Software Requirement Specification

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Manika	8 Sept 2021	Initial Draft	1.0
Justina	10 Sept 2021	Re-assess Quality Attributes	1.1
Jeremy	15 Oct 2021	Add diagrams	1.2
Colin	11 Nov 2021	Final Checks	1.3

1. Introduction

1.1 Purpose

This document lays out the structure of the development of “Castle Sky”, a game application by (group name). The purpose of this document is to provide a detailed overview of our software product, its parameters and goals. The intended readers of this document are for current and future developers to assist in the software delivery lifecycle (SDLC) processes of “Castle Sky”. This plan includes, but is not limited to, a summary of the system functionality and features, functional and non-functional requirements of the system and various analysis models.

1.2 Document Conventions

Under this SRS, the font format will be fixed to tool “Confluence” provided, bold text indicated requirement with higher priority. Inherited requirements will be indicated through a list.

The naming conventions that we will be using in this document are as follows:

Term	Definition
NPC	Non-playable character
TBD	A placeholder to indicate when necessary information is not yet available
Worlds	Each game world represents a teaching subject, i.e. Science, Math, English.
Topics	Within each game world, there will be multiple topics for students to access. Topics will be related to the subject of the game world, i.e Math World will have topics like Whole Numbers, Fractions.
DB	Database
API	Application Programming Interface

1.3 Intended Audience and Reading Suggestions

This project is a prototype for educational gaming purposes and is restricted to primary school usage. This document is intended for any developer, documentation writers, software testers or user involved in the design, development and testing of “CastleSky”. The sequence of the rest of this document is as follows:

1. Overall Description
2. External Interface Requirements
3. System Features
4. Other Non-Functional Requirements
5. Other Requirements
6. Appendix

Developers who are to review application capabilities, system features and functional or non-functional requirements can view sections 1, 3, 4, 5. Those who want to better visualize the application can view the basic insights of our interface under section 2.

1.4 Product Scope

“Castle Sky” is a social application that is aimed to facilitate learning in the form of gamification. It is being developed for both students and teachers in all primary schools. With this application, we aim to create a more interactive approach for teachers to conduct lessons and assessments, at the same time teachers are able to track students' real-time results to plan more effective lessons in class. Through our technology?? used, students will have targeted questions to complete in the mini-games based on their individual mastery skills, and are able to challenge their fellow schoolmates to be the top in the leaderboard, thus creating a more engaging experience. The larger goal of this project is to implement “Castle Sky” to all educational institutions in Singapore.

1.5 References

Listed below are the documents referred to in this document

Use Case Description Document

Design Document

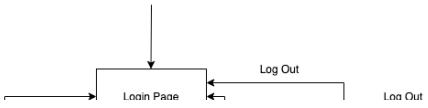
2. Overall Description

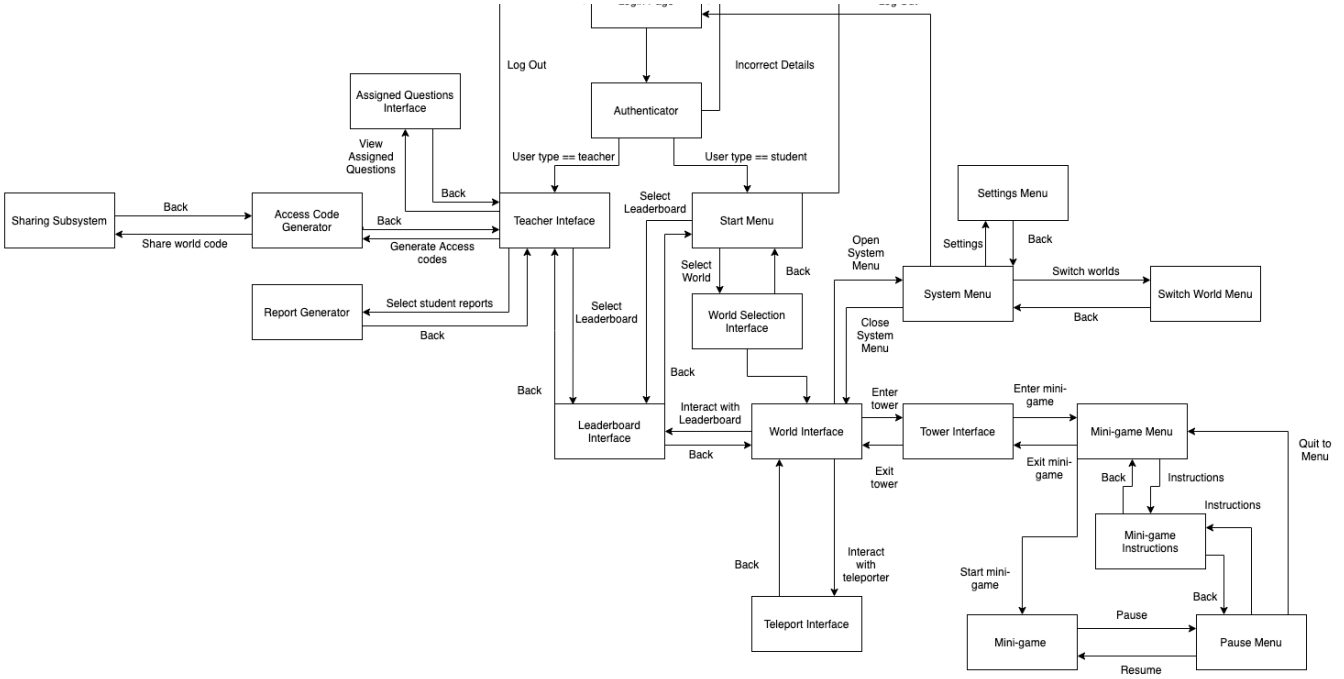
2.1 Product Perspective

CastleSky is a new, self-contained product. It consists of various world maps, and within each world map, there are mini-games to be played.

2.1.1 Dialog Map

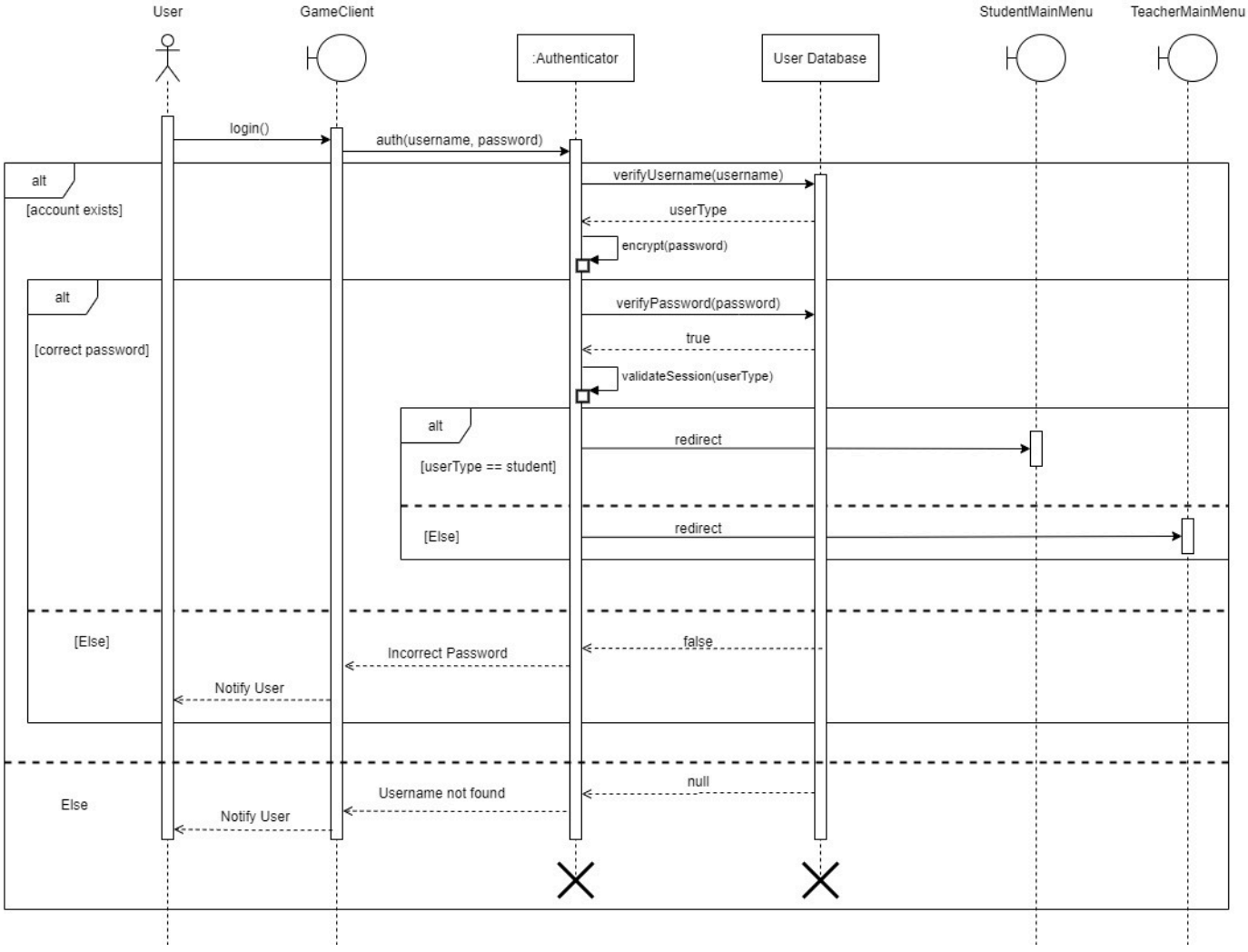
A dialog map is presented below, which illustrates the user interactions with the various CastleSky components, shown at a very high level.





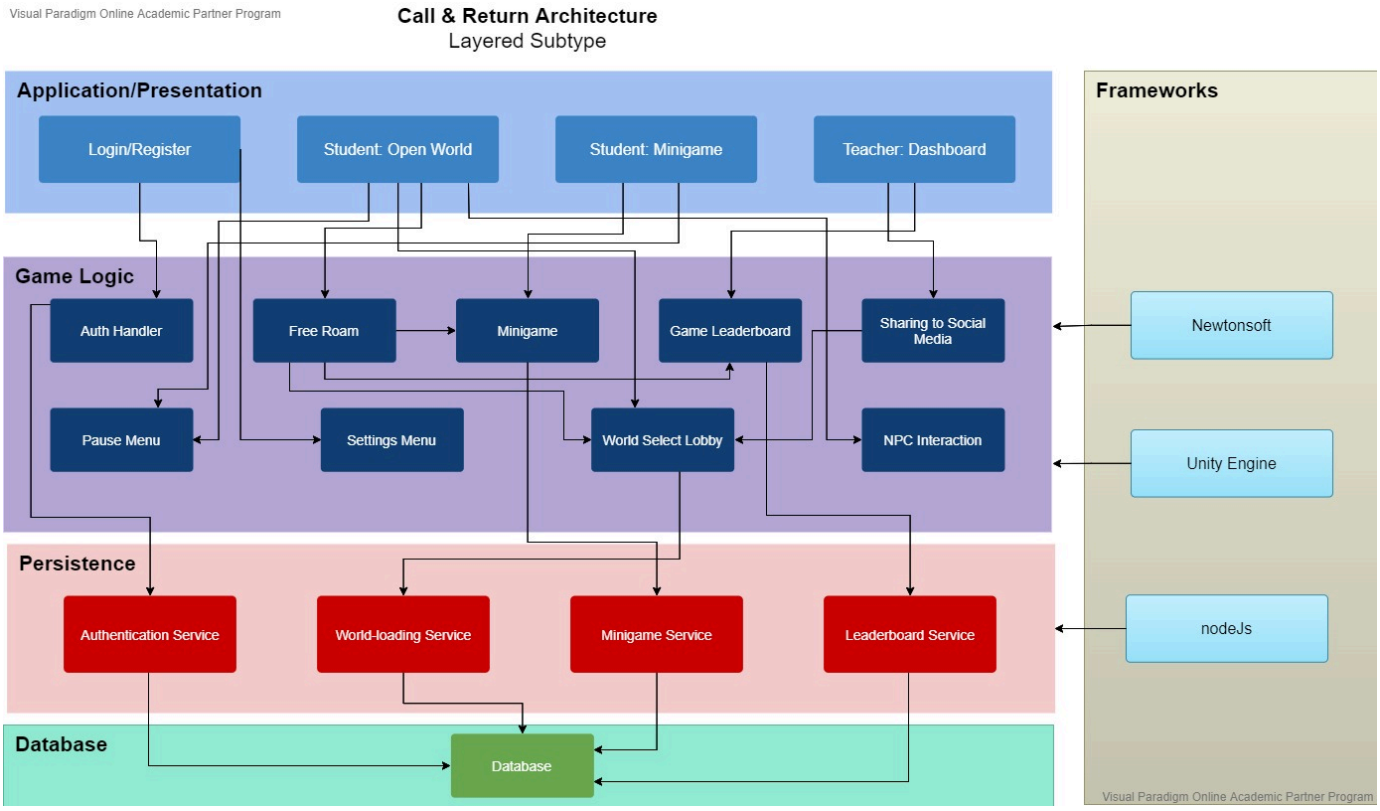
2.1.2 Sequence Diagram

Below is a sequence diagram highlighting the processes and methods used when user authentication is performed.



2.1.3 Architecture Diagram

The architecture diagram below illustrates the Layered Architectural style we adopted for CastleSky development. For a more detailed analysis, please refer to our Design Document.



2.2 Product Functions

Our Product Functions will be split into 3 different segments.
Students, who represent the players who will be playing this game to assess their knowledge on different subjects and topics.
Teachers, who represent the administrators for the game, with several administrative functions to control the students' access in the game.
System, which represents the generic functions that the system should provide to the users.

Students	**CastleSky will allow students to navigate their character in the World**
	CastleSky will allow students to navigate to different Worlds
	CastleSky will allow students to access Mini Games
	CastleSky will allow students to interact with NPCs
	CastleSky will allow students to view Educational Videos upon failing a level of Mini Game.
	CastleSky will allow students to re-attempt the final level of Mini Game to achieve higher scores in the leaderboard system.
	CastleSky will allow students to view the leaderboard.
Teachers	CastleSky will allow teachers to view list of accessible topics (for students)
	CastleSky will allow teachers to view previously assigned list of topics to students
	CastleSky will allow teachers to generate a summary report for each topic.
	CastleSky will allow teachers to assign work to students by generating access codes.
System	CastleSky will allow users to Pause Game.

	CastleSky will allow users to Resume Game.
	CastleSky will allow users to Restart Game.
	CastleSky will allow users to get help in a game.
	CastleSky will allow users to Exit Game.
	CastleSky will allow users to login account.
	CastleSky will allow users to register account.
	CastleSky will allow users to retrieve their Forgotten Password.

2.3 User Classes and Characteristics

As this is an educational product, our main user classes are students and teachers, with the main target audience being primary school students.

Primary school students learn about various topics in school, but usually face the standard mode of education. Students are unable to see how what they learn in school can be used in real life.

Teachers struggle to get the attention and focus of their students. While they have to follow a certain syllabus and curriculum set by the Ministry of Education and the school management, most teachers also wish to be able to inspire their students and teach them in different ways.

2.4 Operating Environment

The application will be able to run on any laptops or desktops with minimally these specifications:

Minimum System requirements

- **Memory:** 5 GB
- **Graphics Card:** NVIDIA GeForce 510
- **CPU:** Intel Core 2 Quad Q9000
- **File Size:** 300 Mb
- **OS:** Windows 10

2.5 Design and Implementation Constraints

Corporate or Regulatory Policies

The syllabus of subjects used in mini-games must be aligned with that implemented by the Ministry of Education (MOE).

Interfaces to other applications

- Backend API
  AWS Elastic Compute Cloud
- DB API
  AWS Relational Database Services
- Graphing API
  Codemonkey.Graphs

Technology Consideration

Technology Name	Description	Justification
Unity	Unity is real-time development platform which can be used to create 2D and 3D games.	Unity is a commonly used platform, thus it can be assumed that most Windows devices are able to support it.
Amazon Web Services EC2 Instance	Amazon Elastic Compute Cloud is a service that provides secure, resizable compute capacity in the cloud. It is designed to make cloud computing with high scalability and availability.	CastleSky backend (API Server) would likely be deployed via AWS EC2.
AWS Relational Database Service		

(RDS)	Amazon Relational Database Service is a database service that can be easy to set up, operate, and scale in the cloud. It provides a cost-effective and high scalable & available database for developers.	CastleSky requires a database for storage of user information as well as a students' attempts and scores. AWS RDS provides suits our project's requirements.

Language requirements

As the team will be using Unity to develop the game, the project will predominantly be using C# for the source code.

Security considerations

- Password encryption
- Access validation for student's information

User Documentation

- Website for installation & more about the game
- In-game tutorial for minigames

2.6 Assumptions and Dependencies

- Working internet connection
- Reasonably functioning computer machine
    - Free of major defects inhibiting ability to perform most functions

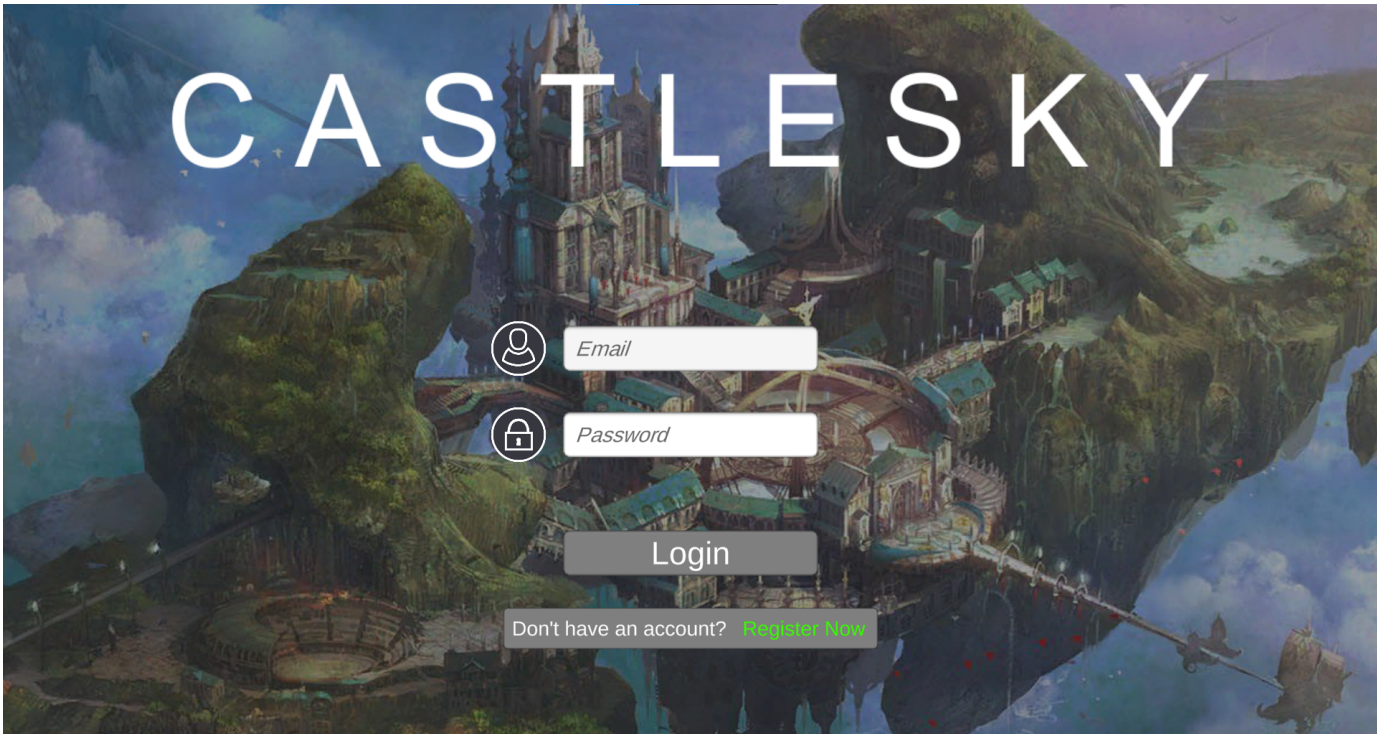
3. External Interface Requirements

For a more detailed exploration of the various external interfaces and subsystems implemented in the development of CastleSky, please refer to our Design Document.

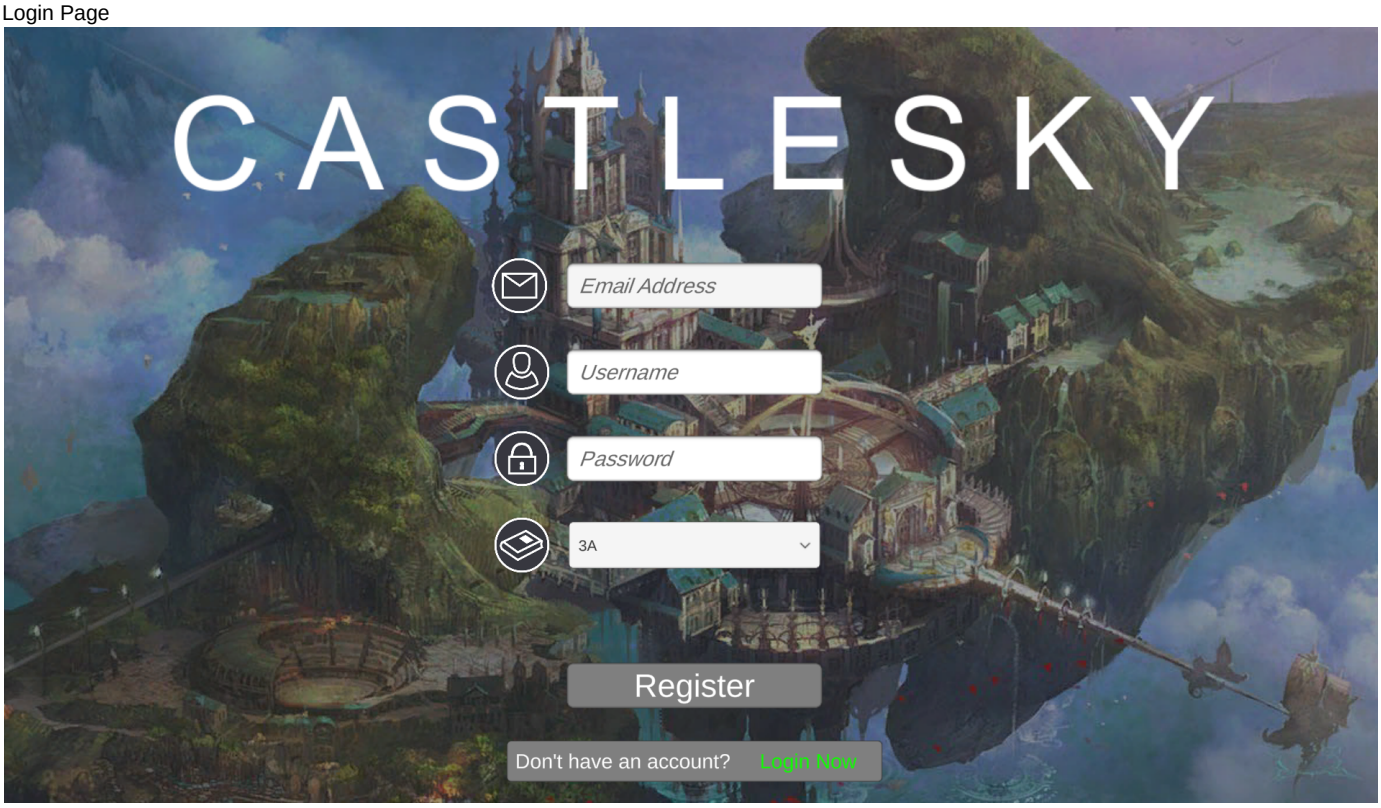
3.1 User Interfaces

Attached are some screenshots of the application, sectioned in System, Student and Teacher modes respectively.

3.1.1 System

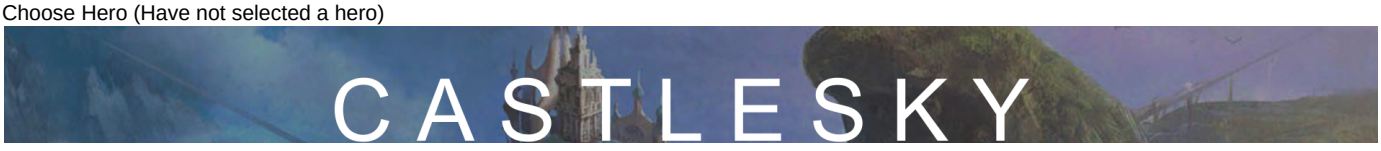


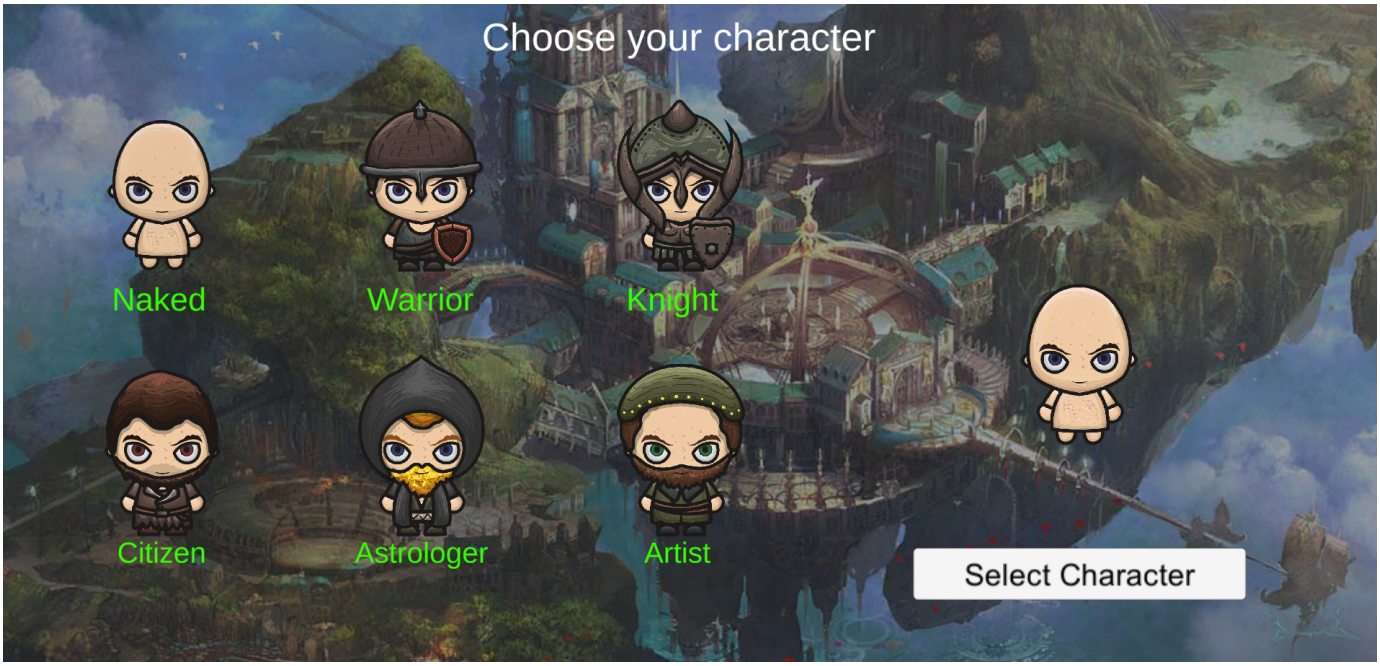
Login Page



Register Page

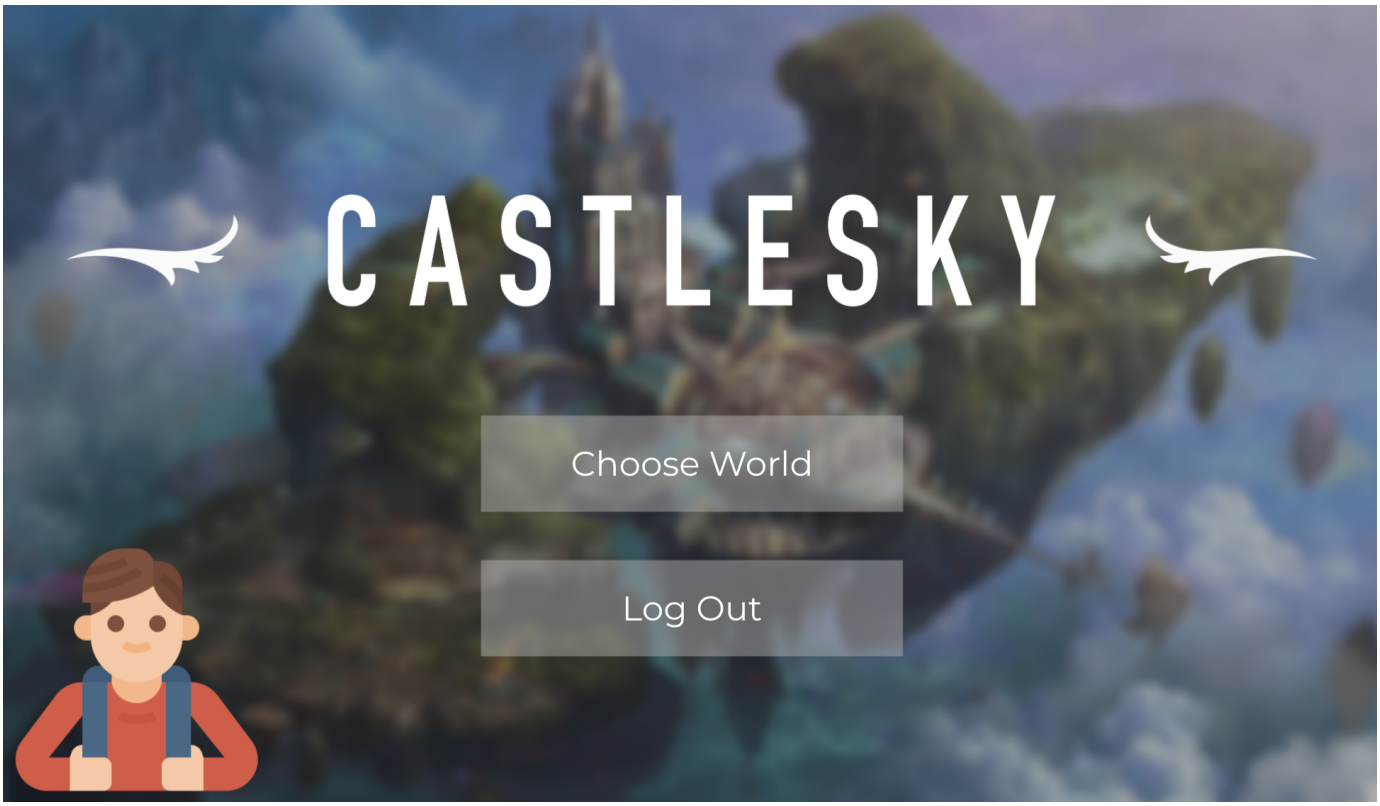
Choose Hero (Have not selected a hero)





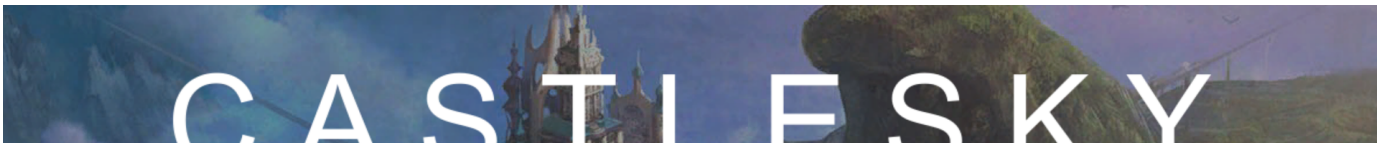
Choose Hero (Selected A Hero)

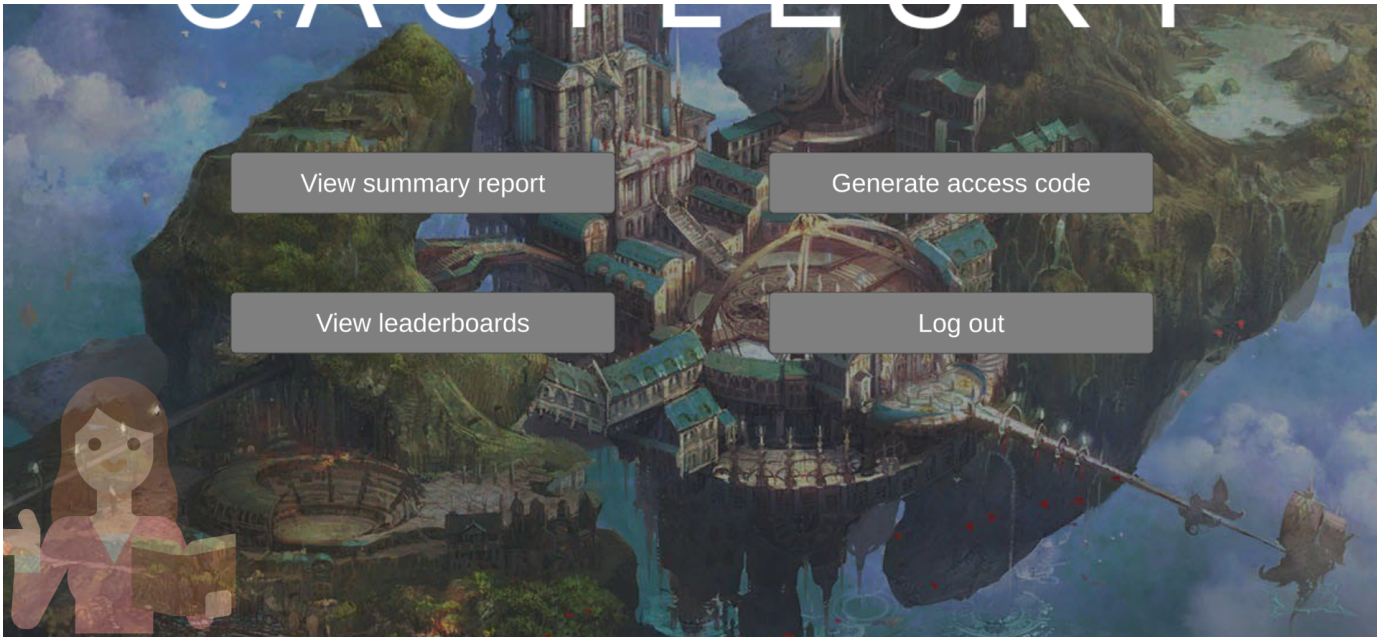
3.1.2 Student



Student Start Menu

3.1.3 Teacher





Teacher Start Menu

3.2 Software Interfaces

Please refer to the Design Document for a more detailed exploration of the various subsystems and interfaces being used in CastleSky's development

4. System Features

For a comprehensive list of features and their corresponding functional requirements, please refer to our Use Case Descriptions Document

5. Other Nonfunctional Requirements

5.1 Performance Requirements

The game must be interactive and delays in gameplay must be minimised. For every user action executed in the game such as navigating within one world, there must not have any immediate delays and be responsive. In the case of opening up menus such as the pause menus or popping of dialogue boxes, the delay must be below 2 seconds. In the case of teleporting across worlds and accessing mini-games, the loading operations must be performed within 3 seconds. In the case of connecting to the database and accessing the leaderboard, such a database request must respond within 1000 milliseconds.

5.2 Flexibility Requirements

The game design must maintain a high level of flexibility and extensibility. The game must allow for the easy addition of new features and capabilities such as new world and mini-games with minimal changes being made to the original source code. This is achieved by ensuring that our game design consists of a codebase that achieves low coupling and high cohesion.

5.3 Maintainability Requirements

The code written for the game must be maintainable. This is achieved by first ensuring that there are minimal dead codes embedded in the program and that there the code has proper comments and documentation for ease of developers to maintain the game cost-effectively over its expected lifetime. Specifically, the game must aim to have low cyclomatic complexity.