**Final Year Project Report**

**Python Coding Playground**

****

**Project Advisor:**

**Khawaja Ubaid ur Rehman**

**Submitted By:**

**Fahad Hassan (f2020266453)**

**Session**

**University of Management and Technology**

**C-II Johar Town Lahore Pakistan**

**Dedication**

This game, 'Python Coding Playground,' is a heartfelt dedication to the imaginative minds of young learners, aged 5-10, who embark on a journey into the world of coding. We believe that every child possesses a unique spark of creativity and curiosity, and this game is designed to ignite that flame through the magic of Python. To the budding innovators, dreamers, and future architects of our digital landscape, this endeavor is devoted to making the learning experience joyful, interactive, and filled with wonder. May 'Python Coding Playground' be the catalyst that transforms coding education into a thrilling adventure, fostering a love for learning and empowering the next generation of tech enthusiasts. Together, let's code, create, and inspire the future.

## 

**Final Approval**

* **Head of Department**  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Department of Software Engineering

School of Systems & Technology

UMT Lahore

* **Program Director (Final Year Projects)** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Department of Software Engineering

School of Systems & Technology

UMT Lahore

* **Supervisor** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Department of Software Engineering

School of Systems & Technology

UMT Lahore

* **Co-Supervisor** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Acknowledgment**

**Project Title**

Python Coding Playground

**Objective**

Python Coding Playground' aims to make learning Python enjoyable and accessible for kids aged 5-10. Through interactive gameplay, the game introduces coding concepts, fostering foundational skills, logical thinking, and problem-solving. We aspire to spark a love for learning and creativity, empowering young minds for a tech-savvy future."

**Undertaken by**

**Supervised by**

Khawaja Ubaid ur Rehman

**Starting Date**

**Completion Date**

**Tools Used**

Unity  
Blender  
Pika art AI

Visual Studios

**Operating System**

Windows 10

**Documentation**

**Plagairism Report**

**Declaration Form**

(*To be submitted by FYP Advisor*)

I have carefully examined the documentation of the Final Year Project titled *“\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_”*; and I endorse that this documentation complies with the standards of an undergraduate level Final Year Project report.

The document has been checked for plagiarism through Turnitin software available in UMT Library. The similarities of the document are within acceptable range.

Moreover, the accompanying CDs contain PDF of the documentation, as well as the source code and binaries with user manual and installation guide.

**FYP Advisor Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Abstract**

Python Coding Playground' is an innovative educational game designed to introduce children aged 5-10 to the fundamentals of Python programming in a playful and engaging manner. Through a combination of interactive gameplay and educational content, the project aims to demystify coding concepts, nurture logical thinking, and develop problem-solving skills. The objective is to create a dynamic and enjoyable learning environment that sparks the curiosity of young minds, fostering a love for coding and setting the foundation for future technological literacy. 'Python Coding Playground' is more than just a game; it is a stepping stone toward empowering the next generation with essential coding skills in a fun and interactive way.

Revision Chart

This chart contains a history of this document’s revisions. The entries below are provided solely for illustration purposes. Those entries should be deleted until the revision/s they refer to have actually been created.

The document itself should be stored in revision control, and a brief description of each version should be entered in the Revision Control System. A brief description can be repeated in this section. Revisions need not be described elsewhere in the document, unless they explain the document.

| Version | Primary Author(s) | Description of Version | Date Completed |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Contents

*New paragraphs formatted as Heading 1, Heading 2, and Heading 3 will be added to the table automatically. To update this table of contents in Microsoft Word, put the cursor anywhere in the table and press F9. If you want the table to be easy to maintain, do not change it manually.*

[Contents ix](#_Toc119337485)

[Definitions and Acronyms xi](#_Toc119337486)

[List of Figures xii](#_Toc119337487)

[List of Tables xiii](#_Toc119337488)

[1. GAME OVERVIEW 1](#_Toc119337489)

[1.1 Game Summary 1](#_Toc119337490)

[1.2 Target Platform(s) 1](#_Toc119337491)

[1.3 Business Model 1](#_Toc119337492)

[1.4 Theme / Setting / Genre 1](#_Toc119337493)

[1.5 Core Gameplay Mechanics 1](#_Toc119337494)

[1.6 Server / Online Mechanics 1](#_Toc119337495)

[1.7 Art Style 1](#_Toc119337496)

[1.8 Look & Feel/ Camera 2](#_Toc119337497)

[1.9 Progression 2](#_Toc119337498)

[2. Story and Gameplay 3](#_Toc119337499)

[2.1 Story 3](#_Toc119337500)

[2.2 Core Gameplay 3](#_Toc119337501)

[3. Domain Analysis 4](#_Toc119337502)

[3.1 Customer 4](#_Toc119337503)

[3.2 Stakeholders 4](#_Toc119337504)

[3.3 Affected Groups with social or economic impact 4](#_Toc119337505)

[3.4 Dependencies/ External Systems 4](#_Toc119337506)

[3.5 Reference Documents 4](#_Toc119337507)

[3.5.1 Related Projects 4](#_Toc119337508)

[3.5.2 Feature Comparison 5](#_Toc119337509)

[4. Requirements analysis 6](#_Toc119337510)

[4.1 Requirements 6](#_Toc119337511)

[4.2 List of Actors 7](#_Toc119337512)

[4.3 List of use cases 7](#_Toc119337513)

[4.4 System Use Case diagram 7](#_Toc119337514)

[4.5 Extended use cases 8](#_Toc119337515)

[5. User Interface / Screens 10](#_Toc119337516)

[5.1 Main Menu 10](#_Toc119337517)

[1. Gameplay Screen 10](#_Toc119337518)

[6. Levels 12](#_Toc119337519)

[6.1 Tool for Level Designing 12](#_Toc119337520)

[6.2 First Time Experience 12](#_Toc119337521)

[7. Asset List 13](#_Toc119337522)

[8. Implementation details 14](#_Toc119337523)

[8.1 Development Setup 14](#_Toc119337524)

[8.2 Deployment setup 14](#_Toc119337525)

[8.3 Algorithms 14](#_Toc119337526)

[8.4 Constraints 14](#_Toc119337527)

[8.4.1 Assumptions 14](#_Toc119337528)

[8.4.2 System constraints 14](#_Toc119337529)

[8.4.3 Restrictions 14](#_Toc119337530)

[8.4.4 Limitations 14](#_Toc119337531)

[9. Testing 15](#_Toc119337532)

[9.1 Extended Test Cases 15](#_Toc119337533)

[15](#_Toc119337534)

[9.2 Decision Table 15](#_Toc119337535)

[9.2.1 Code snippet 15](#_Toc119337536)

[9.2.2 Decision coverage table 15](#_Toc119337537)

[9.3 Traceability Matrix 16](#_Toc119337538)

[9.3.1 RID vs UCID (requirements vs use cases) 16](#_Toc119337539)

[9.3.2 Prototypes (RID vs PID) 17](#_Toc119337540)

[9.3.3 Test Cases (RID vs TID) 17](#_Toc119337541)

[9.3.4 Coverage (UCID vs TID) 17](#_Toc119337542)

[10. Results/Output/Statistics 18](#_Toc119337543)

[10.1 %completion 18](#_Toc119337544)

[10.2 %accuracy 18](#_Toc119337545)

[10.3 %correctness 18](#_Toc119337546)

[11. Conclusion 19](#_Toc119337547)

[12. Future work 20](#_Toc119337548)

[13. Bibliography 21](#_Toc119337549)

[13.1 Books 21](#_Toc119337550)

[13.2 Journals 21](#_Toc119337551)

[13.3 Articles 21](#_Toc119337552)

[13.4 Research papers 21](#_Toc119337553)

[13.5 Other References 21](#_Toc119337554)

[14. Appendix 22](#_Toc119337555)

[14.1 Glossary of terms 22](#_Toc119337556)

[14.2 Pre-requisites 22](#_Toc119337557)

## Definitions and Acronyms

*Provide definitions or references to all the definitions of the special terms and acronyms used within this document*

Table 0.1.1: List of acronyms and definitions

|  |  |
| --- | --- |
| **Acronym** | **Definition** |
| TPP | Third Person Perspective |
|  |  |

## List of Figures

New figures that are given captions will be added to the table automatically.

* **Insert caption:**
  1. select picture
  2. right click
  3. select “insert caption”
  4. under “options”, choose label as “figure”
  5. Under “caption”, an automatic insertion of “figure no” will appear. Give your figure an appropriate caption
* **Update list of figures:**To update this list in Microsoft Word, put the cursor anywhere in the table and press F9.
* **Note:**  If you want the table to be easy to maintain, do not change it manually.

[Figure 4.1: Use case diagram of ATM 7](#_Toc119337558)

[Figure 7.1: Asset list of Dead Zombie FPS game 13](#_Toc119337559)

## List of Tables

New Tables that are given captions will be added to the table automatically.

* **Insert caption:**
  1. select picture
  2. right click
  3. select “insert caption”
  4. under “options”, choose label as “table”
  5. Under “caption”, an automatic insertion of “table no” will appear. Give your table an appropriate caption
* **Update table:** To update this table of contents in Microsoft Word, put the cursor anywhere in the table and press F9.
* **Note:**  If you want the table to be easy to maintain, do not change it manually.

[Table 0.1.1: List of acronyms and definitions xi](#_Toc119337560)

[Table 3.1: List of stakeholders 4](#_Toc119337561)

[Table 3.2: Feature Comparison of your Game with Existing Games 5](#_Toc119337562)

[Table 7.1 Submit Button Test case 15](#_Toc119337563)

# 

# GAME OVERVIEW

## Game Summary

In the enchanting 'Python Coding Forest,' two young adventurers, a courageous boy and an adventurous girl, find themselves on an unexpected quest. While exploring the depths of the forest, they encounter a mysterious group of kidnappers who, in a mischievous twist, separate the duo. The boy awakens to the sight of a looming tower, where his friend is held captive behind five intricately locked doors.

Determined to rescue his companion, the boy embarks on a coding odyssey through the tower's levels. Each door presents a unique coding challenge, symbolizing a stage of the journey. The first level unfolds as the boy discovers a chest in the forest, waiting to be unlocked. To open it, he must collect the code snippet ‘Print("Hello world")' scattered throughout the environment. With a drag-and-drop mechanism, the boy assembles the code, unveiling the key to the initial door.

As the boy advances through the tower's levels, more coding challenges and exciting adventures await. 'Python Coding Forest' weaves the magic of storytelling with the thrill of coding, turning each level into a step closer to rescuing the girl and unraveling the mysteries of the enchanted forest

## Target Platform(s)

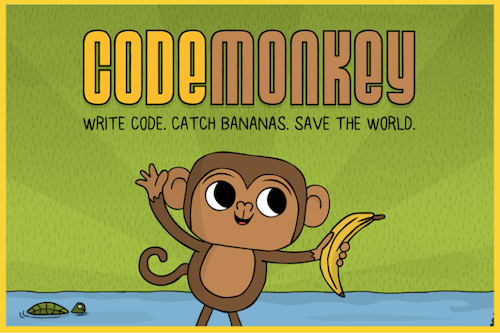
* Modern iOS and Android tablets and phones.
* The game will be in Landscape mode.

## Business Model

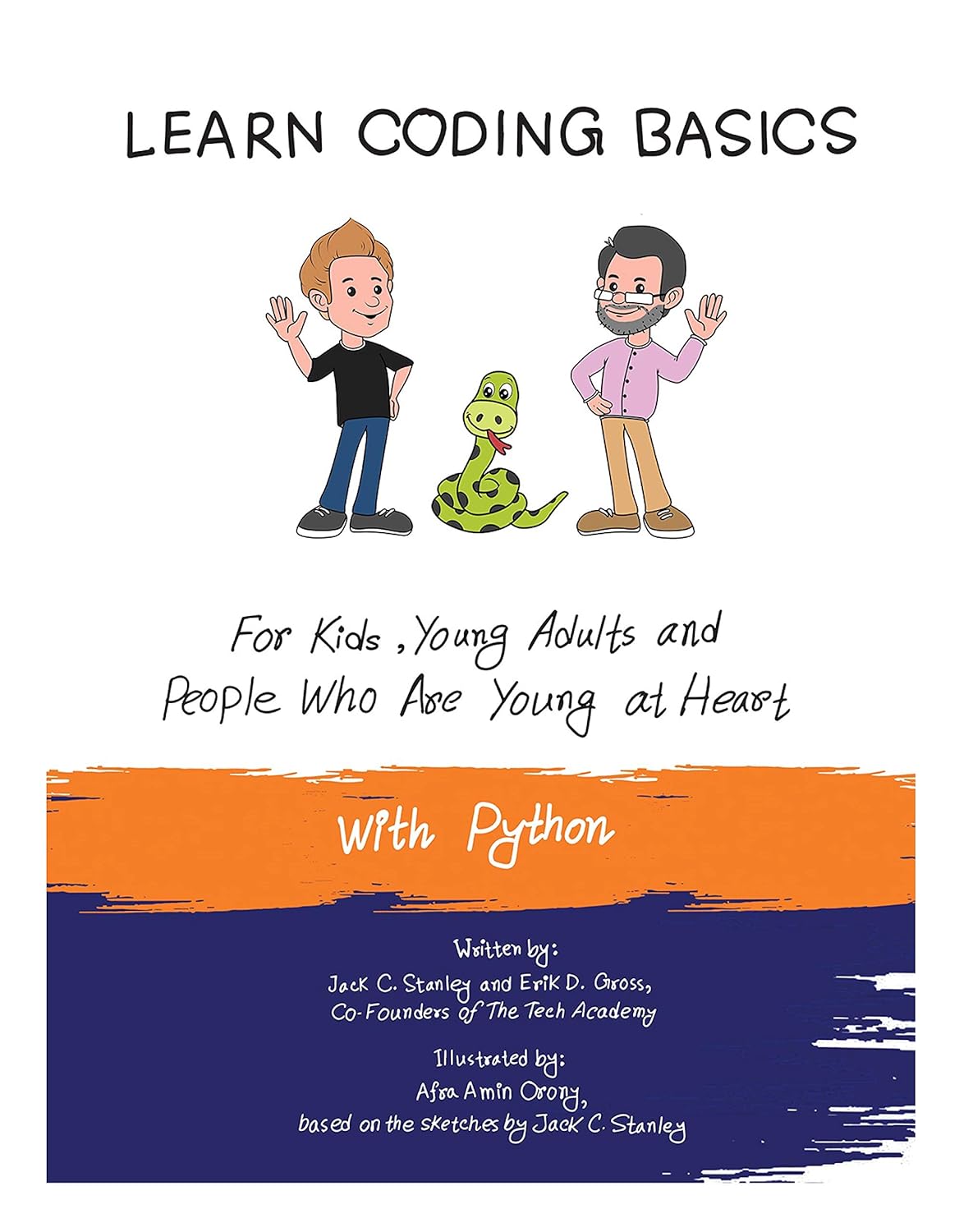
This will be a free to play game with micro-transactions and rewarded video ads.

## Theme / Setting / Genre

**Code Monkey (game)**



**Learn Coding Basics(Book)**



**Coding for Kids (Book)**



## Core Gameplay Mechanics

**Controls**

* Joystick

To move Character around

* Tap

To collect objects and see them

* Drag

To drop that in the chest box bar

* Other

There can be multiple other mechanics too based on the level as each level will has its own purpose and brain storming behind it as next level(level2) is like connecting Dots game which will demonstrate the idea on initializing variables.

## Server / Online Mechanics

Progress will be saved on user device.

## Art Style

* 3D
* Cartoon
* Low Poly

## Look & Feel/ Camera

* Third Person

Most used TPP techniques camera behind the user and it will follow it and for every level it can vary.

## Progression

* Level Based

Each Level Complete will get the user close to saving the girl that was stuck on tower and after completing 5 levels he will be able to free her. Each level will contains its own complexities so the kid can learn basic coding concepts from it,

# Story and Gameplay

## Story

Once upon a time, in the heart of the enchanting 'Python Coding Forest,' a spirited boy and an adventurous girl embarked on a whimsical journey of exploration. Their idyllic quest took an unexpected turn when a mysterious group of kidnappers emerged, disrupting the serenity of the forest and separating the inseparable duo. As the boy regained consciousness, he found himself standing before a towering structure—an enigmatic tower with five imposing doors, each bearing the weight of an unknown code that held the key to reuniting him with his friend.

Thus began the captivating tale of 'Python Coding Playground.' Fueled by an unwavering determination to rescue his companion, our intrepid hero set forth on a mesmerizing coding adventure through the levels of the tower. Each door presented a unique coding challenge, a puzzle that must be solved to progress further into the heart of the forest. The journey was not merely a quest for physical rescue but a transformative odyssey, where coding became the language that unraveled the secrets of the enchanted forest.

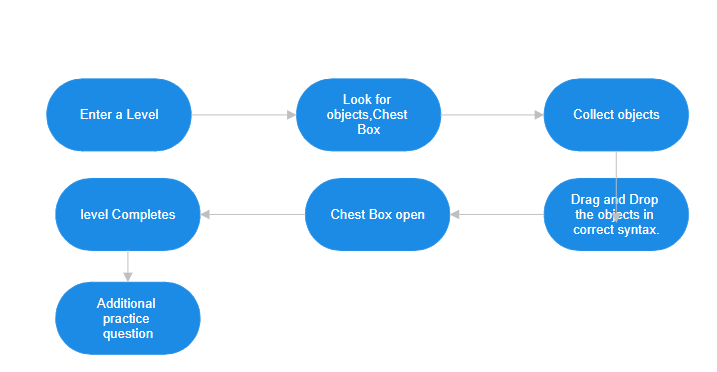
Through each level, the boy faced not only coding challenges but also encountered mystical creatures, uncovered hidden wisdom, and inched closer to a reunion with his friend. 'Python Coding Playground' became a story of friendship, courage, and the enthralling pursuit of knowledge, set against the backdrop of a fantastical world where coding became the key to unlocking doors and uniting kindred spirits. The forest whispered its secrets, and the code illuminated the path, creating a narrative where the magic of coding intertwined with the enchantment of the Python Coding Forest.

**Cut Scenes**

There will be small animation in the start like cut scene to describe the whole story animation can be like comics like picture with sound or actual animation based on the time and resources we get.

## Core Gameplay

There will be no the exact same loop throughout the game as it’s like mini games within game like each level is different with its own challenges and coding learning concepts.  
This flow chart is for level 1 Currently



# Domain Analysis

## Customer

## Stakeholders

Table 3.1: List of stakeholders

|  |  |
| --- | --- |
| **Stakeholder** | **Role in System** |
| Fahad Hassan | Main programmer and Designer |
| Hassam Hameed | Designer, Ui/Ux Artist and programmer |
| Ahmad Rumail | Designer |
| Abdul Basit | AI tools, Sounds, Story Animations |
| Waleed Waheed Butt | Art and Assets for game |

Table 3.1 shows the list of stakeholder involved in Python Coding Playground game.

## Affected Groups with social or economic impact

**1. Children (Aged 5-10):**

The primary beneficiaries of the 'Python Coding Playground' are young children aged 5-10, experiencing an interactive and engaging introduction to coding, fostering early computational skills and logical thinking.

**2. Parents and Educators:**

Parents and educators are indirectly impacted as the game encourages parental involvement in children's learning and promotes the integration of innovative, technology-driven educational tools in both home and classroom environments.

**3. Game Developers and Educators:**

Professionals in game development and education are influenced by the project, inspiring creativity and innovation in the design of educational games, paving the way for novel approaches to teaching coding concepts.

## Dependencies/ External Systems

* Python Programming Language
* Interactive Learning Platforms
* User Interface (UI) and User Experience (UX) Design Tools
* Mobile device
* Art(Blender)
* AI tools for voice and animation

## References

Code Monkey (java script)

### Related Projects

List of all the documents/ projects that you have looked up as reference material for this project along with their links/references. e.g.

In order to develop First Person Shooter horror style game, we looked up several similar systems. Their details are given below

1. Dead Trigger 2 Zombie shooter

“Fight for survival in a zombie apocalypse with this heart-stopping first person shooter. Choose from an insane arsenal of combat weapons and battle against the walking dead in terrifying environments around the globe.”

### Feature Comparison

Table 3.2: Feature Comparison of your Game with Existing Games

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sr No. | Comparison Feature | Game 1 | Game 2 | Game 3 | Your Game |
| 1 | ABC | Game 1 covers the feature ABC completely as desired | Game 2 does not support feature ABC | Game 3 suggests that maximum efficiency can be achieved if ABC is implemented using algorithm ABC. | Using the ABC feature from and improving it with ABC algorithm can provide maximum efficiency |

**Explain your table and refer table caption in your text**

e.g.

In above Table 3.2 we have compared our game with existing games. This feature comparison shows our game includes new features.

# Requirements analysis

## Requirements

*This section is can be skipped, if Requirement Specifications document has been developed for the project. Otherwise this section is mandatory.*

*This section may contain*

*End user, operator, support, or integration functions,*

*Performance requirements,*

*Design constraints,*

*Programming language, and*

*Interface requirements.*

*System functions are descriptions of what a system is supposed to do. They should be identified and listed in logical cohesive groups, with their category (priority) assigned. These system functions will be identified as a result of the requirement gathering process conducted with the client. However, in some cases, prior to the development of the Functional Specifications the requirements may already have been listed in a document: if this is so then a reference to the document may suffice.*

*To verify that some* ***X*** *is indeed a system function; it should make sense in the following sentence:*

*The system should do <****X****>*

*The table below gives an example of how system functions can be listed:*

*The Functions column gives a brief one-line description of the required functionality.*

*The Category column refers to the status of the functionality for the proposed system. The options for the Category are defined below.*

*The Attribute column defines the system characteristics. The Details and Constraints column specifies the conditions within which the attribute is applicable. Section 1.12 defines the default Attributes and the related Constraints. In case, the default conditions are to be over-ridden then the conditions can be defined in this table.*

*Function Categories*

|  |  |
| --- | --- |
| ***Functional Requirements*** | ***The services requested by the user*** |
| *Non-Functional Requirements* | *The supporting requirements for functional requirements. These include the* ***measureable*** *quality attribute.* |
| *Data Requirements* | *How your data will be stored* |
| *Constraints* | *by the client On your system* |
| *External interface requirements* | *How will your system connect to other software/components* |

**Explain your table and refer table caption in your text**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *RID* | *description* | *Category* | *Attribute* | *Details & Boundary Constraints* |
| *NFR1* | *Record the underway sale – the items purchased* | *non-functional* | *System Response time* | *Price listing within 3 seconds*  *Availability agreement in less than 10 sec* |
| *NFR2* | *Reduce inventory quantities when a sale is committed* | *non-functional* | *Concurrent user load* |  |

**All Functional and Non-Functional requirements shall be proper numbered e.g. FR1 for**

**functional requirement 1 and NFR1 for non-functional requirement. Non- Functional**

**requirements are constraints.**

## List of Actors

Define the system boundary and list all actors with the use cases. **All the actors must also be mentioned in your list of stakeholders**

For example:

Cashier: this person performs all the financial activities

Account Manager: this person supervises all financial activities

## List of use cases

List all the use cases, with a brief description (should not exceed two lines):

Buy Item: captures a sale and its payment

Log In: allow user to provide account information and access the restricted services

## System Use Case diagram

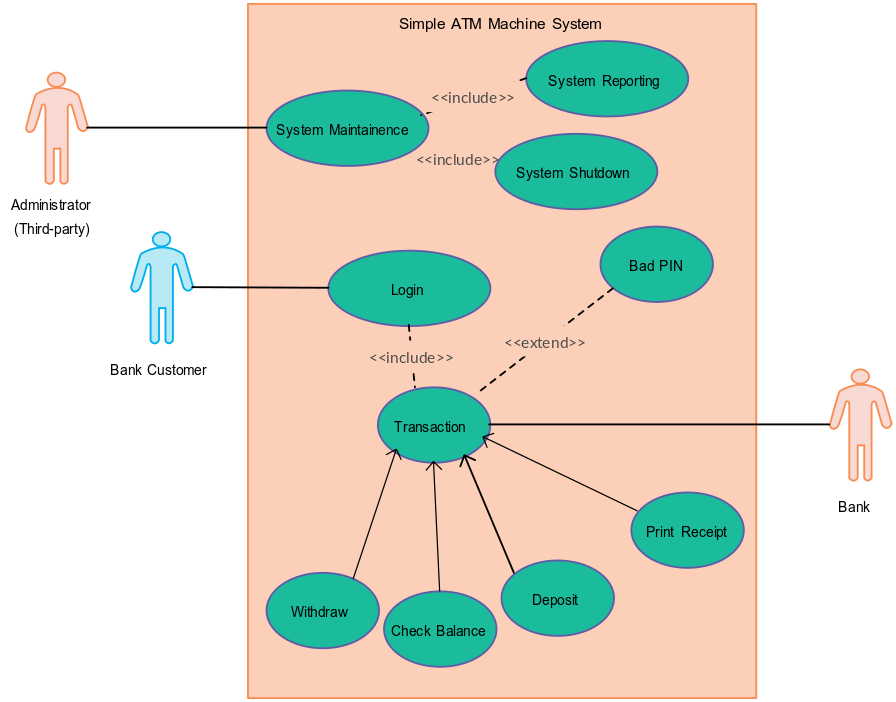


Figure 4.1: Use case diagram of ATM

**Explain your figures and refer figure caption in your text**

e.g.

In above Figure 3.1 we have three actor’s administrator and customer and bank. In Figure 4.1 we have identified the uses cases of administrator, customer and bank.

## Extended use cases

Every use case form the list must be elaborated here. e.g.

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID:** | Enter a unique numeric identifier for the Use Case. e.g. UC-1.2.1 | | |
| **Use Case Name:** | Enter a short name for the Use Case using an active verb phrase. e.g. Withdraw Cash | | |
| **Created By:** |  | **Last Updated By:** |  |
| **Date Created:** |  | **Last Revision Date:** |  |
| **Actors:** | [An actor is a person or other entity external to the software system being specified who interacts with the system and performs use cases to accomplish tasks. Different actors often correspond to different user classes, or roles, identified from the customer community that will use the product. Name the actor that will be initiating this use case (primary) and any other actors who will participate in completing the use case (secondary).] | | |
| **Description:** | [Provide a brief description of the reason for and outcome of this use case.] | | |
| **Trigger:** | [Identify the event that initiates the use case. This could be an external business event or system event that causes the use case to begin, or it could be the first step in the normal flow.] | | |
| **Preconditions:** | [List any activities that must take place, or any conditions that must be true, before the use case can be started. Number each pre-condition. e.g.   1. Customer has active deposit account with ATM privileges 2. Customer has an activated ATM card.] | | |
| **Post conditions:** | [Describe the state of the system at the conclusion of the use case execution. Should include both *minimal guarantees* (what must happen even if the actor’s goal is not achieved) and the *success guarantees* (what happens when the actor’s goal is achieved. Number each post-condition. e.g.   1. Customer receives cash 2. Customer account balance is reduced by the amount of the withdrawal and transaction fees] | | |
| **Normal Flow:** | [Provide a detailed description of the user actions and system responses that will take place during execution of the use case under **normal, expected** conditions. This dialog sequence will ultimately lead to accomplishing the goal stated in the use case name and description.   1. Customer inserts ATM card 2. Customer enters PIN 3. System prompts customer to enter language performance English or Spanish 4. System validates if customer is in the bank network 5. System prompts user to select transaction type 6. Customer selects Withdrawal From Checking 7. System prompts user to enter withdrawal amount 8. … 9. System ejects ATM card] | | |
| **Alternative Flows:**  **[Alternative Flow 1 – Not in Network]** | [Document **legitimate** branches from the main flow to handle special conditions (also known as extensions). For each alternative flow reference the branching step number of the normal flow and the condition which must be true in order for this extension to be executed. e.g. Alternative flows in the *Withdraw Cash* transaction:  4a. In step 4 of the normal flow, if the customer is not in the bank network   1. System will prompt customer to accept network fee 2. Customer accepts 3. Use Case resumes on step 5   4b. In step 4 of the normal flow, if the customer is not in the bank network   1. System will prompt customer to accept network fee 2. Customer declines 3. Transaction is terminated 4. Use Case resumes on step 9 of normal flow   Note: Insert a new row for each distinctive alternative flow. ] | | |
| **Exceptions:** | [Describe any anticipated **error conditions** that could occur during execution of the use case, and define how the system is to respond to those conditions.  e.g. Exceptions to the Withdraw Case transaction  2a. In step 2 of the normal flow, if the customer enters and invalid PIN   1. Transaction is disapproved 2. Message to customer to re-enter PIN 3. Customer enters correct PIN 4. Use Case resumes on step 3 of normal flow] | | |
| **Includes:** | [List any other use cases that are included (“called”) by this use case. Common functionality that appears in multiple use cases can be split out into a separate use case that is included by the ones that need that common functionality. e.g. steps 1-4 in the normal flow would be required for all types of ATM transactions- a Use Case could be written for these steps and “included” in all ATM Use Cases.] | | |
| **Frequency of Use:** | [How often will this Use Case be executed. This information is primarily useful for designers. e.g. enter values such as 50 per hour, 200 per day, once a week, once a year, on demand etc.] | | |
| **Special Requirements:** | [Identify any additional requirements, such as nonfunctional requirements, for the use case that may need to be addressed during design or implementation. These may include performance requirements or other quality attributes.] | | |
| **Assumptions:** | [List any assumptions that were made in the analysis that led to accepting this use case into the product description and writing the use case description.  e.g. For the *Withdraw Cash* Use Case, an assumption could be:  The Bank Customer understands either English or Spanish language.] | | |
| **Notes and Issues:** | [List any additional comments about this use case or any remaining open issues or TBDs (To Be Determined) that must be resolved. e.g.   1. What is the maximum size of the that a use can have?] | | |

**Explain your table and refer table caption in your text**

# User Interface / Screens

***Wireframes*** of every screen or popup in the game and clear info on how they connect and lead to each other.

Numbering these sections makes them easy to find and you may even want to make a master flow-chart with ALL screens showing how each connects to the other.

## Main Menu

* Game Logo
* “Play Now” Button - Goes to 2. Gameplay Screen
* “Store” - Goes to 3. Premium Currency Store
* “Settings”
* “Credits”

## Gameplay Screen

123

+

Settings (See “6. Settings” Popup)

Gear Shop (See “5. Shop” Screen)

Coin (Soft Currency) Balance

Coin Store (See “4. Coin Store” Screen)

Level / Progress to Next Level

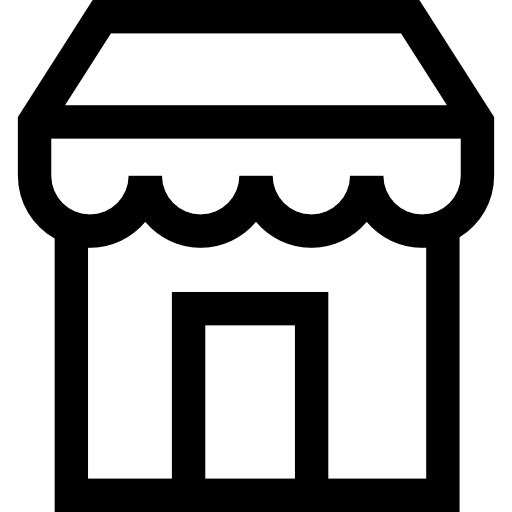
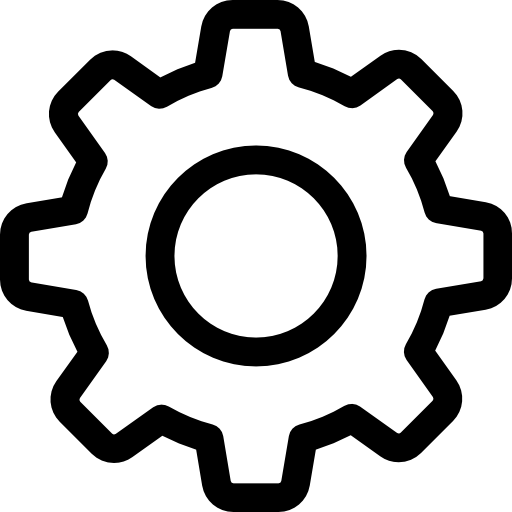
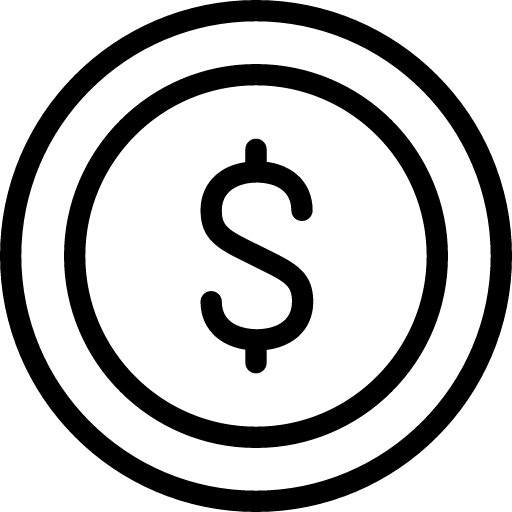
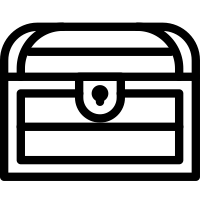
Player Hit Points

Enemy Hit Points

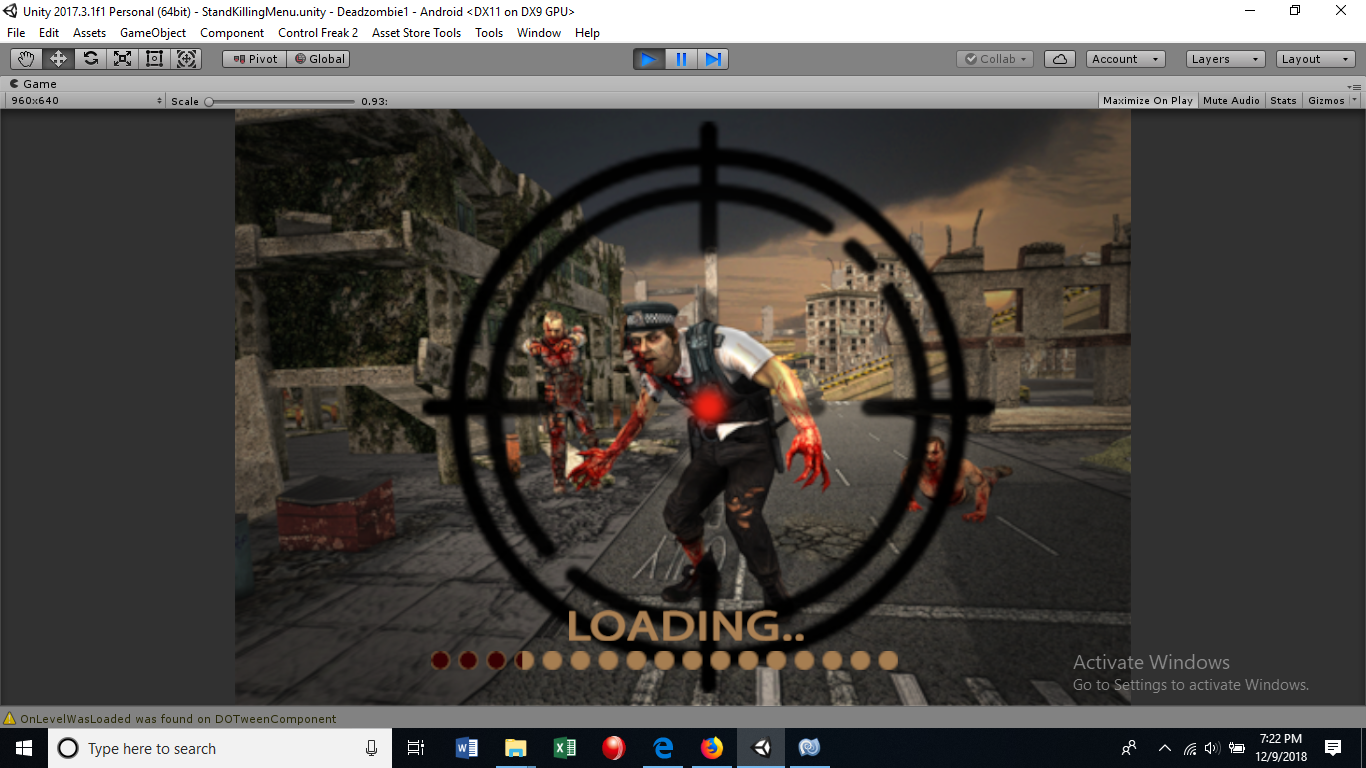
Open Daily Reward Chest

(See “7. Daily Reward” Popup)

5



**Explain your UI and refer UI caption in your text**



Prototype1: (P1) loading

Prototype1 :(P1) is showing user interface loading screen of FPS zombie shooting game

# Levels

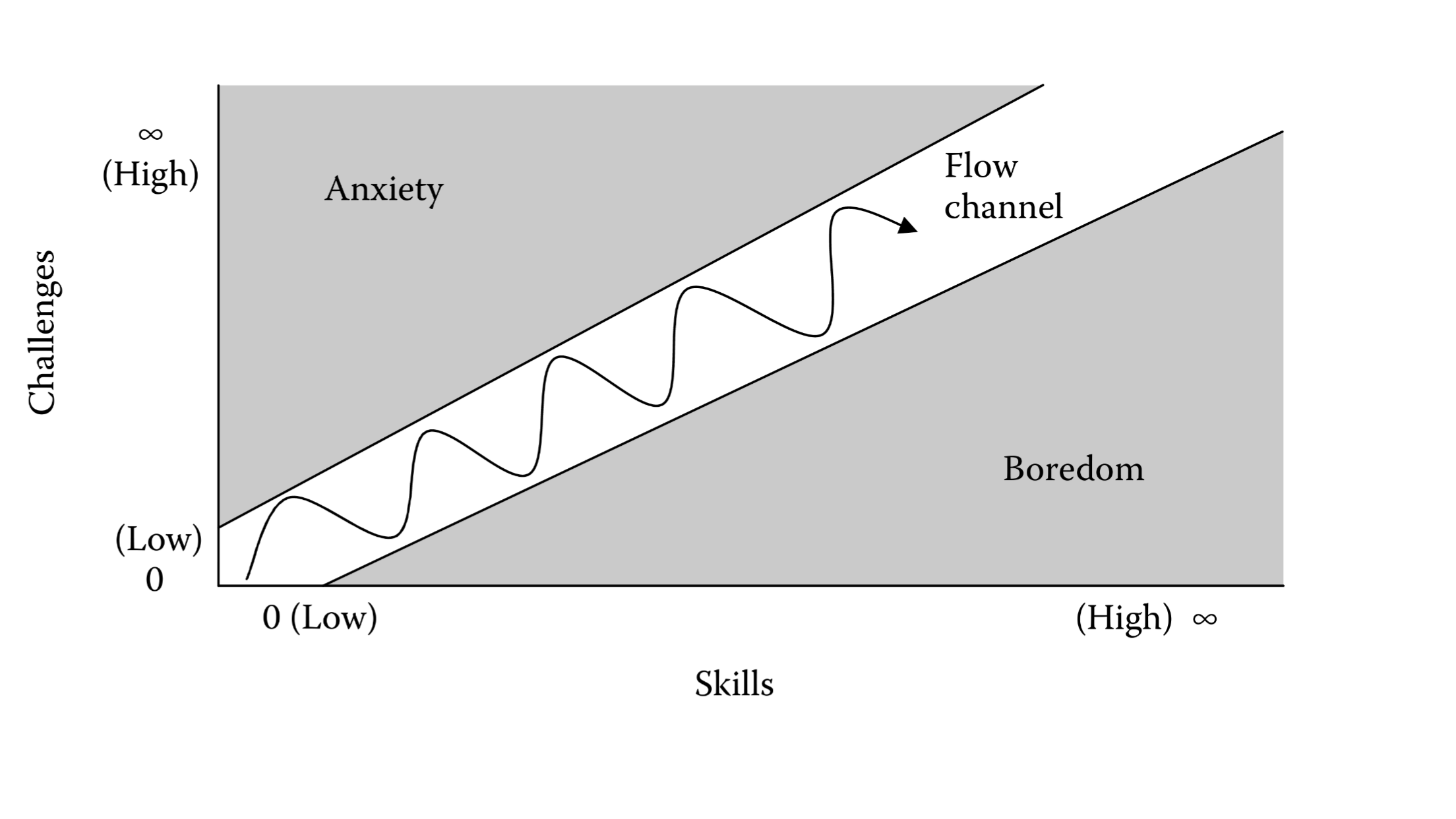
Details about how many levels in your game?

How levels will vary?

An example of a beginner level.

An example of a complex level.

A graph of the level “curve” which is usually starting easy as the player learns the game, getting a bit more difficult until the player faces off against a nerve-wracking “boss fight” or “challenge gate”, then sawtooth down to a bit easier again to let players relax.



## Tool for Level Designing

Tools you will use for level designing?

## First Time Experience

Details of how a new player learns your game step by step elaborate with screen shots.

# Asset List

* Art
  + Note if 2D or 3D
  + If 3D Details on Textures and Poly Count
* Environment Art
  + All Associated Animations
* UI - Icons, Buttons, Pop Ups
* Game Objects
  + Buildings
  + Characters
  + Weapons
  + Tools
  + All Associated Animations
* Particle FX
* Sound Effects
* Music
* All Writing
  + Quest Scripts
  + Story ‘Screenplay’
* Level Design Documents

A big spreadsheet with columns and sections for each screen / major area is helpful:

Figure 7.1: Asset list of Dead Zombie FPS game

|  |  |  |
| --- | --- | --- |
| **Category** | **WORLD MAP** | **Description** |
| Full Screen | Background | Background image that covers the whole scene |
| Icon | City 1 Icon | New York style |
| Icon | City 2 Icon | San Francisco style |
| Icon | City 3 Icon | Paris style |
| Sprite | Locked City | Big lock |
| Icon | Coming Soon | A banner that says ‘Coming Soon’ |
|  |  |  |
| **Category** | **ENVIRONMENT ASSETS** | **Description** |
| **Full Screen** | **Sky Background** | **2D image of sky** |
| **3D** | **Rooftop** | **Roof shingles with chimneys, antenna, etc.** |

# Implementation details

## Development Setup

List your tools and technologies and their role in development.

## Deployment setup

How and where was your software deployed? Did you face any problems, how did you overcome these problems?

## Algorithms

Entire code of software is not required. Just highlight your important (user defined/ improved) algorithms.

**Add at least 2-3 Code snippets of your Gaming Project here and then provide the link of your Github repository**

## Constraints

### Assumptions

Things we assume will be true.

e.g.:

* *We will receive all necessary technical support from the engineers at cMeRun, Select and Mellon Bank to help design the interfaces between their systems and enGyro.*
* *All database maintenance will be handled by the client.*
* *There will be no real-time interfacing with any accounting systems.*

### System constraints

 A constraint specifies how the system must operate or how it must be built

### Restrictions

Constraints applied on the system by the client

### Limitations

Services your software is unable to provide

# Testing

## Extended Test Cases

Table 9.1 Submit Button Test case

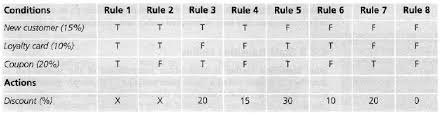
## 

Explain your table and refer table caption in your text

## Decision Table

### Code snippet

### Decision coverage table



## Traceability Matrix

### RID vs UCID (requirements vs use cases)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **UCID/RID** | **R**  **1** | **R**  **2** | **R**  **3** | **R**  **4** | **R**  **5** | **R**  **6** | **R**  **7** | **R**  **8** | **R**  **9** | **R**  **10** | **R**  **11** | **R**  **12** | **R**  **13** | **R**  **14** | **R**  **15** | **R**  **16** | **R**  **17** | **R**  **18** | **R**  **19** | **R**  **20** | **R**  **21** |
| UC 1 | ✓ | ✓ |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| UC 2 |  | ✓ |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| UC 3 | ✓ | ✓ |  |  |  |  |  |  |  |  |  |  | ✓ |  |  |  |  |  |  |  |  |
| UC 4 | ✓ | ✓ |  |  |  |  |  |  |  |  |  |  |  | ✓ |  |  |  |  |  |  |  |
| UC 5 | ✓ | ✓ | ✓ |  | ✓ |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| UC 6 | ✓ | ✓ |  | ✓ |  | ✓ |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| UC 7 | ✓ | ✓ | ✓ |  |  |  |  |  | ✓ |  |  |  |  |  |  |  |  |  |  |  |  |
| UC 8 | ✓ | ✓ |  | ✓ |  |  |  |  |  | ✓ |  |  |  |  |  |  |  |  |  |  |  |
| UC 9 | ✓ | ✓ | ✓ |  | ✓ |  |  |  |  |  | ✓ |  |  |  |  |  |  |  |  |  |  |
| UC 10 | ✓ | ✓ |  | ✓ |  | ✓ |  |  |  |  |  | ✓ |  |  |  |  |  |  |  |  |  |
| UC 11 | ✓ | ✓ | ✓ |  | ✓ |  | ✓ |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| UC 12 | ✓ | ✓ |  | ✓ |  | ✓ |  | ✓ |  |  |  |  |  |  |  |  |  |  |  |  |  |
| UC 19 | ✓ | ✓ | ✓ |  | ✓ |  |  |  |  |  |  |  |  |  | ✓ |  |  |  |  |  |  |
| UC 20 | ✓ | ✓ |  | ✓ |  | ✓ |  |  |  |  |  |  |  |  |  | ✓ |  |  |  |  |  |
| UC 21 | ✓ | ✓ | ✓ |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| UC 22 | ✓ | ✓ |  | ✓ |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| UC 23 | ✓ | ✓ | ✓ |  |  |  |  |  |  |  |  |  |  |  |  |  | ✓ |  |  |  |  |
| UC 24 | ✓ | ✓ |  | ✓ |  |  |  |  |  |  |  |  |  |  |  |  |  | ✓ |  |  |  |
| UC 25 | ✓ | ✓ | ✓ | ✓ |  |  |  |  |  |  |  |  |  |  |  |  |  |  | ✓ |  |  |
| UC 26 | ✓ | ✓ | ✓ | ✓ |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | ✓ |  |
| UC 27 | ✓ | ✓ | ✓ | ✓ |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | ✓ |

### Prototypes (RID vs PID)

### Test Cases (RID vs TID)

### Coverage (UCID vs TID)

# Results/Output/Statistics

## %completion

Use the matrix & values from 7.3.1 to show that all requirements are being fulfilled.

e.g.

Fulfilled requirements: 74

Total requirements: 94

%completion = (74/94)\*100 = 78.723%

## %accuracy

Use the matrix & values from 7.3.3 to show that all requirements have been implemented correctly.

e.g.

Implemented requirements: 69

Total requirements: 94

%accuracy = (69/94)\*100 = 73.40%

## %correctness

Use the matrix & values from 7.3.4 to show that all requirements have been tested to be conforming to requirements.

e.g.

Tested requirements: 79

Total requirements: 94

%accuracy = (79/94)\*100 = 84.04%

# Conclusion

# Future work

# Bibliography

***Use IEEE or ACM format for citations***

## Books

## Journals

## Articles

## Research papers

e.g.

ur Rehman, Khawaja Ubaid, Waqas Ali, Shahid M. Awan, M. Rehan Saleem, and M. Adeel Ashraf. "Impact of Android Applications on Learning of Kindergarten Students." (2020).

## Other References

# Appendix

## Glossary of terms

Make a table of terms and write its explanation. For each technical term used in your project write its explanation.

## Pre-requisites

Must use contents of development/ deployment setup & external system dependencies

e.g.

Requirements (Minimum):

Android Baseband & Kernel V7.1, 2-4 GB RAM, Memory 300 MB.