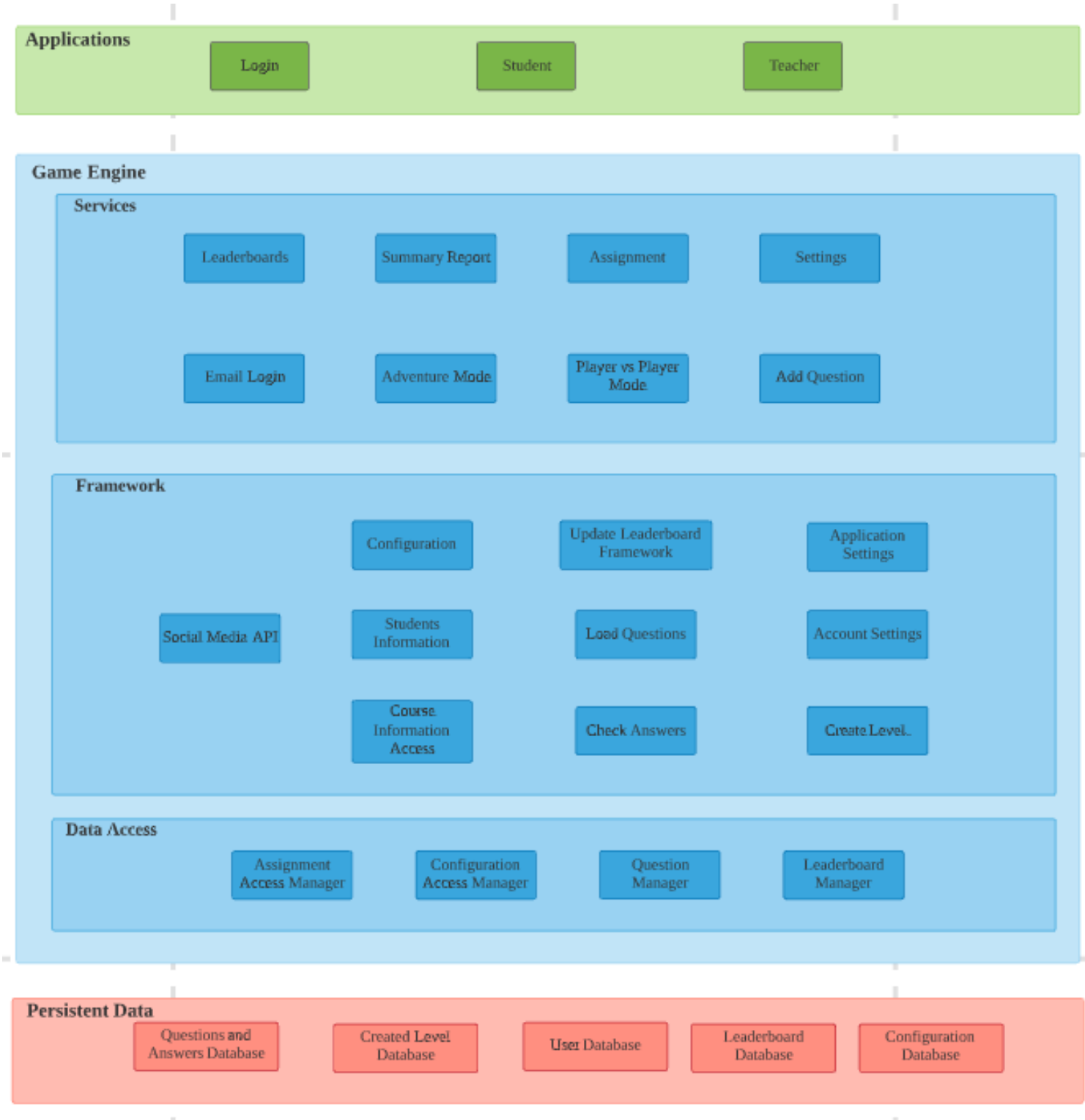
SSADungeon

Candidate Architecture

Call & Return : Layered System

Figure 1.0



8.1 Rationale for Candidate Architecture

8.1.1 Our Approach

Software Architecture is considered as a description of the high level structure of a software system that includes architectural elements such as Components and the interactions between them which are called Connectors. We first begin by analysing the advantages and disadvantages of the various architectural styles, then followed by overall analysis and conclusion of the chosen architecture style.

8.1.2 Compared with other Architecture Styles

Architecture Style	**Advantages**	**Disadvantages**
Independent Components	1. Scalability - Many users can play the game and communicate with the server 2. Concurrency - Processes in the system can run independent of each other at the same time. 3. Accessibility - Server may not be physically close to clients but is still able to be accessed	1. Overloading - If clients request data from server simultaneously, it might get overloaded 2. Fault Intolerant - If server goes down, all clients will be unable to access data
Data Flow (pipe-and-filter)	1. Modularity - All components are independent of one another. 2. Reusability - We can reuse different components in different processes because they	1. Synchronization - Difficult to synchronize the different filters as they need to follow a multi-programming paradigm.

	are independent of one another. 3. Maintenance - Easy to maintain as components can be added and modified without affecting the other components in the process. - This is because the components are not aware of their upstream and downstream components. - A component can be inserted as long as we know the interface. 4. Concurrency - All components are active processes by default. - All components can run independently of each other at the same time. 5. Performance - The advantage where components can run concurrently allows us to achieve parallelism. - This results in increased performance.	- There is a possibility the components will read and write the same data simultaneously. - For synchronization, there is a need to avoid race condition. 2. Bottleneck - May force a lowest common denominator on data transmission - This leads to the performance being dominated by the most time consuming component
Call & Return (Layer System)	1. Abstraction - Supports designs with increasing levels of abstraction. - This allows a problem to be divided into a sequence of incremental steps. 2. Enhancements - Supports enhancements by adding new components.	1. Applicability - Layered System Architecture cannot be applied to all the systems. - This is because not all systems can be easily structured in a layered format. - Even if we are able to structure the

	- Layers of Isolation: Any changes to a layer's function will only affect that particular layer and its associated layers. 3. Reusability - Lower layer's components can be reused while the upper layer's components remain intact. - Allows different implementations of the same layer to be used interchangeably with the condition that they support the same interfaces to their adjacent layers. 4. Testability - Easy to test systems that are structured in layers due to well-defined components and limited component scope.	logic in layers, the need for a better performance might result in closer coupling of logically high- level functions with their low-level implementations. 2. Level of Abstraction - Difficult to find the correct level of abstraction for the system. 3. Performance - Lower Performance - This is because a request for Layer X would have to go through all the other layers that are above it. - This adversely affects the performance as the request has to go through irrelevant layers.

Table 1.0

8.1.3 Rationale for choosing Layered System Architecture

As mentioned previously, in order to select the most appropriate architecture style for our App, we would need a set of criteria or Quality Concerns to aid us in our decision making process. As such the 3 main Quality Concerns we focused on are:

1. Modifiability - ability to make changes to the system
2. Reusability - ability for one component to be used by other components

3. Performance - speed of the system

Keeping these 3 quality concerns in mind, we had decided to not use the pipe-and-filter architecture style despite it allowing components to be reused and high performance. This is because, the key features of the pipe-and-filter architecture style are as follows:

1. The input and output flow continuously through the system.
2. The components include “Filters” which are active components
3. The connectors include “Pipes” which are mere channels through which data flows through.

However this is not the case with our app, the inputs and outputs do not flow continuously through our system and the components of our systems are activated through function calls hence, our components are passive components that might store internal data state and our connectors include function calls through which control and maybe data is passed on. In addition, the pipe-and-filter style might not be the best style when it comes to interactive applications. Due to the pipelining structure, we will have to restart the process every time a user faces an error.Hence, we have chosen not to implement the pipe-and-filter style.

The architecture style Batch Processing was not chosen for the same reasons as not choosing the pipe-and-filter architecture. Another additional reason for not choosing is that the execution of the components happens in a sequential manner thus compromising performance.

Next, the reason why we did not choose independent components is that due to the components being independent, when a component sends a message to another component, it cannot assume that the component that is receiving the message will respond to it. To overcome this problem, interaction protocols must be designed and implemented for all components to follow. This would result in overhead costs in the system. In addition, components of the app can make requests to the server simultaneously so it might get overloaded. Also there is heavy reliance on the server, as if the server goes down all other components will not be able to communicate with the server. Hence, we did not choose independent components.

Hence, the most promising architecture style happens to be the Layered System Architecture which is a substyle of the Call and Return Architecture. We have chosen Layered System architecture style over the Main Program with Subroutine style due to the fact that the Layered style supports design based on increasing levels of abstraction. This allows us to partition a complex problem into incremental steps. For example, having different worlds, stages and levels. To be specific, we have selected the Closed Layered Architecture Style which means that the subroutines in the higher layer can call only the subroutines in the lower layer.

In addition, the Layered Architecture Style fulfills the criteria of Quality Concerns as follows:

1. Modifiability - The subroutines are grouped into layers, hence any changes to a subroutine would only affect that subroutine.

2. Reusability - The subroutines in the higher layer can call any subroutines in the lower layer as such, a subroutine in the lower layer can be called by 2 different subroutines in the higher layer.
3. Performance - There might be overhead costs as a subroutine would have to go through unnecessary layers to get to a subroutine in the lower layer. However, our ap only has 3 main layers and hence performance is not heavily impacted adversely.

Hence we have decided to select the Layered Systems Architecture.

As seen in Figure 1.0, we have 3 main Layers, Applications, Game Engine and Persistent Data. The Layer Game Engine in turn has 3 layers inside of it. These 3 nested layers are Services, Frameworks and Data Access Layer. The components are mostly Passive such as Computational Components, ie: Passive Components that store Data internally while the connectors are Function or Procedural Calls. We have segregated each layer such that each layer manages its own collection of tasks, procedures and controls. This makes the software useful for maintenance and it is easy to delegate different tasks to them.

We have opted to break the Game Engine Layer into three major layers, Framework, Services and Data Access. The lower layer provides the underpinning infrastructure that drives them and consists of structures for question and level access, checking of answers and student information. The Data Access Layer uses the data from the relevant Databases and does the querying for the necessary functions. The subroutines from the above layers such as Framework and Services calls in the subroutines from this layer. A list of the components in the respective nested layers of the Game Engine Layer can be seen in Table 3.0.

A third, auxiliary layer called Persistent Data is located below the layer of the Framework. It includes the databases that the program needs, such as the databases for student and created levels. This allows for a degree of abstraction, as the user faces the services layer, while the layers of the Framework and Permanent Data are not seen by the users.

Attribute Architecture	***Independent Component***	***Pipe and Filter***	***Layered***
Handling Interactive Application	**+**	**-**	**-**

Change in data representation	**+**	**-**	**+**
Reuse	**+**	**+**	**+**
Handling of overloading of server	**-**	**+**	**+**
Performance	**-**	**+**	**-**

Table 2.0

9. Subsystem Interface

Game Engine

The Game Engine Layer contains all the components that are responsible for implementing the main functional requirements of the app. As observed from our Layered Structure Architecture Diagram, we have 3 main Sublayers in the Game Engine Layer, mainly:

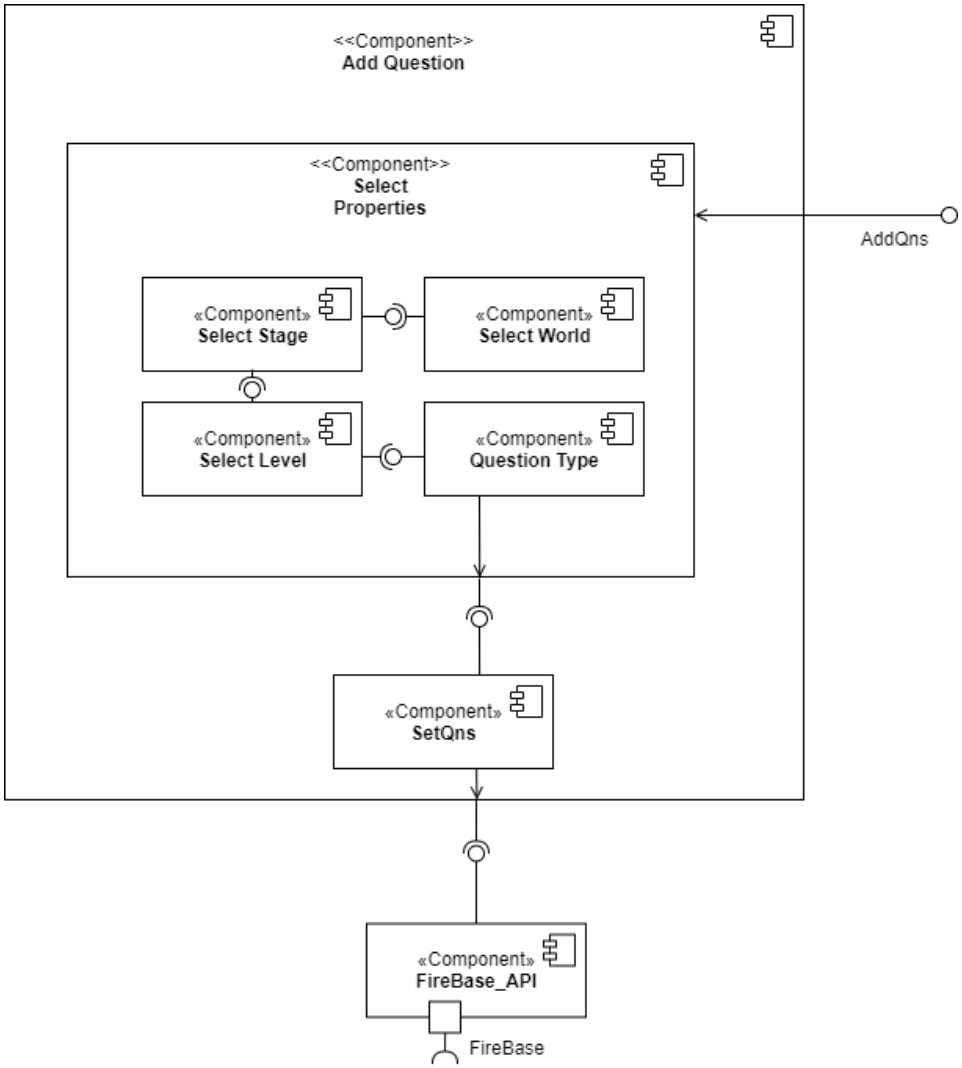
1. Services
2. Framework
3. Data Access

what functions each component uses, and other components, parameters that are passed,
The below table shows the names of the Sublayers and their respective subroutines.

Services	**Frameworks**	**Data Access**
Email Login	Create Levels	Assignment Access Manager
Settings	Social Media API	Leaderboard Manager
Leaderboards	Account Settings	Configuration Access Manager
Adventure Mode	Students Information	Question Manager
Player Vs Player Mode	Configuration	
Assignment	Update Leaderboard Framework	
Add Question	Load Questions	
Summary Report	Check Answers	
	Course Information Access	
	Application Settings	

Table 3.0

Add Questions



This component enables the teachers to create and add questions to the database for the students to answer. It contains subcomponents ‘Select Properties’ which in turn contains 4 subcomponents:

1. Select World
2. Question Type
3. Select Level
4. Select Stage

Component it interacts with: Teacher Component in Applications layer, Configuration and Load Questions, Main Menu

Function Calls and parameters:

	Name	**Parameters**	**Functionality**	**Usage Scenarios**	**Design Rationales**
1	createNewTru	int wNumber,	Create	When user wants	So that

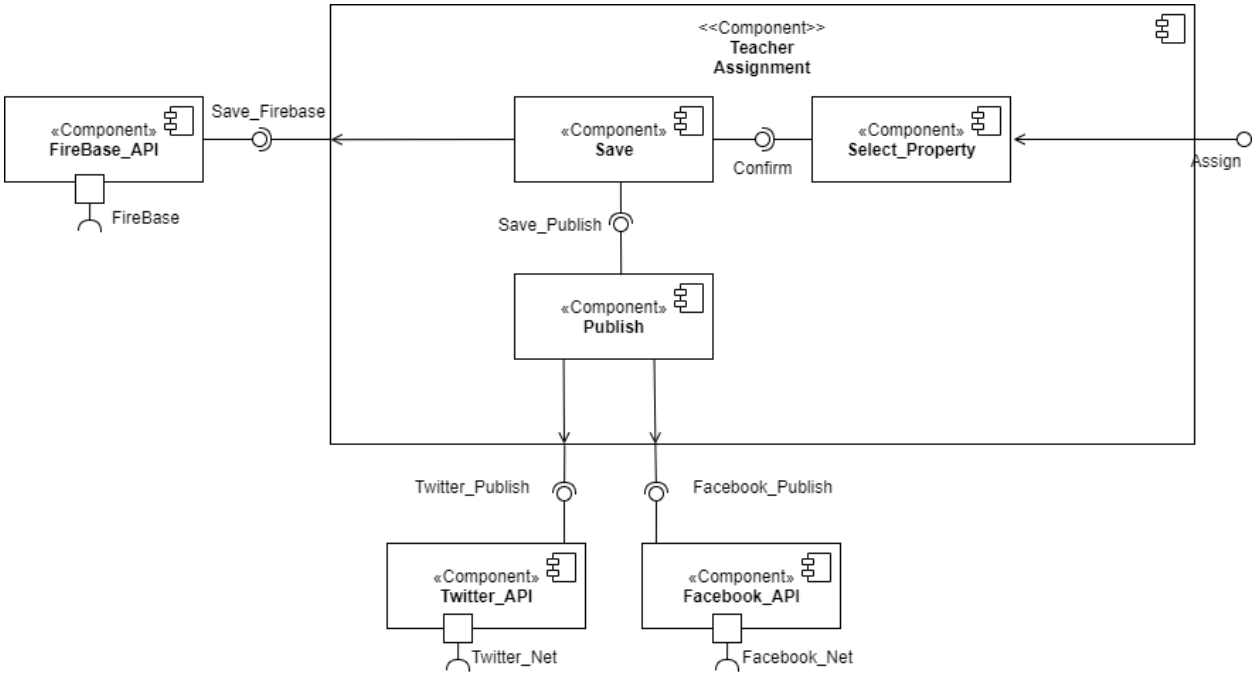
	eFalseQn()	int sNumber, int lNumber, String question, bool answerQ	True/False questions and save it in Firebase	to add new question into question bank	questions will not get outdated and students can have an accurate learning experience
2	createNewFITBQn()	int wNumber, int sNumber, int lNumber, String question, String answerQ	Create Fill-In-The-Blanks questions and save it in Firebase	When user wants to add new question into question bank	So that questions will not get outdated and students can have an accurate learning experience
3	createNewMCQQn()	int wNumber, int sNumber, int lNumber, String question, List<String> answers, String answerQ	Create MCQ questions and save it in Firebase	When user wants to add new question into question bank	So that questions will not get outdated and students can have an accurate learning experience
4	getIdOfAllMcqWorldLevel(), getIdOfAllTFWorldLevel(), getIdOfAllOEWorldLevel	int worldName, int levelName	Retrieve all questions in a level for a specific world	When questions needs to be retrieved	Questions are required for PvP and PvE
5	getIdOfAllMcqQuestions(), getIdOfAllOEQuestions(), getIdOfAllTFQuestions(),	int worldName, int stageName, int levelName	Retrieve all questions	When questions needs to be retrieved	Questions are required for PvP and PvE

Table 4.0

Teacher's Assignments

Teacher's Assignments allows the Teachers to create an assignment by selecting the questions from the database and assigning them to students.It contains 3 components composed inside of it:

1. Save,
2. Select_Property
3. Publish



Component it interacts with: 'Teacher' component from the Applications layer, Configuration, Facebook API, Twitter API, Load Questions

Function Calls and parameters:

	Name	**Parameters**	**Functionality**	**Usage Scenarios**	**Design Rationales**
1	shareAssignment()	NIL	Share assignment created by Teacher via various social media APIs including Twitter and WhatsApp	When Teacher wants to share the created assignment via social media	So that Teachers can let Students know about new assignments
2	savePVPConfigurationForStudent()	int mcqDifficulty, int OEDifficulty, int TFDifficulty, int worldSelected, int noOfMCQ,	Create new PvP level	When teacher wants to create new PvP level as an assignment for Students	So that Teachers can create a PVP level assignments

		int noOfOE, int noOfTF, String createdLevelName			easily
3	giveAccessToTutorialGroup()	String levelName, String tutorialGroupNum	Update tutorial group after world selection	When teacher wants to create new PvP level as an assignment for Students	So that Teachers can create a PVP level assignments easily

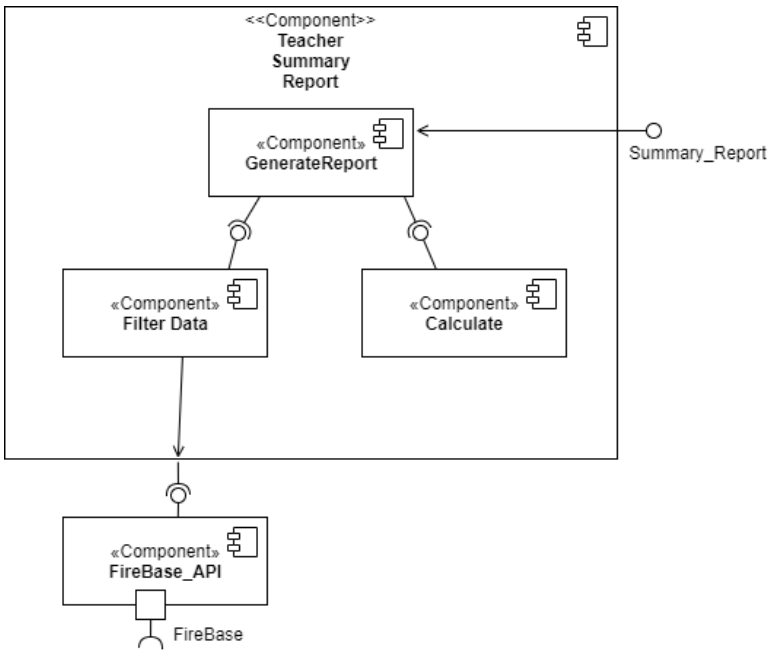
Table 5.0

Teacher's Summary Reports

Teacher's Summary Report includes 3 subcomponents:

1. GenerateReport
2. FilterData
3. Calculate

Which aids in preparing a Summary Report for the Teachers to have an overall idea of their students' performance.



Component it interacts with:'Teacher' in the Applications Layer, Main Menu, Course Information Access

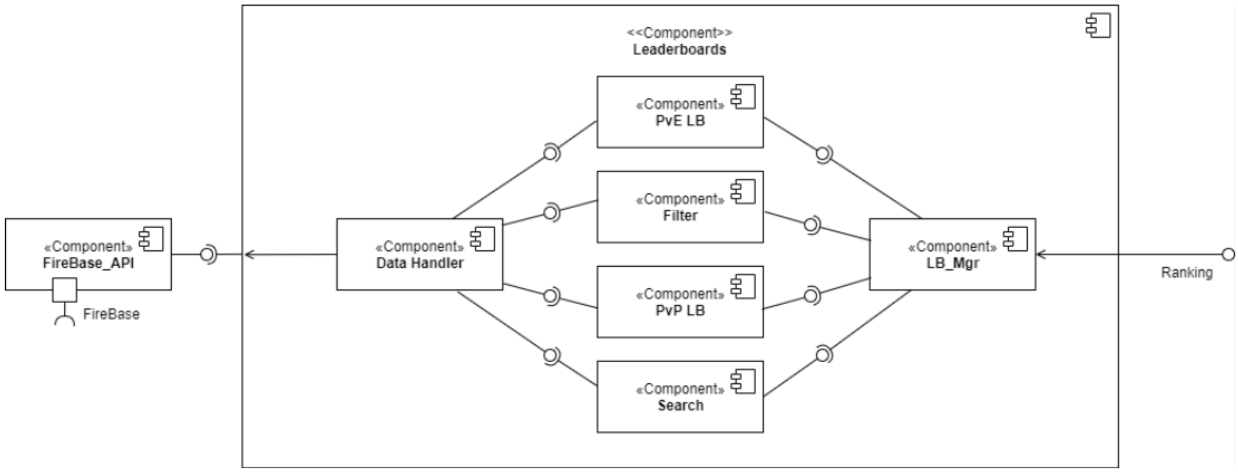
Description of Function Calls:

	Name	Parameters	Functionality	Usage Scenarios	Design Rationales
1	SummaryReportController()	-	Calls the relevant summary report based on the world, stage and level.	When the teacher wants to analyse the students' performance according to the world, stage or/and level.	This function was designed to allow the easy display of the different types of summary reports.
2	setSelectedWorld()	String world	To set a value for the variable stage.	When the teacher chooses to view the students' performance according to the world.	To allow flexibility in designing the View Summary Report functionality.
3	getSelectedWorld()	-	Fetches the selected value of world.	When the teacher chooses to view the students' performance according to the world.	To allow flexibility in designing the View Summary Report functionality.
4	getWorldAverage()	-	Gets the average scores for a world from Firebase	When the teacher chooses to view the students' performance according to the world.	To allow flexibility in designing the View Summary Report functionality.
5	setSelectedStage()	String stage	To set a value for the variable stage.	When the teacher chooses to view the students' performance according to the stage.	To allow flexibility in designing the View Summary Report functionality.
6	getSelectedStage()	-	Fetches the selected value of stage.	When the teacher chooses to view the students' performance	To allow flexibility in designing the View

				according to the stage.	Summary Report functionality.
7	getStageAverage()	-	Gets the average scores for a stage from Firebase	When the teacher chooses to view the students' performance according to the stage.	To allow flexibility in designing the View Summary Report functionality.
8	setSelectedLevel()	String level	To set a value for the variable level.	When the teacher chooses to view the students' performance according to the level.	To allow flexibility in designing the View Summary Report functionality.
9	getSelectedLevel()	-	Fetches the selected value of level.	When the teacher chooses to view the students' performance according to the level.	To allow flexibility in designing the View Summary Report functionality.
10	getLevelAverage()	-	Gets the average scores for a level from Firebase	When the teacher chooses to view the students' performance according to the level.	This function was designed to allow Teachers to create assignments.

Table 6.0

Leaderboards



This component shows the top 50 students PvE and PvP scores, respectively, against one another.

Sub components include:

1. LB_Mgr
2. PvE LB
3. Filter
4. PvP LB
5. Search
6. Data Handler.

Component it interacts with: Student in the Application layer, Students Information, Main Menu, FireBase_API

Description of Function Calls:

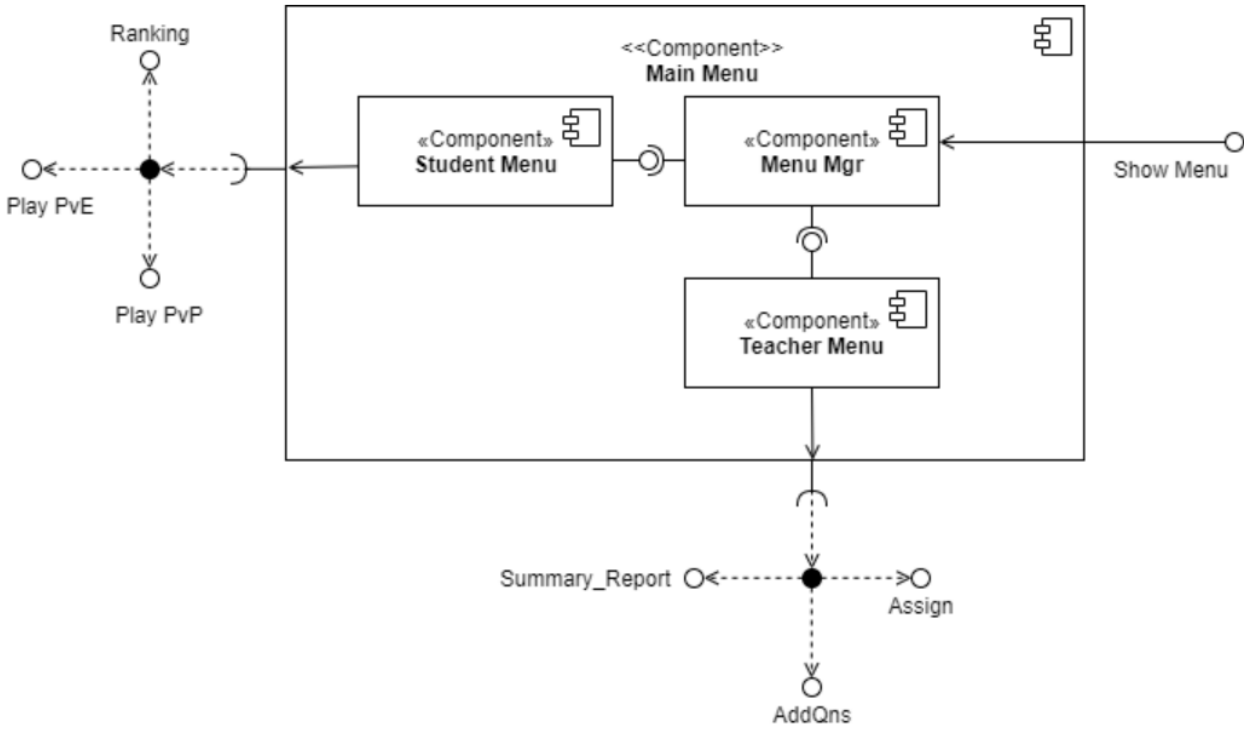
No.	Name	Parameters	Functionality	Usage Scenarios	Design Rationales
1	AdventureLeaderBoardController()	-	Returns users' PvE rankings based on student data	When user wants to know the PvE rankings in the game	To show the level of achievement amongst different players, based on PvE scores earned
2	loadStudentInfo()	String name, int score	Returns students' name and score for PvE	To retrieve student data for display in PvE Leaderboard	If there exists student data, it can be displayed in PvE Leaderboard
3	PvPLeaderBoardController()	-	Returns users' PvP rankings based on	When user wants to know the PvP	To show the level of achievement amongst different

			ratings of the levels created	rankings in the game based on ratings of levels	players, based on PvP level ratings earned
4	loadLevelInfo()	String title, String nameOfCreater, double ratings	Returns title of level, name of creator and ratings for PvP	To retrieve student and level data for display in PvP Leaderboard	If there exists student data, it can be displayed in PvP Leaderboard
5	LevelLeaderBoardController()	-	Returns users' PvP level rankings based on time taken and score earned	When user wants to know the PvP level rankings in the game based on time taken and/or score earned	To show the level of achievement amongst different players, based on PvP individual level's time and score
6	setSelectedLevel()	String level	To set a value for the variable level	When level is required to render corresponding data	User needs to retrieve rankings based on levels in PvP
7	getSelectedLevel()	-	Returns selected level	When level is required to render corresponding data	User needs to retrieve rankings based on levels in PvP
8	getName()	String leaderBoardType, int index	Returns name of users based on time or score	When user wants to know the identity of players in PvP level leaderboard based on time or score	User wants to know the top players in that particular PvP level
9	getScores()	String listName, int index	Returns scores of users based on scores	When user wants to know the identity of players in PvP level leaderboard based on time or score	User wants to know the top players in that particular PvP level
10	getTimes()	String listName, int index	Return time used by users	When user wants to know	User wants to know the top

			based on time	the identity of players in PvP level leaderboard based on time or score	players in that particular PvP level
11	sortMap()	-	Map the name, time and score together	When user wants to know the identity of players in PvP level leaderboard based on time or score	User wants to know the top players in that particular PvP level
12	itemCount()	-	Return top 50 players of that level	When user wants to know the identity of players in PvP level leaderboard based on time or score	User wants to know the top 50 players in that particular PvP level

Table 7.0

Main Menu



This component displays the main page (the page after logging in) for both Students and Teachers.
Sub components include:

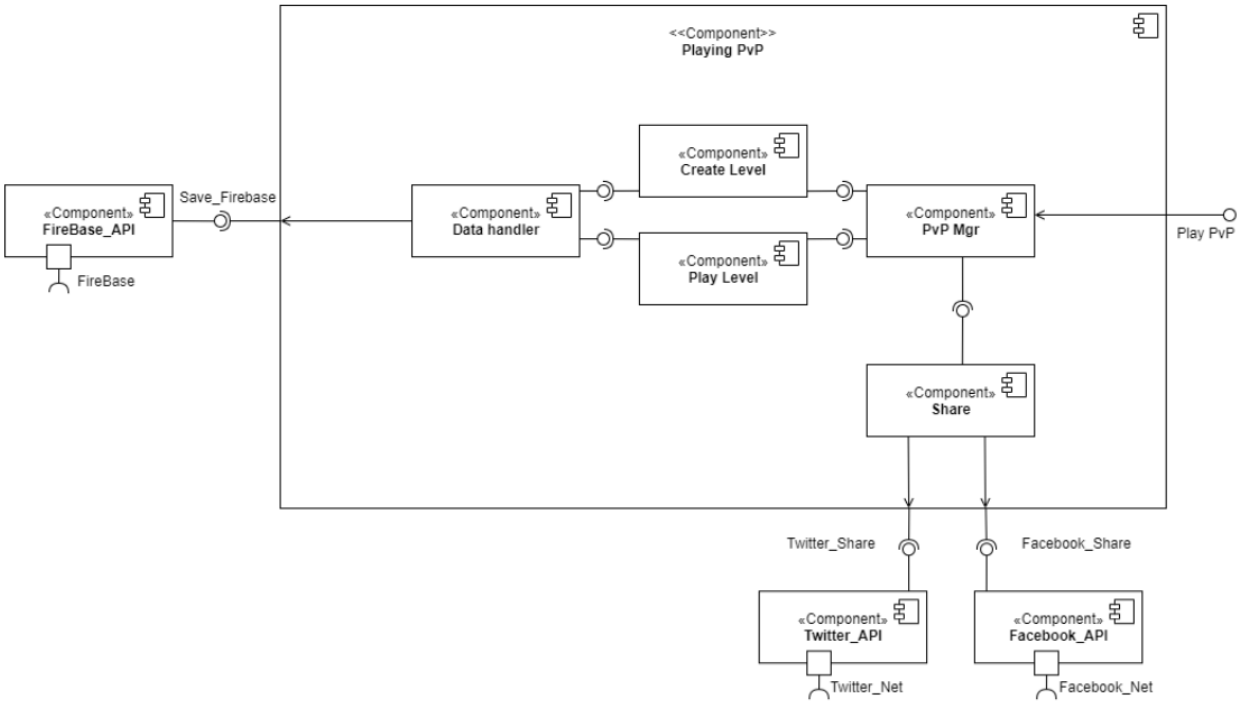
1. Menu Mgr
2. Student Menu
3. Teacher Menu

Component it interacts with: Login, Adventure Mode, PvP Mode, Summary Report, Assignment, Add Question
Description of Function Calls:

No.	Name	Parameters	Functionality	Usage Scenarios	Design Rationales
1.	main()	NA	Entry point of the entire Flutter application	Everytime the Flutter application is run, this main.dart will be the entry point	To ensure all the backend services are initialized and the starting point of the app is properly.

Table 8.0

Playing PvP Mode



This component shows all the available options when a Student selects the PvP Mode, from creation of a level, playing a level to sharing of the results.
Sub component includes:

1. PvP Mgr
2. Share
3. Create Level
4. Play Level
5. Data handler

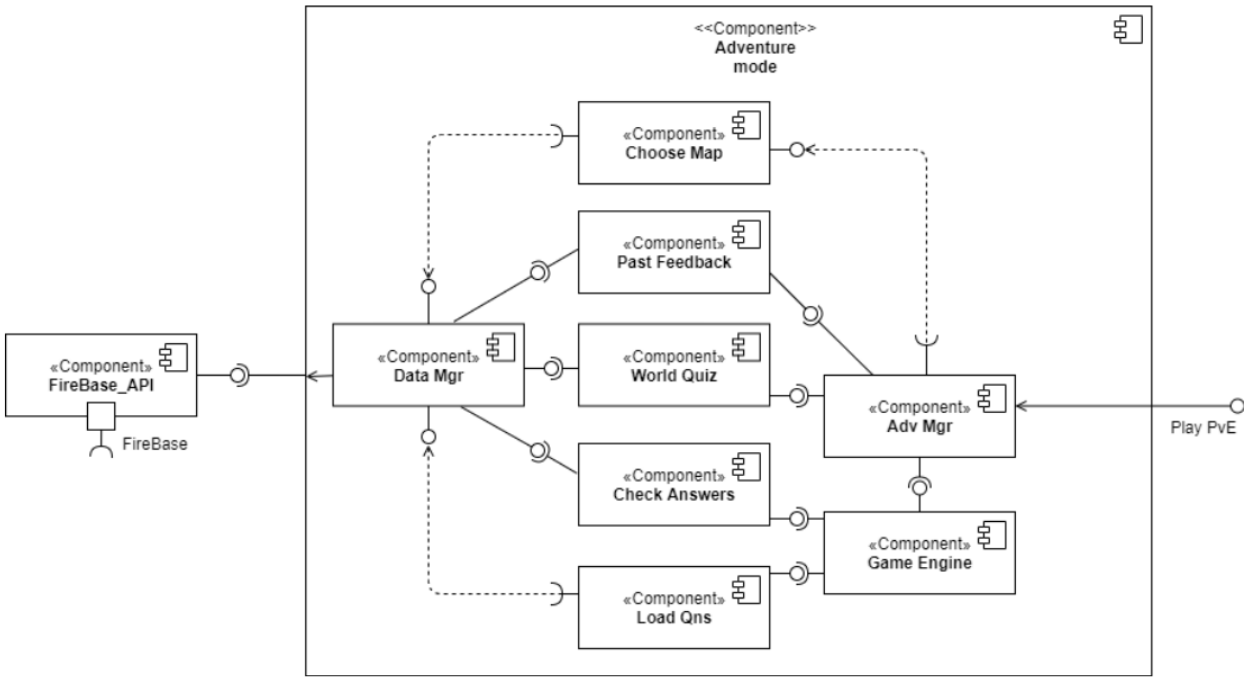
Component it interacts with: Student in the Application layer, Game Engine, Create Level, FireBase_API, Twitter_API, Facebook_API
Description of Function Calls:

No.	Name	Parameters	Functionality	Usage Scenarios	Design Rationales
1	saveScore()	String userID, String levelName, int newScore	User's PvP score after playing will be saved	After user played PvP Mode and the played scores need to be saved	To ensure users' scores are recorded and can be tracked
2	savePVPConfigurationForStudent()	int mcqDifficulty, int OEDifficulty,	Create new PvP level	When user wants to create new PvP level	A free-for-all level creation that promotes friendly

		int TFDifficulty, int worldSelected, int noOfMCQ, int noOfOE, int noOfTF, String createdLevelName		to challenge other players	competition
3	giveAccessToTutorialGroup()	String levelName, String tutorialGroupNum	Update tutorial group after world selection	When user wants to create new PvP level to challenge other players	A free-for-all level creation that promotes friendly competition
4	checkTeacherAssignment()	String tutorialGroupUser	Student can receive assignment created by teacher	When user needs to do assignments created by teacher	An extra homework that targets specific group of students, designed to focus on a specific portion

Table 9.0

Adventure Mode



This component shows all the available options when a Student selects the PvE Mode, from selection of a level, playing of a level to playing the world quiz.
Sub components include:

1. Adv Mgr
2. Game Engine
3. Load Qns
4. Check Answers
5. World quiz
6. Past Feedback
7. Choose Map
8. Data Mgr

Component it interacts with: Student in the Application layer, Worlds, Stages, Level, Game Engine, FireBase_API
Description of Function Calls:

No.	Name	Parameters	Functionality	Usage Scenarios	Design Rationales
1	GameEngine	-	The actual playing of the levels for both PvP and PvE, where the user walks around a map and meets NPCs to answer questions. Every	When user plays a level in PvP or PvE	User can still enjoy the game element even in an educational application, hence more fun and attractive to use the

			level renders a new map.		application
2	getWorldName(), getWorldFName()	Firebase Collection ‘worlds’, currently logged in user	User can view the available worlds in the game and worlds will be unlocked or locked based on user progress	When a user selected PvE Mode and wants to play on a certain world	By showing all worlds in the game with a availability (locking) function, users will be forced to complete the prerequisite to continue progression, controlling users’ progress and learning
3	getWorldsScores(), getWorldsScoresInt()	List<String> wNames	User can view the total scores of a certain world by adding up all the stages scores under that world	When a user selected PvE Mode and wants to play on a certain world	By showing all worlds in the game with a availability (locking) function, users will be forced to complete the prerequisite to continue progression, controlling users’ progress and learning
4	getStageName(), getStageNumber()	Firebase Subcollection ‘Stage’ under ‘worlds’, currently logged in user	User can view the available stages in the game and stages will be unlocked or locked based on user progress	When a user selected PvE Mode and wants to play on a certain stage under a certain world	By showing all worlds in the game with a availability (locking) function, users will be forced to complete the prerequisite to continue progression, controlling users’ progress and learning
5	getStageScores(),	String wName,	User can view the total scores of a	When a user selected PvE	By showing all worlds in the

	getStagesScoresInt()	List<String> sNumbers	certain stage by adding up all the level scores under that stage	Mode and wants to play on a certain stage under a certain world	game with a availability (locking) function, users will be forced to complete the prerequisite to continue progression, controlling users' progress and learning
6	getLevelNumber()	String worldName, String stageNumber	User can view the available levels in the game and levels will be unlocked or locked based on user progress	When a user selected PvE Mode and wants to play on a certain level under a certain stage, which is in turn under a certain world	By showing all worlds in the game with a availability (locking) function, users will be forced to complete the prerequisite to continue progression, controlling users' progress and learning
7	saveStudentAttempt()	HashMap<String, bool> studentResult, String world, int stage, int level	User's results after playing will be saved	After user played PvE Mode and the played results need to be saved	To ensure users' scores are recorded and can be tracked
8	saveAdventureScore()	String userID, int newHighScore, int oldHighScore	User's adventure score after playing will be saved	After user played PvE Mode and the played scores need to be saved	To ensure users' scores are recorded and can be tracked
9	saveAdventureScoreinStages()	String userID, Int newHighScore,	User's adventure score after playing will be saved in stages	After user played PvE Mode and the played scores need to be	To ensure users' scores are recorded and can be tracked

		int oldHighScore(), String world, int stage		saved	
10	checkMCQ(), checkTrueFalse(), checkFillInTheBlanks()	int userAns, int qnid	User's answers for MCQ questions will be checked to identify its correctness	After user played PvP/PvE Mode and the scores have to be checked against its correctness	To ensure users' scores are checked to view if it is correct
11	getTotalAccessForLevels()	String worldName, String stageNumber	Retrieve the total number of level access	For user to know the PvE game progress	Control user's learning progress
12	checkAccessForWorlds()	String worldName	Retrieve the total number of world access	For user to know the PvE game progress	Control user's learning progress
13	getTotalAccessForStages()	String worldName	Retrieve the total number of stage access	For user to know the PvE game progress	Control user's learning progress
14	getCharacterSelection()	-	Return the character selected	When user wants to experience different playstyle	To increase the gaming elements in the application and attract users to have more interest in the game
15	WorldQuizController()	-	Return world quiz questions and the corresponding results	When user wants to progress to the next world	Control user's learning pace

Table10.0