SSADungeon

AGENDA

01 PRODUCT FEATURES

02 USE CASE DIAGRAM

03 SYSTEM ARCHITECTURE

04 ARCHITECTURE ANALYSIS

05 DESIGN PRINCIPLES

01

PRODUCT FEATURES

Stakeholders

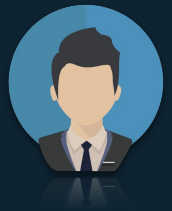
Users

Project Managers

Developers

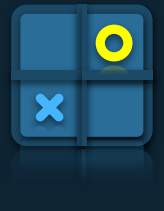
Testers

List Of Features



Login

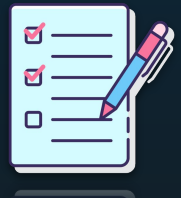
Adventure Mode



PvP Mode

Leaderboard

Summary Report

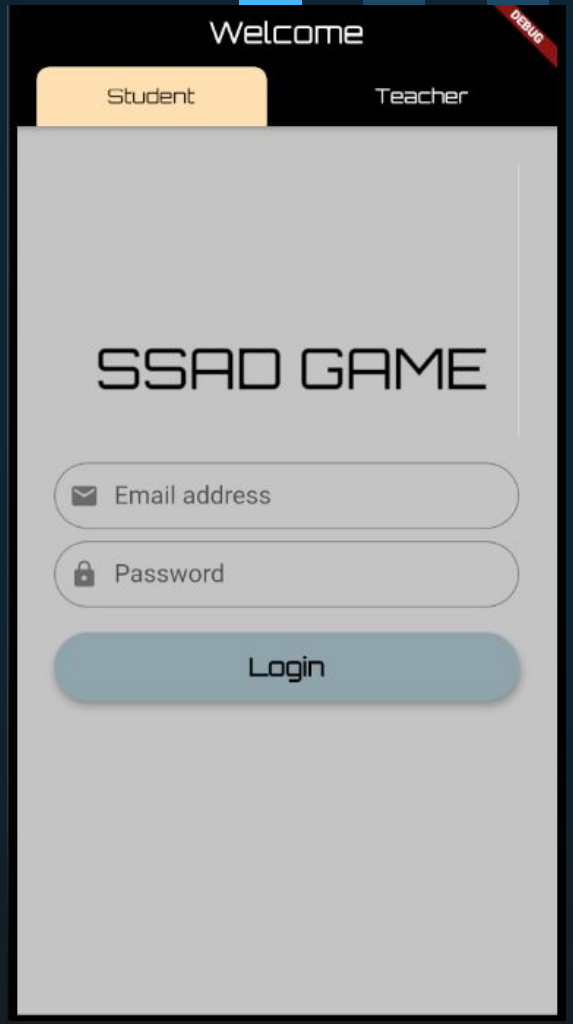


Teacher's Assignment

Login

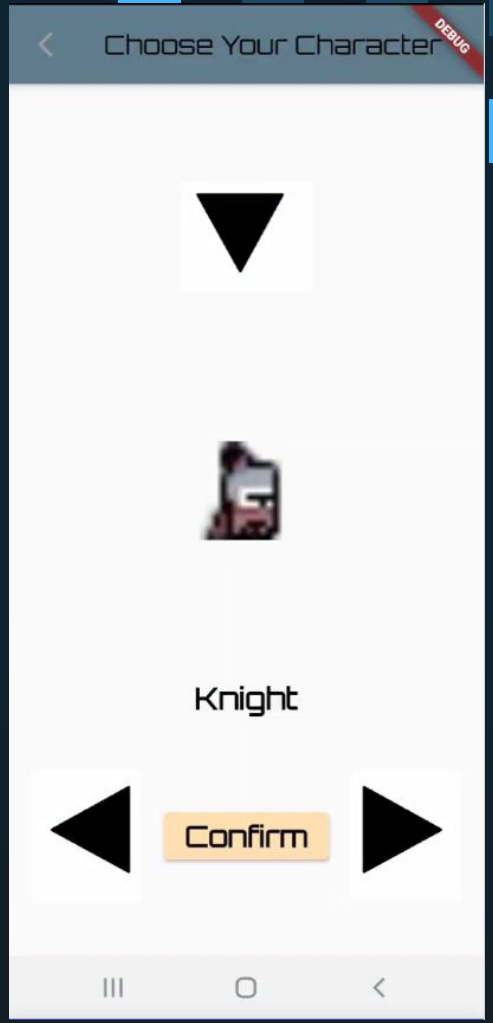
Users can log in as either Student or Teacher.

Credentials are verified through Firebase Authentication API.



Adventure Mode

Students can play an RPG-style adventure game and answer questions to earn points.



PvP Mode

Students can challenge Levels created by others.

They can also create their own Levels with custom difficulty and number of questions.

Leader boards

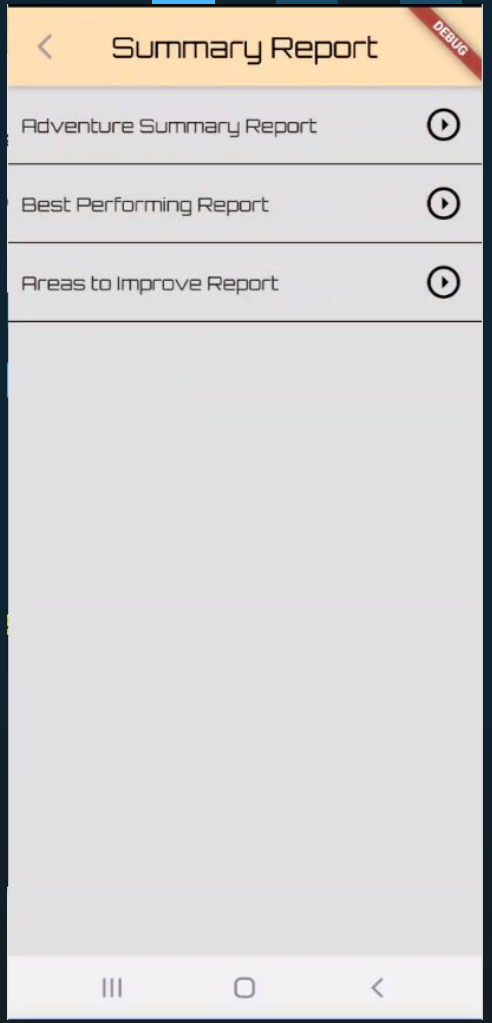
The global leaderboard ranks the collated scores of every user.

Users can view his current standing in the global leaderboard.

Summary Report

Teachers can access a summary of the best performing worlds.

They can also see the detailed breakdown of scores for each Level and Stage.

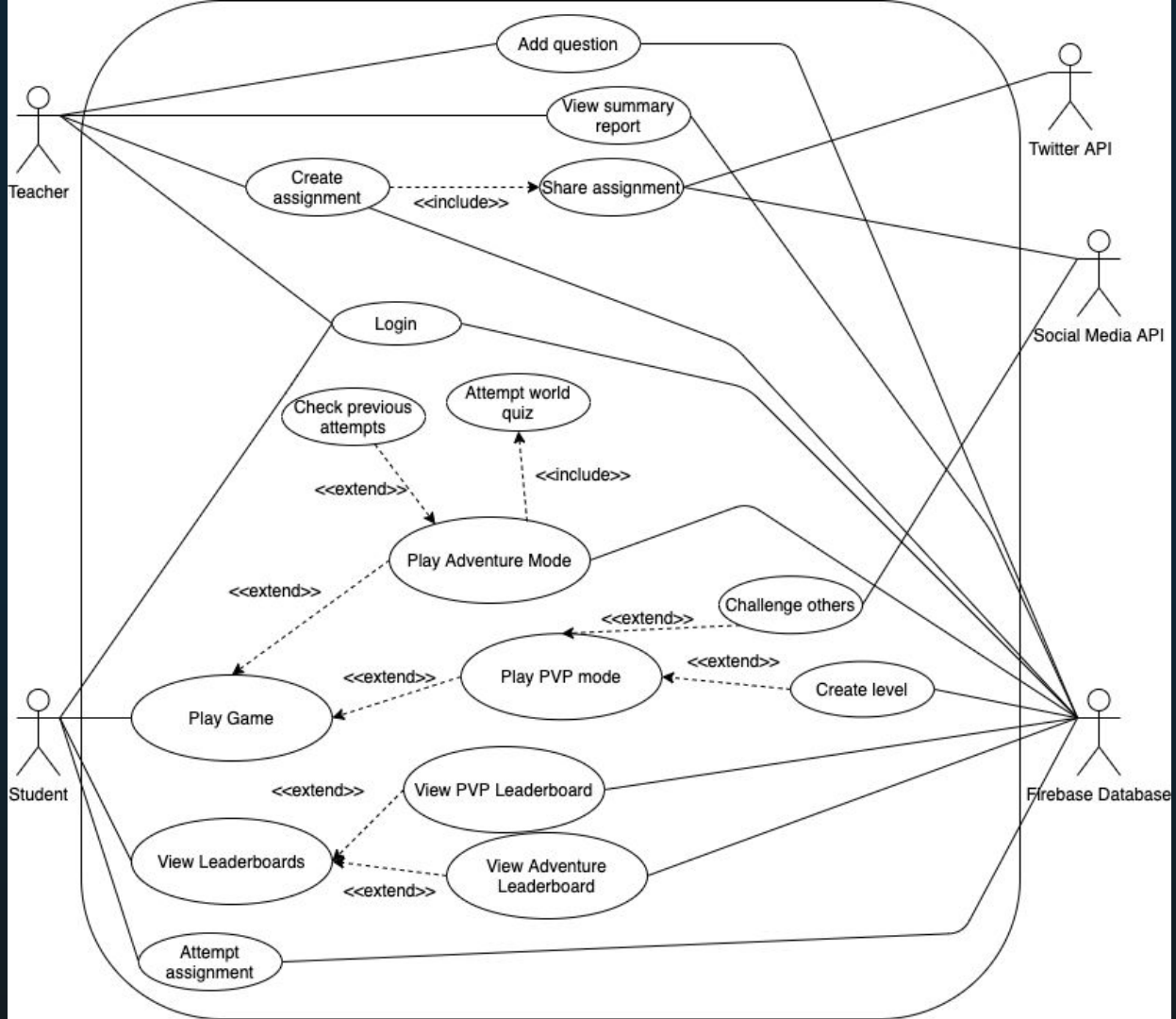


Teacher's Assignment

Teachers can create a level and share it to students as a homework

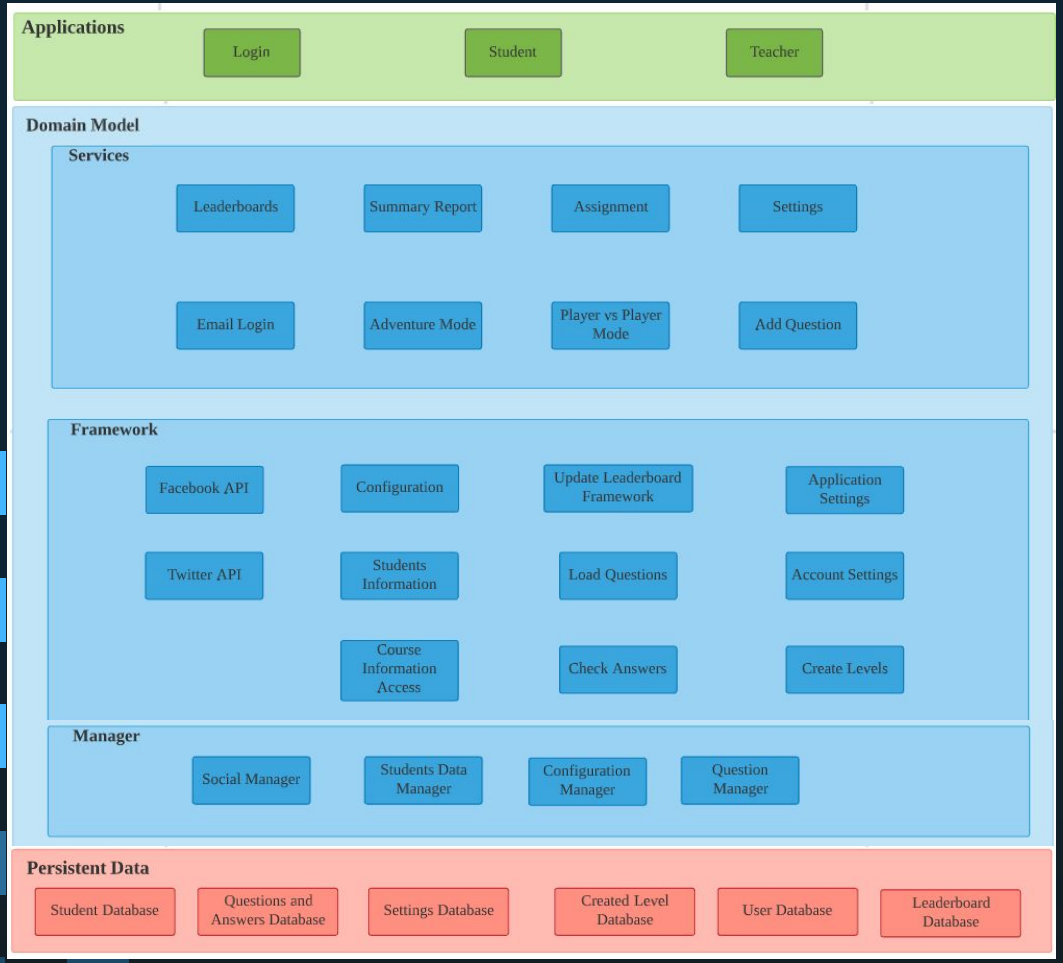
02

USE CASE DIAGRAM


Add question
View summary report
Create assignment
<<include>>
Share assignment
Login
Check previous attempts
Attempt world quiz
<<extend>>
<<include>>
Play Adventure Mode
<<extend>>
Challenge others
<<extend>>
<<extend>>
Play PVP mode
<<extend>>
Create level
<<extend>>
Play Game
<<extend>>
View PVP Leaderboard
View Leaderboards
View Adventure Leaderboard
<<extend>>
Attempt assignment
Teacher
Student
Twitter API
Social Media API
Firebase Database

03

SYSTEM ARCHITECTURE



LAYERED SYSTEM

Components: Passive Components
Connectors: Function Calls

04

ARCHITECTURE ANALYSIS

PROS AND CONS OF LAYERED SYSTEMS

PROS

Modifiability - Changes in one layer will not affect the other layers

Separation of Concerns - Each layer consists of a part of the problem, thus allowing decomposition of problem

Reusability - Components in the lower layer can be reused, thus allowing different implementations of the same layer.

CONS

Performance - Overhead cost of going through intermediate layers. More layers will adversely affect the overall speed of the system

Level of Abstraction - Difficult to gauge the level of abstraction

05

DESIGN PRINCIPLES

NON FUNCTIONAL REQUIREMENTS

RELIABILITY

99% uptime over its entire lifespan

→ App shall always be available other than system update downtime

No deviation from any use case scenarios

→ Execution of the use cases will strictly follow what was proposed in the SRS

ELEMENTS OF GOOD DESIGN

Loose Coupling

Having 3 main Layers and each layer can call components from the lower layer only, thus minimizing dependencies between layers

High Cohesion

All the components providing or accessing a set of services are kept together in the layers.

Eg: The Framework and Service Layer

Single Responsibility Principle

Each class is in charge of one and only one functionality

Eg: LoadQuestionsController is only responsible for retrieving the questions, and nothing else

THANK YOU!