# Tribhuvan University

# Academia International College


## Final Year Project Report
## On
## Mobile Game Application
## [CSC 412]

**Under the supervision of**

**"Er. Ganesh Ram Suwal"**

**Submitted by**

**Chris Maharjan (T.U. Exam Roll No. 26486/077)**

**Palishma Shakya (T.U. Exam Roll No. 26500/077)**

**Luna Thapa (T.U. Exam Roll No. 26496/077)**

**Submitted to**

**Department of Computer Science and Information Technology**

**Academia International College**

**Institute of Science and Technology**

**Tribhuvan University**

**20th January, 2025**

# Tribhuvan University

# Academia International College


## Final Year Project Report

## On

"Mobile Game Application"

[CSC 412]

A final year project submitted in partial fulfillment of the requirement for the degree of Bachelor of Science in Computer Science and Information Technology awarded by Tribhuvan University

**Submitted by**

Chris Maharjan (T.U. Exam Roll No. 26486/077)

Palishma Shakya (T.U. Exam Roll No. 26500/077)

Luna Thapa (T.U. Exam Roll No. 26496/077)

**Submitted to**

Department of Computer Science and Information Technology

Academia International College

Institute of Science and Technology

Tribhuvan University

**20th January, 2025**

# Tribhuvan University

**Institute of Science and Technology**

# Academia International College

**Department of Computer Science and Information Technology**

Email: mail@academiacollege.edu.np


## Supervisor's Recommendation

I hereby recommend that the project work report prepared under my supervision by Mr. Chris Maharjan (26486/077), Ms. Palishma Shakya (26500/077), and Ms. Luna Thapa (26496/077) entitled "Mobile Game Application" be accepted as fulfilling in partial requirements for the degree of Bachelors of Science in Computer Science and Information Technology. In my best knowledge, this is an original work in Computer Science and Information Technology.

.....

Er. Ganesh Ram Suwal

Project Supervisor

Department of Computer Science and Information Technology

Academia International College

Gwarko, Lalitpur


# Tribhuvan University

## Department of Computer Science and Information Technology

## Academia International College

## Certificate of Approval

This is to certify that this project prepared by Mr. Chris Maharjan, Ms. Palishma Shakya, and Ms. Luna Thapa entitled "Mobile Game Application" in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science and Information Technology has been well studied. In our opinion, it is satisfactory in the scope and quality as a project for the required degree.

| | |
|---|---|
| ……..<br>Er. Ganesh Ram Suwal<br>Project Supervisor<br>Department of Computer Science and IT<br>Academia International College | ……..<br>Mr. Bishwas Mathema<br>HOD/Program Coordinator<br>Department of Computer Science and IT<br>Academia International College |
| ……..<br>Internal Examiner<br>Academia International College | ……..<br>External Examiner<br>Central Department of CSIT<br>Tribhuvan University |

# Acknowledgement

We owe our most profound appreciation to Academia International College for giving us a chance to work on this project as part of our syllabus.

Special thanks to our supervisor, Er. Ganesh Ram Suwal (Academia International College), for his consistent guidance, support, and feedback throughout the report's creation. We are generously obligated to him for providing this excellent opportunity to expand our knowledge. It helped us a lot to realize what we studied for.

We would like to express our sincere gratitude to all those individuals, families, friends, colleagues, and teachers for supporting and helping us a lot in finalizing this project within the limited time frame by providing valuable insights and feedback on the report.

Thanking You,

Chris Maharjan (T.U. Exam Roll No. 26486/077)

Palishma Shakya (T.U. Exam Roll No. 26500/077)

Luna Thapa (T.U. Exam Roll No. 26496/077)

# Abstract

The “Catfood Conquest” is a mobile game application that serves to provide a simple and fun source of entertainment to its user. It is a simple offline game from the “endless runner” genre that is catered towards a casual audience with easy and simple yet addictive gameplay mechanics.

This game will consist of a simple UI design for users to navigate through easily, a visually appealing game screen that allows users to control their character (a cat) through various landscapes, collecting various collectables and voiding harmful obstacles along the way. The game will also feature a shop for users to spend their coins (a collectable) to unlock new map(s). To improve the user experience and immersion, we will also be adding a fun background music as well as various SFX for interactions and a settings for the users to control their volume.

The game will be made using Unity engine, with C# as the programming language, its assets will be made using Adobe Illustrator and Medibang or sourced from various free sources. GitHub will be used as a means of collaboration and version control for this project.

# Table of Contents

# List of Figures

# List of Tables

# Chapter 1: Introduction

## 1.1 Introduction

The purpose of this project is to create a simple, fun and casual mobile game application for entertainment purposes, titled “Catfood Conquest”. The main aim for this project was to create a game that deviates from the current unhealthy trends in gaming like the pay to win model.

The gaming industry has evolved from a niche hobby to a global entertainment powerhouse, captivating audiences of all ages and backgrounds. Ranging from triple A(AAA) games created by Top tier companies to smaller indie (independent) games created by small independent creators. Our project aims to contribute to this gaming industry.

The game is a simple endless runner game that can be played immediately after installation. It is an offline game so no Wi-Fi connection is required to play the game. It is a single-player game, meaning there is no stress of competing with online opponents, allowing players to enjoy a relaxed experience at their own pace.

Although similar to many other games in this genre, Catfood Conquest aims to create a simple and more straightforward User Interface, where the player, portrayed by a cat, runs in an infinite straight line across the platform, that will take off to be a celestial body in outer space, where they need to collect cat food and avoid obstacles.

The lore of the game is targeted to invite people to the Catfood Conquest gaming experience and the UX aims to keep these people invested in the game. It is a hyper-casual game idea whose key concepts will involve simple, addictive gameplays designed for short play sessions.

The game will be conceptualized in Figma, designed in Medibang and developed in the Unity Engine with the help of C# programming language.

## 1.2 Problem Statement

With the increase in usage of mobile phones, we can see a massive increase in the number of mobile games being created in order to capitalize on the large number of users (mostly children) that are interested.

Although this may seem like a good thing, due to the scope of revenue that can be generated through mobile games, we see a lot of games developed solely for the purpose of earning money which defeats the whole purpose of developing games which is to entertain and engage users. Some major issues could be as follows:

- Microtransactions:
  Microtransactions are basically online transactions in games where you spend real world money to obtain a virtual product within the game. Although at base, it may seem like a good revenue model for games, the games now-a-days are overly reliant on it. This has happened to a point where a game cannot be played at its fullest without performing some microtransactions. This is more visible in competitive games, where using real world money to buy in-game items give those players a competitive advantage creating a "Pay-to-Win" model of gameplay.
- Over competitiveness:
  Speaking of competitive games, these are games where you compete with other players across the internet, either head on or in a strategic environment. Simply put, games now-a-days have a large focus on the competitive aspect, as we can see most popular games are online multiplayer games. Although some may enjoy this sort of competitive environment, the causal players may feel discouraged to play games, especially with the added "Pay-to-Win" model discussed earlier. Moreover, the over competitiveness also creates a sort of a "toxic" community or environment that drains the joy out of gaming.
- Advertisements:
  From a business perspective, advertisements in games may not sound bad, as it could be a good monetization model for revenue generation while not impacting the gameplay. However, games now-a-days have become overly populated with ads, to a point where in some games, you spend more time watching ads than playing the

actual game. These ad focused games contain unavoidable and sometimes un-skippable ads that creates a negative user experience.

## 1.3 Objectives

The primary objective of this project is to develop a game that is both simple and easy to play while still being enjoyable as well. The objectives we hope to achieve within this project are as follows:

- Easy to use UI/UX: A simple and clean looking UI design that allows users of all kind to easily navigate through the game and have an enjoyable User Experience (UX).
- Smooth gameplay mechanics: Proper implementation of the core functionalities of the game like player movement, object interactions, etc.
- Device compatibility: Compatibility with most mobile devices such as IOS and Android.
- Local data storage: Storing and retrieving user data through the use of local storage techniques.
- Audio integration: Integration of audio for background music as well as SFX for various user interactions.

## 1.4 Scope and Limitation

The scope we have determined for this project are as follows:

i. Core gameplay features: The game will include the core functionalities of moving the playable character left and right, collecting coins and collectables and avoiding obstacles. The game ends upon colliding with obstacle and the obtained collectables are stored in local storage. It will also include a pause button to pause the game if necessary.
ii. Audio settings: The game will include background music and sound effects that the user can easily adjust the volume of.

iii. Other gameplay mechanics: The game will include a shop to spend the coins collected to unlock new levels and a level selection screen to select the unlocked levels.
iv. Other UI features: The game will include a how to play section that simple describes each object and what it does, an about us page that contains small details about the developers.

The limitations of this project are as follows:

i. Login system: As the backend would require hosting a server, and the free ones need to be restarted every so often, the game will not consist of a login system to a server.
ii. Limited maps and ships: Due to provided time limit, there are currently only two maps and a few ships built in the game, limiting user customization.
iii. Inconsistent Art Style: Due to the assets being designed by the development team, the visuals implemented in the game may seem inconsistent and lacking of harmony.

## 1.5 Development Methodology

The methodology used in developing this project is the Agile methodology. This approach to project management involves breaking the project into smaller phases, usually referred to as “sprints”. This methodology emphasizes on the continuous collaboration among team members.

In the initial stage, the project team identifies and documents the needs and expectations of various stakeholders, including clients, users, and subject matter experts which for our case is simple the expectations we have for our project. Using those expectations, we start creating a project plan and allocating resources. In the design phase, detailed specifications are created, which includes data structures, algorithms, interfaces. In development phase, we write the actual code for the software while concurrently conducting unit testing to verify the functionality of individual components. In testing phase, we ensure that different components work together and make sure that the software meets user requirements. Finally, in the deployment phase we build the software into the real world where people

can use it. Additionally, we can have a review (maintenance) phase, addressing and resolving any issues that may arise after deployment and releasing updates and patches to enhance the software and address those problems.

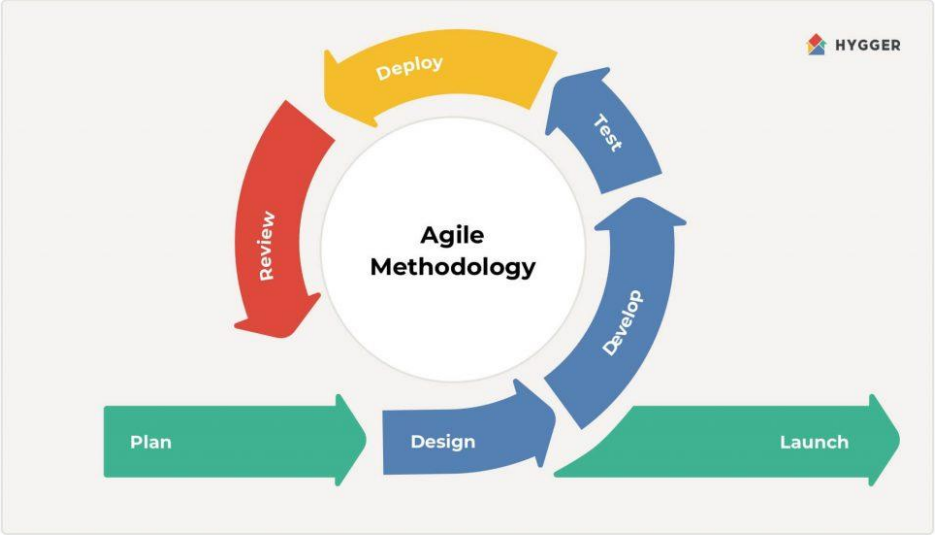


**Figure 1 Agile Methodology**

## 1.6 Report Organization

Project Organization refers to the structure and coordination of all activities and resources within a project. It's essential for ensuring efficient, successful, and on-time project completion. The initial section of the report includes a Title Page, Certificate Page, Acknowledgement, Abstract, Table of Contents, and lists of Abbreviations, Figures, and Tables.

The main report is organized into 6 chapters, aligning with their respective headings and content, which include:

Chapter 1: Introduction

It provides a brief overview of the project by outlining aspects such as project introduction, problem statement, objectives, scope and limitations, and methodology.

Chapter 2: Background Study and Literature Review

It addresses the background study of the project and conducts a review of existing literature, summarizing relevant projects, papers, and articles.

Chapter 3: System Analysis

It focuses on system analysis, covering requirements and feasibility analysis. Functional requirements of the system are defined through a use case diagram. A Gantt chart is used to visually illustrate the time taken for various tasks in the project.

Chapter 4: System Design

It dives deep into system details, focusing on the implementation process and designing the model architecture, database, interface, and forms. Additionally, it provides insights into the algorithms employed in the system.

Chapter 5: Implementation and Testing

It discusses the implementation and testing details, including an overview of the tools and dependencies utilized for implementing the system.

Chapter 6: Conclusion and Recommendations

It summarizes and concludes the project and explores possibilities for future enhancements of the project. The report's final section includes References following IEEE standards and Appendices containing system screenshots and essential source code snippets.

# Chapter 2: Background Study and Literature Review

## 2.1 Background Study

The background study involves exploring the fundamental theories, general concepts, and terminologies related to our project.

The genre of our game project is “endless runner”. Endless runner games belong to the genre of action games where the player character is continuously moving through a procedurally generated world while avoiding obstacles and collecting collectables. These games consist of very basic and simple mechanics:

- Continuous movement: The player is endlessly running without any input from the user. The user has to then simply change directions or respond to obstacles that appear.
- Obstacles: The game spawns various obstacles that the player has to avoid by various means like jumping, sliding, dodging, etc. The difficulty increases as the player progresses.
- Collectables: The game consists of various collectables like coins and powerups that the player can collect for various purposes.
- Procedural Generation: The environment, obstacles and collectables are generated randomly, so very run feels entirely unique.
- Score system: The game keeps track of score that goes up the further you progress or the longer you survive and may also depend upon the collectables collected. It also keeps track of the highest score allowing users to compare with others or try to beat their previous high scores.

This genre has a very addictive gameplay as every run feels unique due to the randomness. There are a lot of games leading the market that fall within this genre of “endless runner” games.

## 2.2 Literature Review

“Endless runner” is a very common genre for mobile games these days. These games typically involve the player running through an infinite environment, avoiding obstacles, collecting items, and achieving high scores. The simplicity of the game mechanics, combined with the addictive nature of the gameplay have contributed to its widespread popularity with games like Temple Run (2011), Subway Surfers (2012), and Jetpack Joyride (2011) leading the market.

- Subway Surfers:
  One of the leading games in the market in the “endless runner” genre is Subway Surfers. It is a game where the player is chased by a non-playable character (NPC) and has to navigate their way through a subway, avoiding obstacles like trains and signs, by moving across lanes, jumping or rolling, collecting coins and other collectables including powerups that provide some sort of boost to the player. This game contains a lot of microtransactions that allow players to effectively never lose in a run using items like hoverboards and keys for revival upon losing. In addition to that, the game is filled with unavoidable ads that make the user experience terrible.
- Temple Run:
  Another game that is in competition with Subway Surfers for the leading game is Temple Run, or specifically Temple Run 2. This franchise has a couple of games under the same name like Temple Run, Temple Run 2, Temple Run Oz, etc. The concept of the game is similar, the player is chased by NPCs and has to navigate along a temple path, avoiding obstacles like roots and broken paths. This game also contains microtransactions that allow for buying characters that provide abilities that make the game easier as well as in game currency “Diamonds” that revives the player upon losing, effectively creating a never-ending run. This game also contains a lot of ads, which combined with clustered and unorganized UI design, make the user experience bad.

- Jetpack Joyride:

  Jetpack Joyride, once upon a time, was also a leading game in the market in this genre of games. Unlike the previous two, this game uses a landscape view but the core mechanics are the same where the player is chased by NPCs, in a science laboratory and has to avoid obstacles like lasers and missiles, collecting collectables on the way. Similar to previous games, this game also contains a lot of microtransactions that make the game easier as well as a revive system making a run effectively infinite. Along with that, this game, as expected, also contains a lot of ads everywhere making user experience very poor.

- Dinosaur Game:

  A classic example of "endless runner" game is the Dinosaur Game that we can play in our google browser, especially when the internet is not working. It is a simple game in design, just a dinosaur running while avoiding cactus and birds by jumping and ducking. As such, there is no more to it, it is simply a browser game with effectively no UI. We can tell that it doesn't keep the user engaged enough as people usually only play it when the internet is not working and even then, would rather play other offline games instead.

# Chapter 3: System Analysis

## 3.1 System Analysis

### 3.1.1 Requirement Analysis

i. Functional Requirements

The functional requirements considered for this project are as follows:

- Buttons to move the player in the game.
- A shop for players to spend their coins.
- Proper display for player's inventory of coins and other collectables.
- Settings to manage the volume of background music and SFX.

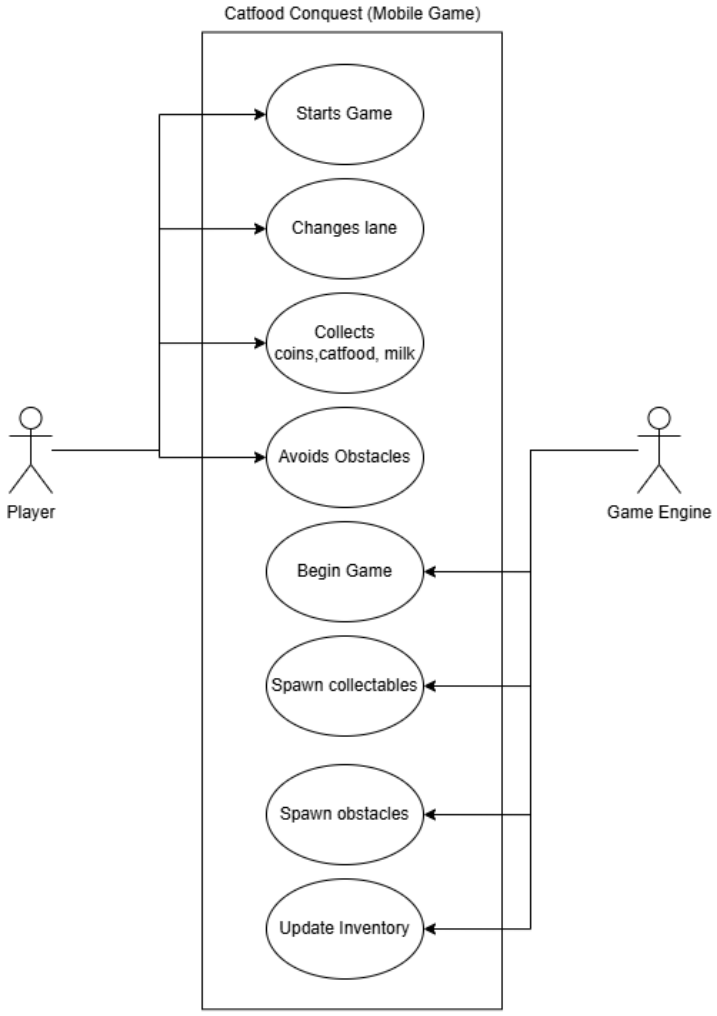


**Figure 2 Use Case Diagram**

ii. Non-Functional Requirements

The non-functional requirements considered for this project are as follows:

- Easy navigation buttons to navigate through scenes like shop, level select, etc.
- Optimized system that runs smoothly without any stutters.
- Platform Support: Support for major mobile platforms (e.g., Android, iOS).

### 3.1.2 Feasibility Analysis

i. Technical

The technological feasibility of this project can be understood in terms of: Game Engine and Programming language, Asset Creation, and Hardware Compatibility.

- The game will be developed in Unity Engine using C# as its programming language. Features of Unity such as: cross-platform compatibility, robust feature set, active community and features of C# such as: integration with Unity, performance and efficiency make this set a preferred choice for this project.
- Medibang Paint implements 2-D asset creation. Familiarity with the application and its tools make it easy and efficient to use. A blend of custom-created and royalty-free assets will contribute to the game's visual appeal.
- The game will be designed to run on a wide range of mobile platforms. Due to minimal use of complex graphic forms, it enables optimized performance.

ii. Operational

The project’s development team consists of three members, each with their field of interest and competence. The fields explore game development, graphic design and UI design. The development team will employ agile methodologies to ensure efficient development, flexibility, and regular progress updates. After completion, the project will be tested to identify and address bugs and ensure a high-quality gaming experience.

iii. Economic

This project is made using Unity Engine, which is a free engine for developing 2D or 3D games, Visual Studio is used for coding in C# which is also a free and open-

source software. Similarly, all the assets are either made in Medibang, which is a free tool for drawing, or obtained via other free sources. Hence, this project is economically viable.

iv. Schedule

The estimated development time is 6 months, divided into phases for planning, development, testing, and polishing. The phases can be further divided into sub activities as follows:

**Table 1 Activity Schedule**

| S.N. | Activity | Start Date | End Date | Duration |
|---|---|---|---|---|
| 1 | Project Selection and Planning | 14-Jul | 14-Jul | 2 |
| 2 | Game Concept Design | 16-Jul | 18-Jul | 2 |
| 3 | Art Asset Creation | 19-Jul | 18-Aug | 30 |
| 4 | Programming Core Gameplay | 20-Jul | 10-Aug | 21 |
| 5 | UI Design | 15-Aug | 29-Aug | 14 |
| 6 | Sound and Music | 1-Sep | 15-Sep | 14 |
| 7 | Testing and Debugging | 20-Sep | 4-Oct | 14 |
| 8 | Final Polishing and Deployment | 5-Oct | 4-Nov | 30 |
| 9 | Final Documentation | 19-Jul | 15-Nov | 10 |

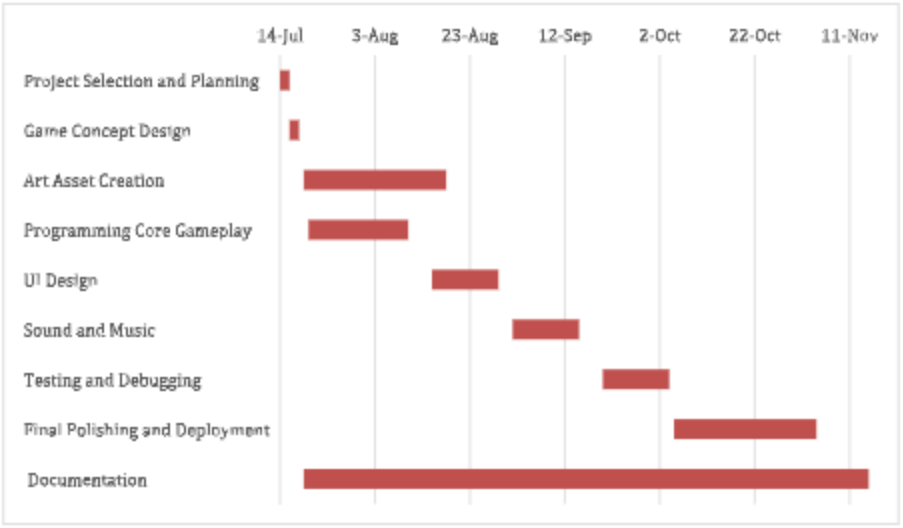


**Figure 3 Gantt Chart**

### 3.1.3 Analysis (Object Oriented)

This project of developing a mobile game follows a object-oriented approach as the programming language used (C#) is an object oriented programming language.

i. Object modelling using Class and Object Diagrams

The Class and Object diagram for our project is as follows:

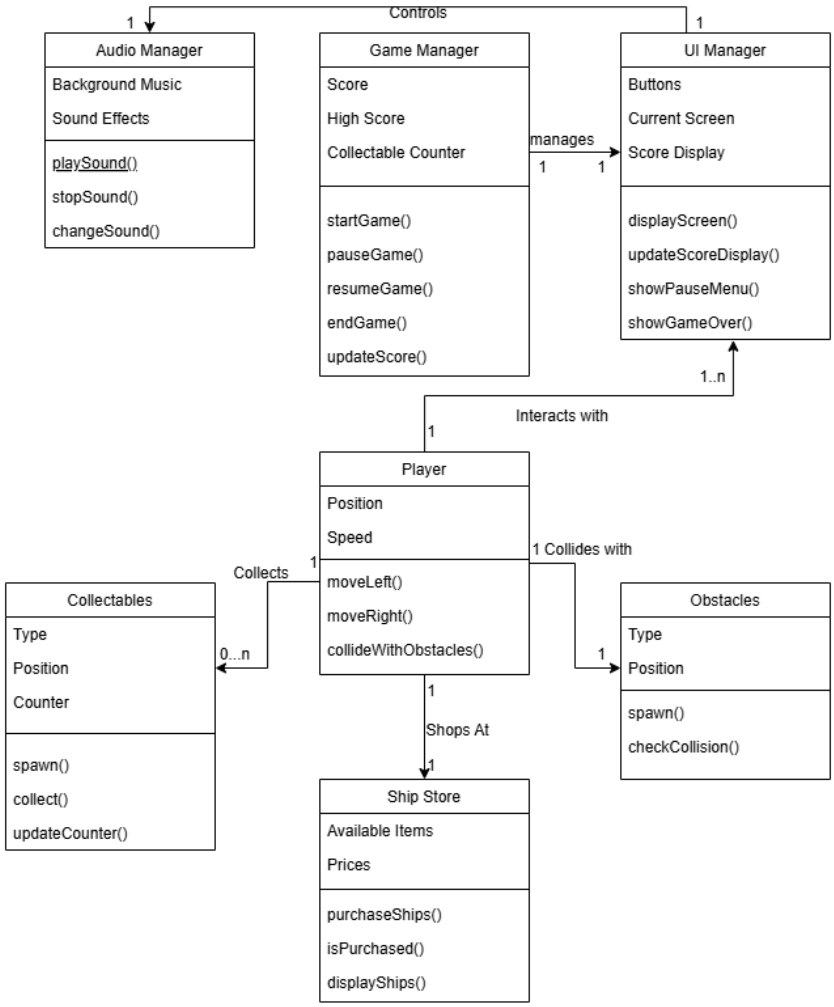


**Figure 4 Class Diagram**

This diagram shows all the classes involved within our project and how they are related with another. Our project only creates a default instance or object for each of these classes so the object diagram would be the same.

ii. Dynamic modelling using State and Sequence Diagrams

The state diagram for our project is as follows:

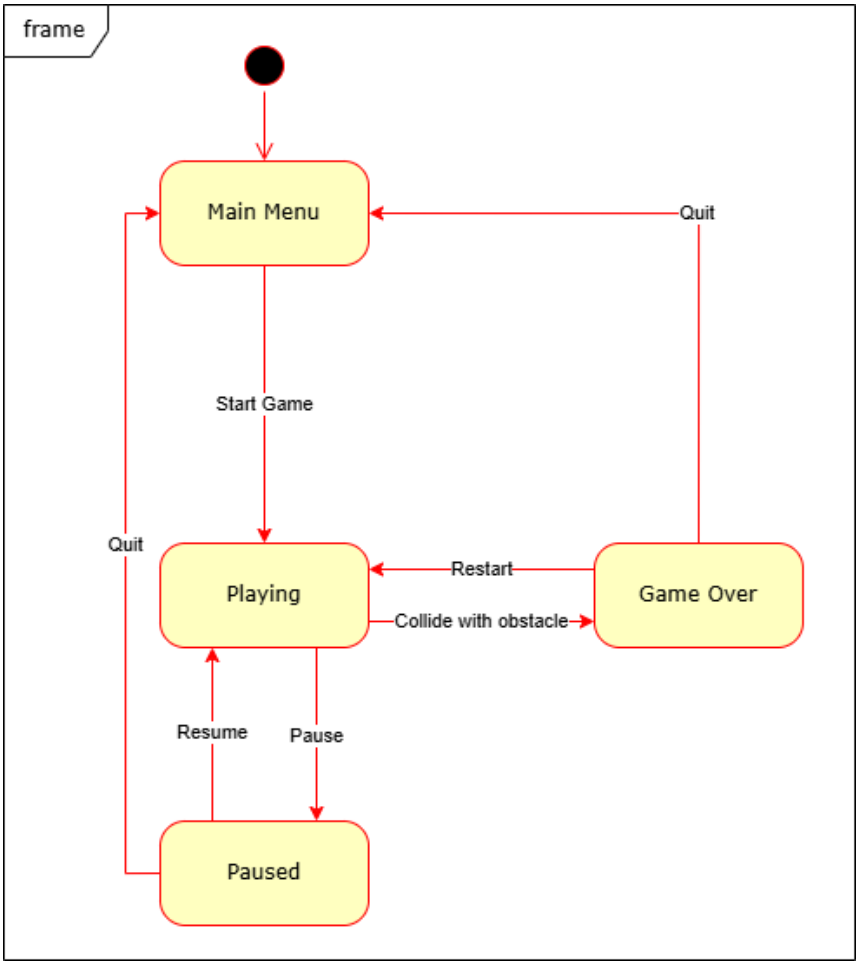


**Figure 5 State Diagram**

This diagram shows the different states within our game and how it changes from one state to the other depending on various player events.

The sequence diagram for our project is as follows:

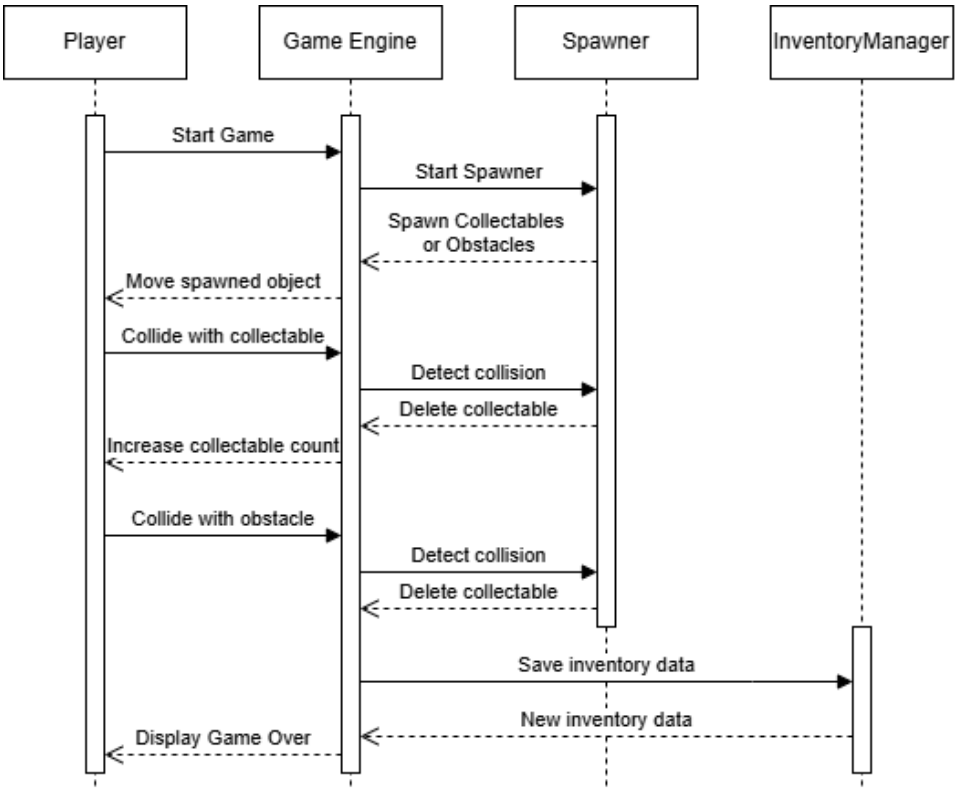


**Figure 6 Sequence Diagram**

This diagram shows the sequence of events and interactions between the various components from start of a game till the end of a game.

iii. Process modelling using Activity Diagrams

Activity diagrams are simply an advanced from of flow-chart diagram that model the working of the system from one activity to another. An activity diagram for our project is as follows:

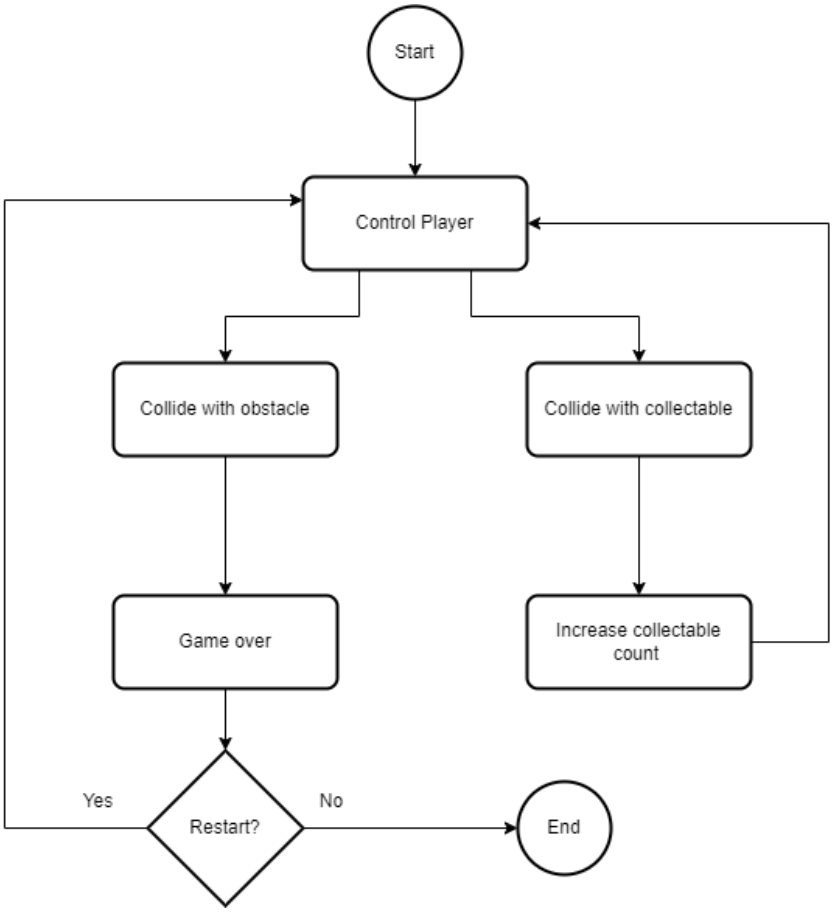


**Figure 7 Activity Diagram**

This diagram shows the activities that occurs once you start the game. Once started, you control the player character, moving left and right as obstacles and collectables spawn. If you collide with a collectable, the count of that collectable increases, and the process repeats. However, if you collide with an obstacle, you lose and hence the game is over. You are then given the option to restart the game or not, which if you chose yes then you go back to the start, else the game ends and you are brought out of the game scene, into the Title Scene.

# Chapter 4: System Design

## 4.1 Design (Object Oriented)

As discussed in the analysis chapter, the system design also follows an object-oriented approach.

- Refinement of Class, Object and Activity Diagram

  Since this is a very basic project with not much complexity, the class diagrams, object diagrams and activity diagrams designed earlier are already refined enough to be applicable within the project.

- Component Diagram

  The component diagram for our project is as follows:

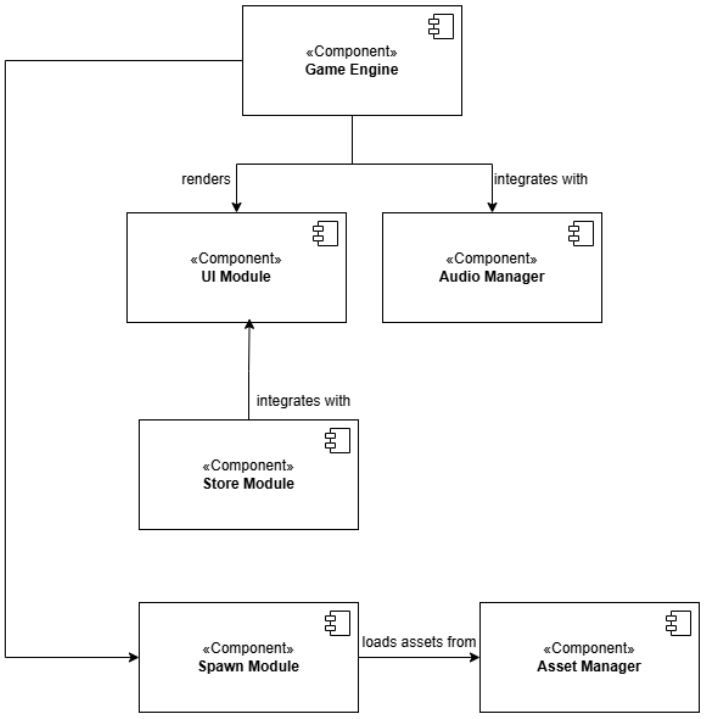


**Figure 8 Component Diagram**

This diagram shows all the component modules that we have built within our project and how they interact with each other.

## 4.2 Algorithm Details

The algorithms used within this project are as follows:

i. Randomized Spawning Algorithm: A probability-based randomization algorithm that randomly spawns(places) objects like obstacles and collectables. We use a cooldown-based method to implement rarity in the spawned objects and a recent spawn tracking method to make sure there is variety in spawning objects.

   This algorithm works as follows:

   1. Initially a cooldown value for each rare item is set to the minimum number of objects that need to spawn before they spawn.
   2. At the start of the algorithm a random value is chosen from range 1 to 100
   3. If the random value is below 60, another random value is chosen from range 1 to 3, depending on which, one of the common objects is spawned (60% probability).
   4. Else if the random value is less than 85, and the cooldown value for a rare object is 0, the rare item is spawned (25% probability).
   5. Else if the random value is greater than 85, and the cooldown value for a rarer object is 0, the rarer item is spawned (15% probability).
   6. Else if none of these conditions are true, it falls back to the common item spawn. So, random value is chosen from range 1 to 3, depending on which, one of the common objects is spawned.
   7. Finally, cooldowns are decremented.
   8. Recent spawns are tracked using a simple queue that enqueues spawned item type from one end and dequeues them from the other once it reaches the threshold
   9. Then that stack is checked in each step to see if the next spawn object is already within that stack or not.

   We also use a constrained randomization algorithm to randomize the lane where the obstacles spawn making sure the same lane isn't chosen too many times consecutively.

ii. Difficulty Scaling Algorithm: A simple difficulty scaling algorithm that increases the difficulty as time passes, to make the game gradually more challenging. We use

a simple linear difficulty scaling algorithm that simply increases the movement speed of the player and decreases the spawn interval of obstacles.

It is as simple as just incrementing the movement speed and decrementing the spawn rate variables withing the Update function provided by unity, that executes every frame the game is running.

iii. Infinite Scrolling Algorithm: We use a simple movement algorithm on each spawn able object to move it down once they spawn and a constant looping animation in the background to provide an illusion of an infinite scrolling environment.

# Chapter 5: Implementation and Testing

## 5.1 Implementation

### 5.1.1 Tools Used

The tools we used for implementing this project are as follows:

iv. Unity

Unity is a cross-platform game engine developed by Unity Technologies, that supports a variety of desktop, mobile as well as console platforms. It gives users the ability to create games and experiences in both 2D and 3D, and the engine offers a primary scripting API in C# using Mono, for both the Unity editor in the form of plugins, and games themselves, as well as drag and drop functionality.

v. C#

C# is a general-purpose high-level programming language based on object-oriented programming created by Microsoft. It is a popular and easy to learn as well as simple to use programming language having huge community support.

vi. Medibang

Medibang is an illustration application designed to make the best creative environment for iPad artists. It allows creators to make the art they want to see using all kinds of brushes, fun filters, and other intuitive tools.

vii. Illustrator

Adobe Illustrator is a vector graphics editor and design software developed and marketed by Adobe that allows for the creation of everything from single design elements to entire compositions.

viii. GitHub

GitHub is one of the most widely used platforms for hosting code. It allows for version control, to track the changes made to the project and the sate of progress of the project. It also allows to collaborate on projects from anywhere and monitor the activities of the team members on the project.

### 5.1.2 Implementation Details of Modules

This project consists of the following modules:

i. Game Module

This is the central component of the game, in charge of managing the underlying mechanics and systems.

- Physics: Controls object interactions, player movement, and collision detection.
- Rendering: Produces visual representations of the game's characters, obstacles, and surroundings.
- Input handling: Manages the player's actions by processing user inputs like taps and swipes.

ii. UI Module

The User Interface module manages all the screens and visual elements that the player interacts with.

- Main Menu: The initial screen for starting a game, accessing settings, or entering the store.
- Pause Menu: Provides options when the game is paused (e.g., resume, settings, or exit).
- Game Over Screen: Displays the player's final score, rewards, and options to restart or go to the main menu.

iii. Audio Module

Manages all of the game's audio-related functions: Music: Plays background tracks, such as the theme music.

- Sound Effects: Plays sounds for events like collecting coins, or colliding with obstacles.

iv. Asset Module

Manages all game assets and ensures efficient use of resources:

- Assets: Handles models, textures, and animations for characters, obstacles, and environments.

## 5.2 Testing

### 5.2.1 Test Cases for Unit Testing

Unit testing refers to testing individual units or components (such as functions, methods, or classes) to ensure that each part of the code works as expected. Some test cases are as follows:

Test Case 1: Player Movement Function

Objective: To verify that the function responsible for moving the player character works as expected.

**Table 2 Player Movement Function Test**

| Test ID | Test Steps | Expected Result | Actual Result | Remarks |
|---|---|---|---|---|
| 1 | Tap on the left button | Player moves left | Player moves left | Pass |
| 2 | Tap on the right button | Player moves right | Player moves right | Pass |
| 3 | Tap on the left button when player is on left side of screen | Player stays on same position | Player stays on same position | Pass |
| 4 | Tap on the right button when player is on right side of screen | Player stays on same position | Player stays on same position | Pass |

Test Case 2: Obstacle Collision Function

Objective: To verify that the function responsible for ending the game works accordingly.

**Table 3 Obstacle Collision Function Test**

| Test ID | Test Steps | Exoected Result | Actual Result | Remarks |
|---|---|---|---|---|
| 1 | Let the player character collide with a rock | The game ends and gameover scene is shown. | The game ends and gameover scene is shown. | Pass |
| 2 | Restart the game and let the the character collide with a poison bottle. | A paused scene is shown that allows you to continue the game by spending milk. | A paused scene is shown that allows you to continue the game by spending milk. | Pass |
| 3 | Click on continue. | The milk count goes down and the game resumes. | The milk count goes down and the game resumes. | Pass |
| 4 | Repeat step 3 and click on quit. | The game ends and gameover scene is shown. | The game ends and gameover scene is shown. | Pass |

### 5.2.2 Test Cases for Integration Testing

Integration testing refers to ensuring that different components of the game, which were developed independently, work together as expected when integrated.

Test Case 3: Player Data Persistence

Objective: To ensure that the game correctly saves and loads the player's score and progress to/from a local storage or cloud database.

**Table 4 Player Data Persistence Test**

| Test ID | Test Steps | Expected Result | Actual Result | Remarks |
|---|---|---|---|---|
| 1 | Start a new game and play a level, collecting coins and points | The coin counter and score goes up accordingly | The coin counter and score goes up accordingly | Pass |
| 2 | Collide with an obstacle to end the game | GameOver screen appears and your progress is saved | GameOver screen appears and your progress is saved | Pass |
| 3 | Close the game and reopen it | The coin counter and highscore restored correctly | The coin counter and highscore restored correctly | Pass |

### 5.2.3 Test Cases for System Testing

System testing refers to testing the entire game as a whole to ensure all components function together as expected.

Test Case 4: Full Game Playthrough

Objective: To verify that the game runs smoothly from start to finish without major issues or crashes.

**Table 5 Full Game Playthrough Test**

| Test ID | Test Steps | Expected Result | Actual Result | Remarks |
|---|---|---|---|---|
| 1 | Launch the game | Game opens with no issues | Game opens with no issues | Pass |
| 2 | Play through a few times, completing objectives | Gameplay is fluent and progress is saved throughout the game | Gameplay is fluent and progress is saved throughout the game | Pass |
| 3 | Purchase items in the shop | Shop display changes accordingly | Shop display changes accordingly | Pass |
| 4 | Restart the game, load progress, and continue playing | All the player data is reloaded correctly | All the player data is reloaded correctly | Pass |

## 5.3 Result Analysis

Result analysis comprised of examining the results of tests at each step of development. Through this analysis, we were able to analyze and identify errors and issues, their effects, locate their causes and resolve them before it caused any major issues. Once the system was verified to be flawless, we built an APK file to test the compatibility on different android devices. We allowed our friends to test the system and collected their feedback to fix any bugs found and improve on the system accordingly.

# Chapter 6: Conclusion and Future Recommendations

## 6.1 Conclusion

The hyper-casual mobile game, Catfood Conquest aims to be a fun and interactive indie game in the genre of “Endless Runner”. Integrated with C# programming language with assets designed in Medibang Paint, the project has been brought to life with libraries and frameworks of the Unity Game Engine. The game tests the human capability to react to environmental stimuli based on how they successfully avoid the obstacles and grab the collectables all of which are spawned on the basis of random spawning controlled by the randomization algorithm. To add to the interestingness of the game, difficulty scaling has been implemented assuring the game remains challenging and engaging.

## 6.2 Future Recommendations

- The protagonist of the game upgrades skins based on the missions accomplished.
- The game can include multiple maps (specially more planets of the solar system) and more ships to travel to these planets.
- The project can adopt a login system, that opens the player to global competitiveness where players globally can compete for spots on leaderboards.