To maintain an average frame rate above 30 FPS on devices with 3GB RAM.
- To ensure the game is resilient against security vulnerabilities and exploits.

3.8.2 Testing Levels

Unit Testing:

Objective: Ensure individual game elements function as intended.
Approach: Developers conduct unit tests with a focus on components like airplane controls, citizen interactions, mission completion, and basic security checks etc.
Test Cases:

- Validate responsiveness of airplane controls.
- Verify different citizen interaction scenarios.
- Confirm successful completion of various mission objectives.

Integration Testing:

Objective: Verify seamless interaction of multiple game components.
Approach: Test the combination of airplane controls, citizen interactions, and mission completion during simulated rescue scenarios, and other components.
Test Cases:

- Validate the integration of controls and environment reactions.
- Confirm data flow between different game modules.

System Testing:

Objective: Validate the entire game, flow, data flow etc.
Approach: Test the overall flow of different citizen rescue scenarios, user flow through user interface, data flow in the backend, storage of data, varying environmental conditions.
Test Cases:

- Conduct end-to-end testing of entire gameplay.
- Simulate diverse environmental conditions for realism.
- Test overall game stability under extended play, with an emphasis on performance.

User Acceptance Testing:

Objective: : Ensure the game meets user expectations and aligns with the intended user experience.
Approach: Allow actual end-users to play the game and provide feedback on overall satisfaction, functionality, and security aspects.
Test Cases:

- Conduct real-world gameplay scenarios.
- Collect user feedback on overall game experience.

3.8.3 Testing Techniques

Functional Testing:

Objective: Ensure core functionalities meet design specifications, including gameplay aspects.

Approach: Test user interactions, mission objectives, and aircraft movement mechanisms, with a focus on secure functionality.
Test Cases:

- Validate the accuracy of user flow in user interface.
- Confirm mission objectives align with game requirements.
- Test Gameplay.

Usability Testing:

Objective: Evaluate the preliminary UI for intuitiveness.
Approach: Conduct user feedback sessions on menu navigation, button placement, and overall user experience.
Test Cases:

- Assess ease of in game navigation.
- Evaluate the intuitiveness of button placement and working.
- Gather user feedback on overall experience.

Compatibility Testing:

Objective: Ensure smooth gameplay on devices with 3GB RAM and above, considering security implications.
Approach: Test on a range of Android devices, considering different screen sizes, resolutions, and RAM.
Test Cases:

- Validate game performance on devices with varying specifications.
- Ensure compatibility with different screen resolutions.
- Confirm responsiveness on devices with 3GB RAM and above.

Security Testing:

Objective: Identify and address potential security vulnerabilities and exploits.
Approach: Conduct security testing to uncover vulnerabilities in the game's code, data handling, and communication.
Test Cases:

- Perform penetration testing to identify potential weaknesses.
- Check for secure data storage.
- Validate user flow without encountering any bug which may end up misuse game backend data.

Performance Testing:

Objective: Assess the game's performance metrics.
Approach: Measure frame rates, loading times, and resource usage using performance testing tools.
Test Cases:

- Conduct stress testing to assess maximum load capabilities, including security stress scenarios.
- Measure frame rates under different in-game scenarios.
- Evaluate resource usage for potential optimizations.

3.8.4 Risk and Contingencies

- **Mitigation:** Priortizing UAT inorder to mitigate ususal scenrio like glitches.
- **Contingency:** Allocate additional time for critical testing phases if resource constraints arise.

3.9 Number of Iterations

The Software Development Plan is a detailed and strategic roadmap that explains about the iterative process for the effective development of our 3d game project.

This section will offer a comprehensive examination of the development plan iterations, clearly explaining the objectives and tasks of each iteration. If talk about each iteration individually, from the initial planning and requirement engineering to the finalization of the game's user interface effects and sound effects, plays a important role in determining the project's direction. The process involves a sequence order of organized iterations, with each one intended to gradually improve and enhance the project's scope, features, and functions in order to reach the goal of the game. The Software Development Plan guarantees the effective distribution of resources, timely achievement of milestones, and the provision of a great gaming experience through careful planning, analysis, and implementation.

3.9.1 iteration 1:

1. Plan the project by undertaking the time and resource estimations. Assign the positions based on team members skills.
2. Initiated requirement elicitation from several sources (existing game reviews, meeting and interviews) and assessed all the information available to best fit the project goals and technological capabilities.
3. Develop 2d joystick controls for both the helicopter and jet to reduce number of different direction buttons and integrate them as one .
4. Conduct asset hunting for both helicopter and jet models to be used in the game, ensuring high-quality and relevant to the requirement of the game.
5. Selected a dynamic code and modular system approach for the game. This will enable for easy performance improvements and reduce spaghetti coding, hence avoiding excessive nesting.
6. Begin the documentation phase of the project, detailing initial plans and advancements about the game.
7. Develop a risk management plan to identify potential issues and mitigation solutions for them.
8. Establish a communication plan utilizing platforms like Google Meet and Trello to guarantee smooth team cooperation.

3.9.2 iteration 2:

1. Further plan and refine the scope of the project.
2. Develop the backend of the garage system to facilitate helicopter and jet selection and their upgradation.
3. Implement the feature for players to select jet in the garage and link this selection with the gameplay via the backend.
4. Confirm the selection of the jet and equip it for the gameplay missions, ensuring seamless integration between the components.
5. Continue the documentation process, soliciting needs from diverse sources and refining them.
6. Continue the documentation process according to the criteria provided in the handbook.

7. Set up a version control system to track changes and collaborate efficiently.
8. Perform requirement analysis and prioritized high-impact requirements to ensure key features are developed early and clear game plan.
9. Conducted a feasibility analysis to examine the important features and timetable against available resources.

3.9.3 iteration 3:

1. Begin the design phase of the project, generating an initial software development plan.
2. Create architectural diagrams, use case diagrams, and flow charts to define system structure and interactions.
3. Utilize Blender to manipulate assets according to the requirement, including bursting of parts.
4. Develop collision detection mechanics for helicopter and jet, interacting with the surroundings.
5. Manipulate and alter assets in Blender for size optimization.

3.9.4 iteration 4:

1. Start planning the game environment, including terrain layouts and overall map design.
2. Gather assets and select low poly assets for the game.
3. Deployed assets for developing maps and made multiple prototypes, resolving errors through trial and error.
4. Create prototypes of the landscape and select the best appropriate solutions for the game environment.
5. Continue the documentation process, focusing on refining the project's scope, requirements, and design.
6. Perform asset hunting, focusing on optimized size assets, and manage the budget for asset purchase.

3.9.5 iteration 5:

1. Plan all game missions and the overall storyline, guaranteeing a consistent and engaging game storyline narrative.
2. Implement tutorial levels, targeting the first two levels for both helicopter and jet.
    - Create a helicopter tutorial involving flying through the rings.
    - Develop a jet tutorial centered on firing at targets and flying through the rings.
3. Develop a user guide for controlling aircrafts, integrating pictures to enhance player understanding.
4. Plan the game's complexity escalation, ensuring a fair challenge curve for players.
5. Design rewards for mission accomplishment to inspire and engage players.
6. Integrate armaments on the jet and extensively evaluate them for functionality and balance.
7. Compile and finish the Software Requirements Specification (SRS) document.
8. Create system sequence diagrams to detail interactions and system processes.
9. Develop a detailed test strategy to lead the testing phase and ensure complete evaluation of the game's features.

3.9.6 iteration 6:

1. Implement levels 3 and 4, which are the rescue missions , modifying mission plans and assets as appropriate.
2. Integrate in-game elements such as safe/waiting zones and rescue zones inside the map and game environment.
3. Set mission timers and plan bonus prizes for accomplishing tasks.
4. Ensure smooth interaction between the helicopter, the environment and obstacles during missions.
5. Write dynamic scripts for helicopter capacity and safe zone capacity, adding complexity to the gameplay.

6. Implement a spawning mechanism to generate citizens in the rescue regions, and fix any related errors.
7. Add a checker to verify if the player has rescued all citizens.
8. Additionally Equip the jet with explosives and apply the physics for bomb falling.
9. Equip enemy units with weapons for a balanced battle experience.
10. Conduct playtesting for rescue missions to gain feedback and make required adjustments

3.9.7 iteration 7:

1. Plan and construct combat levels 5 and 6, focusing on fierce aerial fights and engaging gameplay mechanisms.
2. Gather assets such sound effects, background music, and impact sounds to enhance the immersive experience for the combat levels.
3. Design enemy, which comprise of aerial drones and ground enemy, and organize their behavior and attacks.
4. Implement movement and shooting mechanics for airborne drones and ground enemy, employing techniques like nav mesh and scripting for accurate control.
5. Integrate power-ups into the garage system, allowing players to purchase and utilize them for strategic advantages during gaming.
6. Write dynamic scripts to control the behavior of drones, modifying aim accuracy, attack patterns, and movement characteristics.
7. Create 2D body of the jet to signal approaching attacks and differentiate between hits and misses on the player's aircraft.
8. Plan and execute a process for unlocking stuff in the garage and purchasing new equipment using in-game reward points.

3.9.8 iteration 8:

1. Integrate rescue and combat levels to progress the storyline, ending in the final mission that mixes together components from both gameplay types.
2. Gather and evaluate sound effects, using user feedback to ensure immersive and impactful audio experiences.

3. Plan the aesthetic design of the user interface, focusing on clarity, usability, and thematic continuity with the game's scenario.

4. Gather UI components like buttons, menus, and HUD elements, ensuring they align with the chosen design and allow easy navigation for players.

5. Conduct end-user testing to gain feedback on gameplay mechanics, level design, and overall user experience, iterating based on user input to improve satisfaction.

6. Continue the documentation process, improving and updating project documentation to reflect changes and advances achieved during development.

3.9.9 iteration 9:

1. Finalize the user in game routes for playing missions, pausing gameplay, exiting missions, and quitting the game, providing smooth navigation and intuitive controls.

2. Complete the reward system for the user for completing the missions, including the integration of power-ups available for purchase in the garage system, and link the garage system with the reward system for seamless credit deduction.

3. Complete the creation of the garage system, ensuring all features are incorporated and tested for functionality and usability.

4. Finalize the user interface design, including menus, HUD components, and in-game prompts, combining the storyline and visual assets such as photographs to enhance the narrative experience. Integrate all sound effects and music into the game to support gameplay and boost immersion.

5. Conduct final rounds of documentation, including integration testing to ensure all components work together smoothly. Evaluate the game against project objectives and user feedback to discover any remaining requirement or opportunities for improvement before release.

6. Implement any final optimizations of game size or bug fixes based on user feedback and testing results to guarantee the game satisfies quality standards and provides a nice experience for players.

7. Updating our Git repository and uploading code to have complete history of project development.

Chapter 4

Implementation

This chapter explains the full process of building our game project. We describe the agile methodology and systematic approach employed, then proceed to an in-depth look at requirement elicitation, prioritising, verification and validation. The design phase comprises the construction of UML diagrams and architectural designs that drove development. We present a thorough evaluation procedure, embracing user feedback for continual improvement. The employed approaches section emphasises tactics and instruments for efficient development and project management. Finally, we assess the project's success using qualitative evaluation measures, concentrating on user satisfaction, maintainability, and optimization. This chapter presents a clear account of the diligent preparation, execution, and evaluation that led to the production of our game.

4.1 Adopted Methodology for Each Iteration

4.1.1 Iteration 1

1. **Sprint Planning:** In the first iteration, the sprint planning meeting occurred with the product owner, the scrum development team, and the stakeholder to establish the project's initial scope and objectives. A number of important tasks were noted, such as project planning, requirement gathering, developing joystick controls, asset searching, implementing dynamic code, initiating documentation, organising risk management, and creating communication plans. For efficient work distribution and goal alignment, the team estimated the amount of time and resources needed and assigned tasks based on individual competence.

2. **Sprint Backlog:** The sprint backlog for this iteration contained the following duties such as

    - Perform time and resource estimations.
    - Allocate positions based on team members' competence.

- Initiate requirement elicitation from diverse sources.
- Analyze acquired information in accordance with project objectives and technological capabilities.
- Develop joystick controls for the helicopter and jet.
- Perform asset searching for airplane models.
- Implement a dynamic code and modular system.
- Kick off the documentation process.
- Build a risk management plan.
- Set up a communication plan using Google Meet and Trello.

3. **Sprint Meetings:** Daily stand-up meetings were realistically not feasible to hold alternatively online meeting were scheduled amongst the team and product owner to discuss progress, address barriers, and alter plans as appropriate. These sessions provided continuing communication and coordination amongst the team members, assuring that everyone remained on track and any concerns were promptly resolved.

4. **Sprint Review:** At the completion of the iteration, a sprint review meeting was held. The team demonstrated the created joystick controls and displayed the selected airplane assets. The dynamic code and modular system implementation were assessed, and preliminary documentation was shared. The product owner and stakeholder offered input, praising the smooth gameplay mechanics and the quality of the selected assets. Recommendations were given to improve the risk management plan and modify the communication strategy.

5. **Sprint Retrospective:** In the sprint retrospective meeting, the team evaluated on the iteration's methods and results. The following topics were discussed:

    - What Went Well: Effective role assignment, easy joystick control development, and successful asset search.
    - Areas for Improvement: Need for more tailored risk management methods and greater usage of communication technologies.

4.1.2 Iteration 2

1. **Sprint Planning:** The sprint planning session for the second iteration focused on refining the project scope, progressing the backend development, and guaranteeing seamless integration between the garage system and gameplay. The team, including the product owner, scrum development team, and stakeholder, examined suggestions from the prior iteration and prioritized tasks.

The planning also included establishing a version control system, performing requirement analysis, and executing a feasibility assessment to align the planned features with the resources at hand and schedule.

2. **Sprint Backlog:** The sprint backlog for the present iteration contained the following assignments:

   - Further evaluate and refine the scope of the project.
   - Construct the backend of the garage system for aircraft selection and administration.
   - Implement feature for selecting aircraft in the garage and connecting this choice with gameplay.
   - Confirm and equip the selected aircraft for gameplay missions.
   - Continue the documentation process, eliciting and revising requirements from multiple sources and following the provided handbook.
   - Build up a version control system for effective collaboration.
   - Conduct requirement analysis and prioritize high-impact requirements.
   - Perform a feasibility study to analyze functionalities and timetable against resources.

3. **Sprint Meetings:** Online meetings were scheduled to guarantee constant communication and to solve any obstacles. These sessions enabled the team to remain aligned with the sprint objectives and facilitated promptly changes to the plan as required.

4. **Sprint Review:** Following the sprint review meeting, the team demonstrated the developed backend for the garage system and the executed aircraft selection features. The integration between the garage system and gameplay was demonstrated, displaying the seamless integration of selected aircraft into missions. Documentation development was evaluated, and the requirement analysis and feasibility study results were provided. The product owner and stakeholder provided constructive criticism on the backend development and the effective version control mechanism.

5. **Sprint Retrospective:** During the sprint retrospective meeting, the team addressed the following highlights:

   - What Went Well: Successful backend development, effortless integration of aircraft selection, and an effective version control setup.
   - Areas for Improvement: Improved and comprehensive requirement analysis and improved prioritization of essential features was advised.

4.1.3 Iteration 3

1. **Sprint Planning:** During the planning phase for the third iteration, the team concentrated on the design element of the project. This included defining a software development plan along with creating architecture diagrams, use case diagrams, and flow charts. The team also aimed to leverage Blender for integrating collision mechanics and size optimization of assets, along with creating collision detection for helicopters and jets.

2. **Sprint Backlog:** The sprint backlog for this iteration included:

   - Initiate the design phase and build an initial software development plan.
   - Construct architecture diagrams, use case diagrams, and flow charts.
   - Utilize Blender to implement collision physics for assets, including exploding into bits.
   - Manipulate and alter assets in Blender for size optimization.
   - Create collision detection and mechanics for helicopters and jets that engage with the surroundings.

3. **Sprint Meetings:** Frequent Google Meet sessions occurred during the iteration to discuss progress, manage difficulties, and alter plans as required. Such virtual meetings promoted steady team communication and guaranteed alignment with sprint goals.

4. **Sprint Review:** At the sprint review meeting, the team put forward:

   - Architectural diagrams, use case diagrams, and flow charts illustrating the system structure and interactions.
   - Collision detection and mechanics for helicopters and jets interacting with the surroundings.
   - The demonstrations of collision mechanics applied in Blender, involving assets bursting into bits.
   - Updated and optimized assets for size efficiency.

5. **Sprint Retrospective:** During the sprint retrospective, the team discussed:

   - What Went Well: Successful demonstration of design phase in documentation, successful utilization of Blender for asset manipulation, and development of collision detection mechanics.
   - Areas for Improvement: Refine the architectural diagram and address minor flaws. Research more adaptable assets for customization.

4.1.4 Iteration 4

1. **Sprint Planning:** During this iteration, the team intended to concentrate on the construction of the game environment. This involves designing city layouts, sourcing assets, and deciding the overall aesthetics of the game. The team also intended to refine the project's scope, requirements, and design.

2. **Sprint Backlog:**

   - Environment Planning: Construct city layouts and entirety of the map structure.
   - Asset Collection: Collect and acquire UI assets for the game, focusing on optimum size while managing the budget.
   - Asset Testing: Construct maps utilizing the collected assets and execute bug fixes via trial and error.
   - Terrain Prototyping: Develop and evaluate terrain prototypes to determine the best possibilities for the game environment.
   - Documentation: Continue revising the project's scope, requirements, and project documents.

3. **Sprint Meetings:** Multiple virtual meetings were carried out via Google Meet during the iteration to go over the progress, address concerns, and make needed revisions. These meetings promoted ongoing communication and problem-solving amongst team members.

4. **Sprint Review:** At the sprint review, the team displayed the core game environment layouts, the acquired assets, and the prototypes of the terrain. Feedback was received from the product owner and the stakeholder, underlining the need for additional improvement in certain areas.

5. **Sprint Retrospective:** The retrospective revealed significant areas for improvement:

   - What Went Well: Effective asset collecting and selection, productive virtual sessions.
   - What Could Be Improved: More comprehensive initial testing of assets to minimize trial and error. Acquiring more size efficient assets to lower the size of the map.

4.1.5 Iteration 5

1. **Sprint Planning:** Within this iteration, the team worked on organizing the game's objectives and plot, integrating tutorial levels, creating a user guide,

and resolving the game's difficulty progression and rewards. Furthermore, the integration and testing of jet weapons, finalizing the SRS document, the creation of system sequence diagrams, and the development of a test plan were emphasized.

2. **Sprint Backlog:**

   - Mission and Storyline Planning: Plan all game missions and the overall plot to guarantee a cohesive and interesting narrative.
   - Tutorial Levels: Build tutorial levels for both helicopters and jets.
   - Helicopter Tutorial: Create a tutorial involving traversing through rings.
   - Jet Tutorial: Develop an tutorial focused on shooting at targets.
   - User Guide Development: Create a user guide to operating aircraft, integrating pictures to enhance player grasp.
   - Difficulty Progression: Plan the game's difficulty progression, guaranteeing a fair challenge curve for players.
   - Reward Design: Design rewards for task accomplishment to motivate and captivate the players.
   - Weapons Integration: Integrate Weapons on the jet and extensively test them for functionality and stability.
   - SRS Finalization: Compile and finish the Software Requirements Specification (SRS) document.
   - System Sequence Diagrams: Create system sequence diagrams to detail interactions and system operations.
   - Test Plan Development: Develop a detailed test plan to lead the testing phase and ensure complete evaluation of the game's functionalities.

3. **Sprint Meetings:** Several virtual meetings using Google Meet were conducted to manage the planning of missions, production of tutorial levels, and compilation of the user guide. These meetings ensured that team members had common ground and could handle any difficulties collaboratively.

4. **Sprint Review:** Throughout the sprint review, the team displayed the intended missions and plot, the implemented tutorial levels, plus the basic user guide. Feedback was obtained from the product owner and stakeholder on the narrative coherence, tutorial efficiency, and user guide clarity.

5. **Sprint Retrospective:** The retrospective emphasized the following points:

   - What Went Well: Effective planning of missions and storyline, effective implementation of tutorial levels, and rich user guide development.

- What Could Be Improved: More efficient integration of jet weapons and update the repository for improved version control.

4.1.6 Iteration 6

1. **Sprint Planning:** In this iteration, the team concentrated on introducing and improving rescue missions, guaranteeing smooth gameplay mechanics and interactions. Key work included implementation of levels 3 and 4, incorporating mission features, and improving gameplay complexity.

2. **Sprint Backlog:**

   - Level Implementation: Develop and improve levels 3 and 4, focused on rescue missions.
   - Feature Integration: Add safe/waiting zones and rescue zones within the game world.
   - Mission Timers and Bonuses: Set timers on missions and plan bonus prizes for accomplishing goals.
   - Gameplay Interaction: Assure seamless interaction between the helicopter and the environment.
   - Spawning functionality: Implement a functionality to spawn citizens in rescue regions and fix related bugs.
   - Rescue Checker: Integrate a checker to confirm if all citizens have been rescued.
   - Jet Bombing: Equip jets with bombs and add mechanics for bomb falling.
   - Dynamic Scripts: Develop scripts to alter helicopter and safe zone capacity dynamically, adding gameplay complexity.
   - Enemy Weapons: Equip enemies with weapons for fair combat.
   - Playtesting: Conduct playtesting for rescue missions to gain feedback and make improvements

3. **Sprint Meetings:** Frequent virtual meetings were held using Google Meet to go over the status of mission implementation, feature integration, and script development. These meetings promoted continual team collaboration and figuring out solutions.

4. **Sprint Review:** The sprint review featured the newly added rescue missions, integrated features, and dynamic scripts. Feedback was obtained from the product owner and the stakeholder, emphasizing on gameplay interaction and mission complexity balance.

5. **Sprint Retrospective:** The retrospective emphasized the following points:

    - What Went Well: Successful implementation of rescue missions and integration of dynamic scripts.
    - What Could Be Improved: Initial bug fixing and testing methods could be more effective. Optimize the task timer and bonuses by investigating the gameplay and improving user feedback.

4.1.7 Iteration 7

1. **Sprint Planning:** During the iteration, the team worked on generating striking combat levels, strengthening gameplay mechanics, and integrating audio-visual elements to increase the player's immersive experience. Furthermore, methods for aircraft unlocking and power-ups were set up.

2. **Sprint Backlog:**

    - Combat Levels: Plan and implement combat levels 5 and 6 with an emphasis on aerial combat and stimulating gameplay mechanisms.
    - Enemy Design: Design enemy components, including aerial drones and ground rivals and arrange their behavior and strikes.
    - Enemy Mechanics: Implement movement and shooting mechanics for airborne drones utilizing nav mesh and scripting methods.
    - Asset Gathering: Collect sound effects, background music, and impact sounds to enhance the game's feelings.
    - Power-ups Integration: Integrate power-ups into the garage system, enabling players to purchase and utilize them strategically.
    - Item Unlocking: Plan and develop a process for unlocking items in the garage and acquiring new equipment with in-game currency.
    - Visual Effects: Create 2D visual effects to show approaching attacks and distinguish between hits and misses.
    - Dynamic Scripting: Write scripts to alter drone behavior, including aim precision, patterns of attack, and movement characteristics.

3. **Sprint Meetings:** Online meetings were organized via Google Meet to supervise the development of battle levels, enemy mechanics, and integration of new assets. These meetings ensured constant alignment and resolving issues by the team members.

4. **Sprint Review:** Following the sprint review, the team showed the freshly developed combat levels, enemy units, and integrated sound effects. Feedback

was obtained from the product owner and stakeholder, notably on the effectiveness of enemy behaviors and the immersive quality of audio-visual features.

5. **Sprint Retrospective:** The retrospective emphasized these points:

   - What Went Well: Successful implementation of combat levels and dynamic enemy behaviors.
   - What Could Be Improved: Bug fixing throughout gameplay, enemy behaviour fixes and better integration.

4.1.8 Iteration 8

1. **Sprint Planning:** During this iteration, the team concentrated on integrating rescue and combat levels to develop the storyline, collecting and testing sound effects, planning and implementing the user interface, and conducting end-user testing.

2. **Sprint Backlog:**

   - Level Integration: Integrate rescue and combat levels to put together the storyline, ending in a final mission that integrates features from both gameplay styles.
   - UI Design: Plan the aesthetic design of the user interface, emphasizing clarity, usability, and thematic continuity with the game's setting.
   - UI Components: Assemble UI components, including buttons, menus, and HUD elements, guaranteeing they correspond with the chosen design and allow straightforward navigation for players.
   - End-User Testing: Conduct end-user testing to gain feedback on gameplay mechanics, level design, and overall user experience, iterating based on user input to enhance experience.
   - Sound Effects: Gather and evaluate sound effects, utilizing user feedback to make sure of an immersive and compelling audio experiences.
   - Documentation: Continue the documentation process, enhancing and revising project documentation according to the changes and progress accomplished during development.

3. **Sprint Meetings:** Virtual meetings using Google Meet were held to organize the integration of levels, collect and test sound effects, and develop the UI design. These meetings promoted constant cooperation and problem-solving among individuals on the team.

4. **Sprint Review:** During the sprint review, the team exhibited the integrated rescue and combat levels, verified sound effects, and early UI designs. Feedback was obtained from the product owner and stakeholder on the effectiveness of the level integration, auditory experiences, and UI aesthetics

5. **Sprint Retrospective:** The retrospective emphasized these specific points:

    - What Went Well: Successful integration of rescue and combat stages, and effective user feedback integration for sound effects.
    - What Could Be Improved: Optimizing UI assets size and integration plan. Modify and adhere to the test plan.

4.1.9 Iteration 9

1. **Sprint Planning:** In this iteration, the team worked on completing user paths, finalizing the reward system, garage system development, user interface design, documentation, and final optimizations. These efforts sought to assure seamless gaming, effortless integration of features, and overall quality enhancement prior release.

2. **Sprint Backlog:**

    - User Pathways Finalization: Finalize user pathways for playing missions, pausing gameplay, ending missions, and quitting the game to provide clear controls and swift navigation.
    - Reward System Completion: Finish the reward system for completing missions, integrating power-ups accessible for purchase in the garage system, and linking it with the reward system for smooth credit deduction.
    - Garage System creation: Complete the creation of the garage system, ensuring all features are integrated and extensively tested for functioning and usability.
    - User Interface Design: Conclude the user interface design, including menus, HUD elements, and in-game prompts, including storyline and visual information such as photos for an immersive narrative experience.
    - Sound Effects and Music Integration: Integrate all sound effects and music into the game to compliment gameplay and boost immersion.
    - Documentation and Integration Testing: Conduct final rounds of documentation, including integration testing, to ensure all components function together seamlessly and evaluate the game against project objectives and user input.

- Optimizations and Bug Fixes: Implement any necessary optimizations or bug fixes based on user feedback and testing results to guarantee that the game satisfies quality standards and delivers an enjoyable experience for players.
- Git Repository Update: Update the Git repository and upload code for optimal version control and collaboration among team members.
- User Testing: Conduct final rounds of user testing to obtain feedback on the completed game components and identify any residual issues or areas for improvement.
- Final Review with Stakeholders: Hold a final review session with stakeholders to demonstrate the completed game and receive their input before release.

3. **Sprint Meetings:** Online meetings using Google Meet were carried out numerous times during the iteration to evaluate progress, resolve any issues or issues, and maintain harmony among team members about the finalization of game components.

4. **Sprint Review:** During the sprint review, the team showcased the finalized user pathways, finished reward system, garage system development, user interface design, sound effects, music integration, documentation, and optimizations. Feedback was obtained from the product owner and stakeholder on the overall gaming experience and any outstanding issues or areas for enhancement.

4.2 Requirements of Iterations:

In the following section, we describe the software requirements for each iteration of the development process, including numerous activities of requirement engineering to design, version control, documentation, and communication. Every iteration is guided by a disciplined approach to software development, guaranteeing that project goals are met promptly and effectively. The requirements stated here serve as recommendations for the development team, offering a roadmap for developing features, maintaining project assets, and fostering collaboration. With thoughtful preparation and execution of these objectives, the development team attempts to provide a high-quality software solution that meets stakeholder expectations and user needs.

4.2.1 Software Requirements for Iterations:

Below are the general requirements for the iterations.

1. **Requirement Engineering Process:**

    - Elicit requirements from multiple sources including existent game reviews, interviews with stakeholders, and talks with the development team.
    - Analyze and improve requirements to ensure clarity, accuracy, and consistency.
    - Verify and validate requirements through reviews, walkthroughs, and talks with stakeholders to ensure alignment with project goals.
    - Conduct a feasibility analysis to examine the technical and resource restrictions against requested features and schedule, ensuring practical planning and decision-making.
    - Prioritize requirements by considering their significance, urgency, and feasibility to steer development efforts properly.

2. **Design Activities:**

    - Develop architectural diagrams demonstrating the system's overall structure, component interconnections, and data flow to assist implementation activities.
    - Write fully dressed use cases and create use case diagrams to capture functional requirements and system behavior from the user's point of view.
    - Design system sequence diagrams to demonstrate the interactions between system components and external actors, detailing the sequence of events throughout system operation.
    - Design flowcharts that illustrate the flow of control and data inside specific components or processes, improving in understanding and communication of system logic.

3. **Version Control:** Establish and manage a version control system, Git, to keep track of changes and cooperate effectively.

4. **Documentation:**

    - Start and maintain comprehensive project documentation including Software Requirements Specification (SRS), design documents, test plans, and user manuals.
    - Compile and complete the SRS document, combining system sequence diagrams, fully dressed use cases, and other related artifacts to offer an in-depth description of system requirements.

- Write a test plan describing the testing approach, test scenarios, and acceptance criteria to guarantee thorough examination of system functionality.
- Document the design phase, implementation details, and the outcomes to allow maintenance and future enhancements of the system.

5. **Communication and Collaboration:** Develop a communication plan utilizing platforms like Google Meet and Trello to enable effortless collaboration among teams. Update the Git repository often, uploading code and essential files to ensure version control and cooperation.

6. **Risk Management:** Create and maintain a risk management plan that outlines potential risks and mitigation techniques throughout the lifespan of the project.

7. **Asset Management:** Do asset hunting for quality assets, focusing on optimal size assets, while overseeing the budget for asset acquisition. Test assets by developing maps and prototypes, addressing bugs through trial and error.

8. **Testing and Feedback:** Conduct end-user testing to gain input on gaming mechanics, level design, and general user experience. Iterate based on consumer input to improve experience.

4.3 Design, Code and Testing Results of each Iteration:

4.3.1 For Iteration 1, 2, 3:

Design

Design efforts were undertaken in Iterations 1, 2, and 3 to translate project requirements into actionable models directing development. Using technologies like PlantUML and MS Visio, architectural diagrams, use case diagrams, and flowcharts were built to illustrate system structure and interactions. The Component Based Architecture serves as the backbone, ensuring modularity and scalability. System sequence diagrams were developed to describe the general flow, while Blender was used to implement collision mechanics for objects, optimizing their size and guaranteeing smooth interaction with the environment. The design documents, produced using UML as the modeling language, provided a thorough framework for development, aligned with the project's objectives and technical needs.

Testing Results During Iteration 1,2,3, user feedback was collected through surveys and casual playtesting sessions. The feedback revealed multiple areas for improvement, including ideas for boosting the simplicity of the joystick controls, opti-

mizing asset choice for better aesthetic value, and refining the overall user interface for simpler navigation. Minor issues were detected in the joystick movement controls for both the helicopter and jet. These flaws mostly involve sensitivity adjustments and calibration differences, leading to unexpected erratic behavior during gameplay. To solve these concerns, improvements were implemented to the control algorithms to ensure smoother and more responsive mobility.

4.3.2 For Iteration 4, 5, 6:

Design

Implementation throughout these iterations was prompted by the enhanced quality of existing UML diagrams and architecture designs. In this instance, the implementation process was directed by the UML diagrams and architecture sketches, making sure that the development activities stayed aligned with the intended system structure. Use cases and use case diagrams were evaluated and evolved to better correspond with the developing demands of the project. The emphasis was on transforming these design artifacts into concrete development tasks, driving the execution of game features and mechanisms.

Testing Results

- Following user feedback testing, minor bugs were detected in the jet tutorial level which were promptly addressed.
- A severe problem was detected in the jet's mobility and firing accuracy, which was addressed and repaired to improve gameplay.
- Bugs were detected in the spawning mechanism, particularly with the generation of citizens in rescue regions, and they were corrected to ensure proper functionality.

4.3.3 For Iteration 7, 8, 9:

Design

Throughout iterations 7, 8, and 9, the development plan was carefully followed in the context of Scrum methodology. The UML diagrams and designs for architecture generated in earlier rounds were implemented to support the development of gaming features and data management. This time concentrated mainly on the application and refinement of these plans thus moving the project closer to conclusion. Implementation of dynamic scripts for enemy behavior and merging 2D visual effects, which were intended and constructed based on the existing UML diagrams and architectural designs.

Testing Results

- **Iteration 7:** Bugs were detected in enemy behavior, including concerns with its mobility and fire mechanics. They were addressed by adjustments in the dynamic scripts and modifying parameters for more realistic and demanding enemy actions. A major bug where opponent health was not dropping correctly was detected and fixed, assuring balanced combat mechanics.
- **Iteration 8:** Integration testing was executed to check that rescue and combat levels were properly merged and that the user interface components operated correctly. Feedback from user testing prompted additional refining of the user interface and gameplay mechanics, boosting overall satisfaction among players.
- **Iteration 9:** Final iterations of integration testing were undertaken to make sure all components operated together properly. Stakeholder reviews were undertaken to enhance and conclude the documentation and overall gameplay. User feedback proved crucial in identifying minor bugs and areas for improvement, resulting to final optimizations and bug fixes to fulfill quality requirements and create a worthwhile playing experience.

4.4 Utilized Techniques:

Within this section, we detail the different techniques we applied throughout the development of our gaming project. These techniques span a range of methodologies and activities, including project management, requirement engineering, design strategies, programming practices, testing procedures. By employing these strategies, we secured a disciplined, and effective development process, leading up to a high-quality end product.

4.4.1 Agile Methodology (Scrum)

- **Sprint Planning:** Planning and organizing the tasks that need to be done in every sprint.
- **Sprint Backlog:** Identifying and listing tasks and prioritizing them for every sprint.
- **Sprint Meetings:** Frequent meetings to talk about progress and plan next actions.
- **Sprint Review and Retrospective:** Evaluating the finished work and finding areas for improvement.

4.4.2 Requirement Engineering

- **Elicitation:** Collecting requirements from game reviews, interviews, market surveys, and stakeholder involvement.
- **Analysis:** Aligning obtained information with project goals and technological capabilities.
- **Verification and Validation:** Ensuring requirements are complete, accurate, realistic, and verifiable.
- **Prioritization:** Concentrating on high-impact and critical requirements.
- **Feasibility Study:** Evaluating planned functionalities against resources and given deadlines.

4.4.3 Design Techniques

- **Modular Architecture:** Developing a component based and modular architecture to enable for quick changes and maintenance.
- **UML Diagrams:** Designing architectural diagrams, use case diagrams, system sequence diagrams, and flowcharts for illustrating system interactions.
- **Software Development Plan:** Describing the phases and objectives for development.

4.4.4 Programming Techniques

- **Dynamic Coding:** Writing flexible code to reduce unnecessary nesting and increase performance.
- **Objected Oriented and Data structures:** Applying object oriented principles and data structures throughout the project to implement a component ba
- **Scripts for Game Mechanics:** Writing dynamic scripts for features such as drone motions, attack patterns, and mission-specific interactions.

4.4.5 Testing Techniques:

- **Integration Testing:** Making sure new features integrate properly into current gameplay.
- **User Testing:** Collecting feedback from users to detect and fix bugs.

- **Stakeholder Reviews:** Incorporating suggestions from stakeholders to better documentation and gameplay.

4.5 Evaluation Metrics

In this subsection, we detail the qualitative measures utilized to evaluate the **performance and quality** of our game development project. These metrics provide information regarding consumer satisfaction, product completeness, and overall project performance.

- **User Feedback:** We consistently gathered feedback from users during the development process. This feedback was used to tweak game controls to enhance gameplay mechanics. Users reported that the controls were straightforward and the gameplay was engaging, indicating a positive response.
- **Stakeholder Satisfaction:** Satisfaction of stakeholders, including developers, project managers, and clients, was monitored through frequent review meetings and progress reports. Stakeholders showed satisfaction with the timely delivery of project objectives and the standard of implemented features.
- **Bug Reports and Fixes:** Within the testing rounds, various bugs were detected and tracked. Particularly, major issues relating to jet movement and shooting precision were resolved. The spawning feature also had issues that were eventually fixed, assuring robust and reliable gaming.
- **Documentation Quality:** Comprehensive and up-to-date documentation was produced throughout the project. This includes requirement specifications, design documentation, and user guides. The documentation process was frequently revised to reflect changes and maintain accuracy.
- **Feature Completeness:** All intended features were implemented as expected. This included the aircraft selection mechanism, collision physics, and many gameplay levels. Frequent testing guaranteed that these features worked effectively and fulfilled the objectives of the project.
- **Gameplay Smoothness:** The fluidity and responsiveness of gameplay were significant emphasis areas. Joystick controls for aircrafts were created and adjusted based on user feedback, resulting in a seamless and engaging gameplay experience.
- **Maintainability and Flexibility:** The code is designed to be easily maintainable and changeable through dynamic component-based coding and script-

ing. This strategy allowed for fast performance tweaks and minimized unnecessary nesting, maintaining a clean and modular codebase.

- **Optimization for Low-End Devices:** Considerable efforts were taken to optimize the game for improved performance on low-end mobile phones. This involved lowering the game size to under 100MB and optimizing assets and mechanics to ensure that it runs smoothly even on devices with inferior specifications.

Chapter 5

Results and Discussions

5.1 Introduction

In this chapter of "Results and Discussion", we will thoroughly examine the outcomes of each development iteration, by carefully assessing the expected results in comparison to the actual results accomplishments. We conduct a thorough assessment, to evaluate the progress made in achieving project goals of the game. This assessment includes identifying the accomplishments, problems, and areas that need improvement. By comparing the expected outcomes with the actual achievements, through this we may obtain useful knowledge about the efficiency of our development process and the overall progress of the project. This chapter offers a thorough analysis of the project's progression, establishing a basis for well-informed decision-making and future development efforts.

5.2 Result and Analysis

This section evaluates the expected and actual outcomes of the project iterations, examining these outcomes in order to understand the advancement.

5.2.1 Iteration 1

Expected Results:

- Implementation of 2d joystick controls for aircrafts is necessary.
- Requirements must be elicited and reviewed.
- A dynamic and modular architecture based system must be built to allow for easy changes.
- It is necessary to choose airplane models that have the ideal size and high quality.

- For virtual communication, employing platforms like Google Meet and Trello must be put in place.

Achieved Results:

- 2d Joystick controller for helicopter and jet were created and tested.
- Requirements were extracted from existing game reviews and interviews, coinciding with project aims.
- A dynamic and modular architecture was built, allowing for easy performance changes and preventing spaghetti coding.
- Quality airplane models were picked and optimized for the game.
- A communication plan was built utilizing Google Meet and Trello, facilitating smooth team cooperation

Comparison and Analysis:

- In comparison outcomes, where mostly outcomes were reached, with successful creation and integration of 2d joystick controls and the modular system.
- Minor improvements were made to joystick control mechanics based on first testing input.
- The requirement elicitation approach discovered, new user revealed which incorporated within the project scope.

5.2.2 Iteration 2

Expected Results:

- The scope of the game project must be further refined and planned.
- Backend for aircraft selection must be developed and also the whole garage system UI and its required elements must be developed.
- Functionality for players where it can pick aircraft from garage and play it in gameplay, this functionality must be added.
- A version control system must be set up in order to track changes.
- Requirements must be prioritized and examined for possible outcomes.

Achieved Results:

- The project scope was successfully refined, and comprehensive blueprints were prepared.
- The backend for the garage system was constructed, allowing player to do aircraft selection and management.
- Functionality for aircraft selection in the garage and its integration with gameplay was successfully implemented.
- A version control system was built, enabling efficient tracking of changes and collaboration.
- Requirements were prioritized based on their impact and feasibility were conducted.

Comparison and Analysis:

- The predicted results were successfully accomplished, boosting the game structure and progress.
- After the revised project scope and careful planning, it gave a clear roadmap for future iterations.
- The integration of aircraft selection with gameplay provided a crucial feature to the game, boosting user experience and user flow from garage to gameplay.
- The version control system promoted smooth collaboration among team members, ensuring changes were successfully managed and documented.
- The feasibility assessment and finding important requirement ensured that high-impact features will be addressed early in the project.

5.2.3 Iteration 3

Expected Results:

- An initial software development plan for the game must be defined.
- Important diagrams like Architectural diagrams, use case diagrams, and flowcharts illustrating system structure and interactions, must be completed.
- Collision mechanics for jet and helicopter must be finalized, including there behavior when exploding into parts, these functionality must be implemented.

- Models must be modified and optimized for size using Blender.
- Collision detection system and mechanics for helicopter and jet interacting with the environment must be created.

Achieved Results:

- A thorough software development strategy was developed, guiding the project's progression.
- Important diagrams like Architectural designs, use case diagrams, and flowcharts were successfully completed, providing explicit system structure and interactions.
- Collision mechanisms for model were implemented successfully, allowing them to burst into bits upon collision.
- Models were optimized for size using Blender, boosting game performance and small size.
- Collision detection and mechanics system for helicopter and jet interacting with the environment were implemented, boosting gameplay realism.

Comparison and Analysis:

- All expected results were achieved successfully, with the design and development phase significantly advancing.
- The compilation of precise diagrams and flowcharts offered a solid platform for further advancement.
- Implementing and tuning collision mechanics increased the game's visual and game look.
- The successful creation of collision detection and physics provided depth to the gameplay, making interactions more realistic and interesting for the player.

5.2.4 Iteration 4

Expected Results:

- Planning the game environment, comprising city, dam, terrain, must be started.
- Assets and aesthetics for the game must be finalized and gathered.

- Main focus on assets will their optimized size assets, managing the budget for procurement.
- Environment assets must be tested by developing different maps, addressing errors through trial and error.
- Prototypes of the whole environment must be constructed, with the best-suited alternatives picked for the game environment.

Achieved Results:

- The planning of the game environment, including city layouts and overall map design, began and progressed nicely.
- Aesthetic of the elements and required assets for the game were gathered and selected, ensuring a unified themed visual style.
- Successfully obtained assets having optimal size assets, budget friendly.
- Assets were evaluated during map building, identifying and fixing the issues faced.
- Multiple Terrain prototypes were constructed, and the most suitable possibilities were picked for the game.

Comparison and Analysis:

- The expected results were met successfully, with planning and gathering of assets proceeding as intended.
- Minor bugs were detected from the review of the third iteration, particularly in collision detection and mechanics, which were rectified.
- New defects were detected during this iteration, such as concerns with asset scaling and texture mapping, which were also rectified.
- Encountered multiple bugs in the collision mesh of the asset during testing of map assets, bugs were carefully handled and successfully solved.
- The expected prototype look of the environment and the obtained results were not totally matching with the game aesthetics as planned, consequently hunt for better fitting assets was done again.

5.2.5 Iteration 5

Expected Results:

- The development of all game missions and the overarching storyline must be completed.
- Tutorial stages for helicopter and jet must be built, focusing on basic controls, obstacles and mechanics.
- A 2d user guide must given to teach the user about controlling of aircrafts, with integration of pictures for improved player understanding.
- The increase in game's difficulty must be designed, creating a fair challenge curve for players.
- Rewards for mission accomplishment must be directed to motivate and engage players.

Achieved Results:

- Comprehensive planning of all game missions and the overarching storyline was completed, this feature resulted in increasing game's experience and engaging narrative.
- Tutorial stages for helicopters and jets were developed, with the helicopter Tutorial including traveling through rings and the jet tutorial focusing on firing at targets.
- A complete user guide for controlling aircrafts was prepared successfully, including assisting graphics to assist player understanding.
- The increase in game's level difficulty was successfully planned, creating a balanced challenge curve that steadily develops in complexity.
- Reward on mission completion were set up adding motivational features to engage players gameplay.

Comparison and Analysis:

- The expected results were mostly attained, with the planned missions and storyline of the game being well-executed.
- The tutorial levels developed worked as anticipated, but upon feedback and testing of the level, few minor errors were detected in the jet tutorial, which were carefully addressed.

- It was believed that the user tutorial guide and difficulty progression would function smoothly. However, on feedback revealed a several issues in the jet's mobility and firing accuracy, which was subsequently corrected.
- Additional errors relating to the implementation of the tutorial level and reward system were uncovered, they were carefully handled which resulted in ensuring a smoother gameplay experience.
- This iteration's focus was on balancing difficulty and developing enticing reward was successfully satisfied, coinciding with the project's aims for player motivation and progression.

5.2.6 Iteration 6

Expected Results:

- In game Levels 3 and 4, which are the rescue missions, must be implemented, improving mission plans and addition of assets appropriate with level setting.
- In-game features such as safe/waiting zones and rescue zones must be interconnected with the game 2d map and game environment.
- Mission with limited timer and bonus prizes for accomplishing mission goal must be set.
- Smooth interaction between helicopter and the environment during mission must be ensured.
- Dynamic scripting is used to alter helicopter capacity and safe zone capacity must be written, adding complexity to the gameplay.

Achieved Results:

- Airplane Citizen Rescue Game's Levels 3 and 4 are rescue missions were executed, with enhanced mission plans and desired assets.
- Safe/waiting zones and rescue zones were introduced in both the levels.
- Mission having timer to increase excitement level of the game and bonus reward after completion of the mission were set up.
- Interaction between the helicopter and the environment during mission was programmed successfully.
- Dynamic scripting technique was used to build, to change helicopter and safe zone capabilities which can be changed by developer later on.

Comparison and Analysis:

- The expected results were achieved successfully, with successful deployment and improvements in rescue missions.
- The integration of safe/waiting zones and rescue zones worked as expected, however minor issues were detected in the safe zone, which were corrected successfully.
- Although mission timers and additional incentives functioned correctly, user feedback and testing revealed flaws in the spawning functionality of citizens, which were fixed.
- It was believed that the interaction between the helicopter and the environment would be smooth; nevertheless, additional testing showed minor issues that were fixed.

5.2.7 Iteration 7

Expected Results:

- It will be expected to plan and implement jet combat levels 5 and 6, focusing on intense aerial fights and compelling gameplay mechanism.
- Gathering of assets such as sound effects, background music, and impact noises to improve the immersive experience would be expected.
- Designing enemy units, including aerial drones and ground enemy, and planning their behavior and attacks according to player's movement will be anticipated.
- Implementation of movement and shooting mechanics for aerial drones, by employing techniques like nav mesh and scripting for accurate control will be expected.
- Dynamic scripting technique will used to vary the behavior of drones, modifying aim accuracy, attack patterns, and movement characteristics will be anticipated.

Achieved Results:

- Jet Combat stages 5 and 6 were developed and realized, featuring fierce aerial battles and thrilling gaming experience.

- Combat mission Assets, including sound effects, background music, and impact sounds, were gathered and blended to improve immersion.
- Enemy units, including aerial drones and ground enemies, were built with planned behaviors and attacks.
- Movement and firing mechanism for airborne drones were created utilizing nav mesh and scripting for accurate control.
- Dynamic scripting technique was used to build drones and alter there behavior according to the developer plan, including aim accuracy, attack patterns, and mobility characteristics.

Comparison and Analysis:

- The concept and implementation of combat levels 5 and 6 met the expectations successfully, providing intense and exciting gameplay.
- The integration of sound effects and background music successfully enhanced the game's immersive experience to player.
- Enemy units like drones and ground enemy were built and behaved as planned; however, small flaws were found in aerial drone movement and firing, which were soon resolved.
- After Implementing movement and firing mechanisms for drones was successful, though some issues in the incoming attack feature were detected and fixed.
- Jet integration with Enemy unit concerns were predicted and detected during testing, and these defects were also resolved.

5.2.8 Iteration 8

Expected Results:

- It will be expected to incorporate rescue and combat stages for final level 7 according to the storyline designed, concluding final mission that mixes together features from both gameplay modes.
- Gathering and testing of sound effects, including user feedback to ensure immersive and impactful effects experiences would be anticipated.
- Planning the aesthetic design of the user interface, where main focus on clarity, usability, and thematic alignment with the game's setting would be expected.

- User Interface components, including buttons, menus, and HUD elements, will be gathered and expected to conform with the proposed design of the game.
- Part of testing will be undertaking End user to test the mission gain feedback on gaming mechanics, level design, and overall user experience.

Achieved Results:

- Rescue and combat levels were linked together, resulting in a final mission that encompassed both gameplay types, resulting as a final boss level.
- Different sound effects were gathered and evaluated, including user feedback for an immersive audio experience.
- The aesthetic UI was designed and executed, focusing on clarity, usability, and theme consistency from the game.
- UI components, including buttons, menus, and HUD elements, were gathered and integrated, keeping with the anticipated design.
- End-user testing was undertaken to gain feedback on gameplay mechanics, level design, and general user experience, so that improvements and optimization can be done.

Comparison and Analysis:

- The combination of rescue and combat levels moved the storyline according to expectation; however, issues were detected after the integration, particularly in integrating the helicopter and jet into the same operation, which were resolved.
- Sound effects and UI components were successfully merged, boosting the game's immersive experience.

5.2.9 Iteration 9

Expected Results:

- In this iteration, complete the user different flow routes like playing missions, pausing game, exiting missions, and quitting the game.
- Development of power-ups in the garage system and integrating it with the reward system for seamless shop feature to be completed.

- Finalizing the everything needed to develop the garage system and ensuring all elements are functional and Interface be user friendly for the user which is one of the key objective.
- The user interface design, including menus and other setting options in both modes, HUD elements, and in-game prompts, will be mainly focused to enhance the narrative experience.
- Conducting final rounds of documentation, also including integration testing, to ensure faultless operation of all components is planned.

Achieved Results:

- The user routing paths for mission navigation and gameplay controls were finished, mainly focusing on ensuring ease of use.
- Inclusion of power-ups into the garage system and integrating it with the reward system was successfully completed.
- Development of the garage system was concluded ensuring functionality and user friendliness.
- The user interface design was refined to enhance the narrative experience and provide easy navigation.
- Final documentation round was completed and reviewed, and after integration testing of different elements confirmed all elements operated smoothly.

Comparison and Analysis:

- Minor problems detected during user testing and stakeholder review were resolved, ensuring the project's readiness for release.

Chapter 6

Conclusion

"Airplane Citizen Rescue Game" emerges as a unique mobile game experience for the users, blending the thrill of aerial battle with the moral objective of saving citizens. Developed in Unity Editor

A three-dimensional game with scripting in C sharp, our venture aimed to fill a space in the android gaming market, delivering players a dynamic and purposeful gameplay experience. The idea for the game originated from a lack in games that blended the thrill of aerial battles and the humanitarian act of rescuing humans. Thus, we undertook the task to build this game, offering to players desiring both adrenaline-fueled action and a sense of righteousness.

Our development journey was led by the principles of the Scrum methodology, fostering agility and adaptation as we went through nine various iterations. Embracing a component-based strategy for game design, as we planned a modular development and scalability for the game, establishing a sturdy basis for our game's evolution.

Beginning with requirement engineering, we reviewed existing game evaluations and conducted interviews to align project goals with technical capabilities. Planning was extremely important, with thorough estimations directing our course forward. Design tasks centered on developing architectural diagrams and system sequence diagrams, assuring a solid framework for implementation.

Implementation is the most important part of project, so translating ideas into concrete gameplay elements was crucial. From building joystick controls to integrating complicated opponent behaviors, each iteration moved us closer to our vision. Testing was rigorous, with user feedback serving as a compass for course corrections. Stakeholder reviews increased our understanding of customer expectations, pushing further refinement.

Documentation, a crucial component of our process, which functioned as a roadmap, recording the evolution of our project and guiding future initiatives. From the Software Requirements Specification (SRS), UML diagrams, Implementation

details to extensive test plans, our documentation showed our dedication to transparency and responsibility.

In conclusion, "Airplane Citizen Rescue Game" stands as a tribute to our dedication to innovation and competence in game production. By leveraging technology and creativity, we have built a game that not only entertains but also inspires. As we look to the future, we take with us the lessons learned and the obstacles overcome, ready to continue pushing the boundaries of gaming.

References

[1] T. Games, "War dogs : Air combat flight s," https://play.google.com/store/apps/details?id=com.teapotgames.wardogs&hl=en&gl=US.

[2] threye Games, "Fighter pilot: Heavyfire," https://play.google.com/store/apps/details?id=com.threye.fighterpilot.cas.a10.

[3] AppsInteractive, "City rescue: Fire engine games," https://play.google.com/store/apps/details?id=com.appsinc.cityrescue.fireengine.games&hl=en&gl=US.

[4] S. Games, "Emergency firefighter police," https://play.google.com/store/apps/details?id=com.shockwavegames.emergency.rescue.service&hl=en&gl=US.