Candidate Architecture

for

Edumon

Version 1.0 approved

Prepared by Group1:

Beh Chee Kwang Nicholas (U1824125F)
Chen Xueyao (U1922640G)
Chong Jing Hong (U1922300B)
Goh Hong Xiang, Bryan (U1920609E)
Huang Shaohang (U1921245D)
Lim Jun Wei (U1922756C)
Muhammad Al-Muhazerin (U1921191L)
Quah Dian Wei (U1920106C)
Shauna Tan Li-Ting (U1840754E)
Swa Ju Xiang (U1822040G)

18/10/2021

Table of Contents

Candidate Architecture 1

Table of Contents 1

Layered Architecture 4

Data Flow Architecture 5

Main Program and Subroutine Architecture 6

Conclusion 7

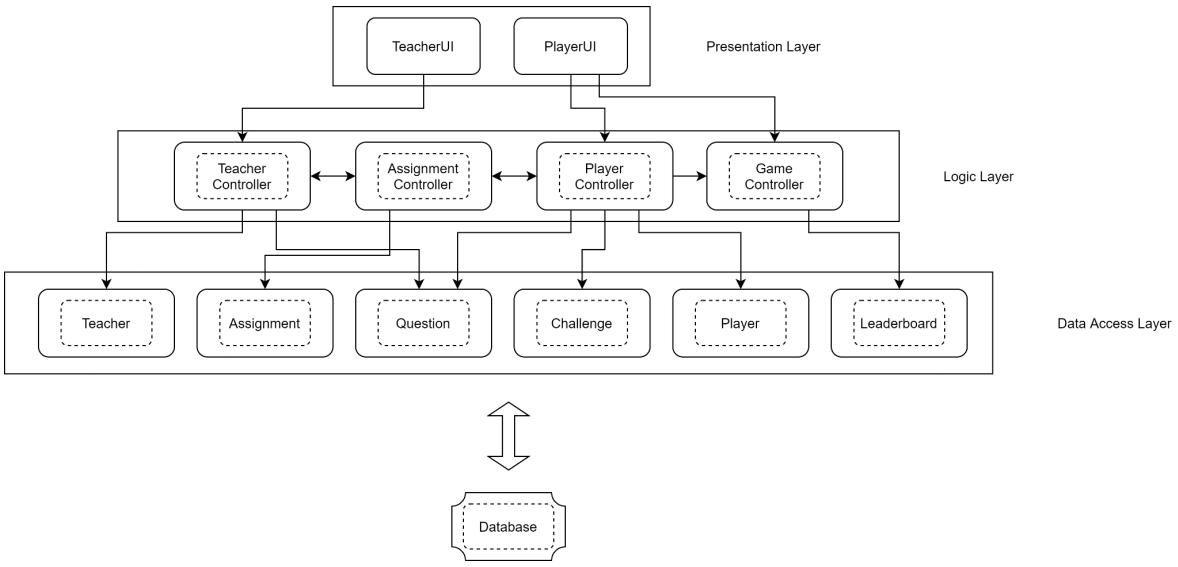
Revision History

Name	Date	Reason for change	Version
Everyone	13/09/2021	Initial draft	0.1
Everyone	17/10/2021	For submission	1.0

Layered Architecture

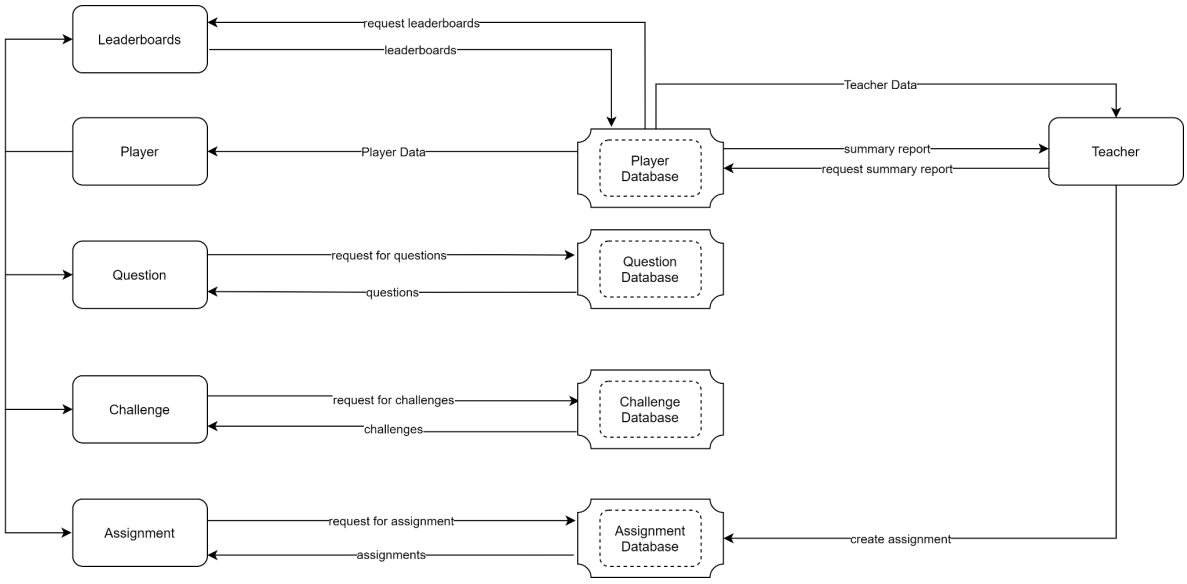
We will be analyzing different types of architectures and select the one most suitable for our application.

The first type of architecture uses abstraction to group the application into 4 main subsystems, presentation layer, logic layer, data access layer and the database. The interactions between the different subsystems are clearly defined and shown as shown in the diagram below. However, it is difficult to determine the data that is passing from one component to another. Connections are also lacking to show the required data.



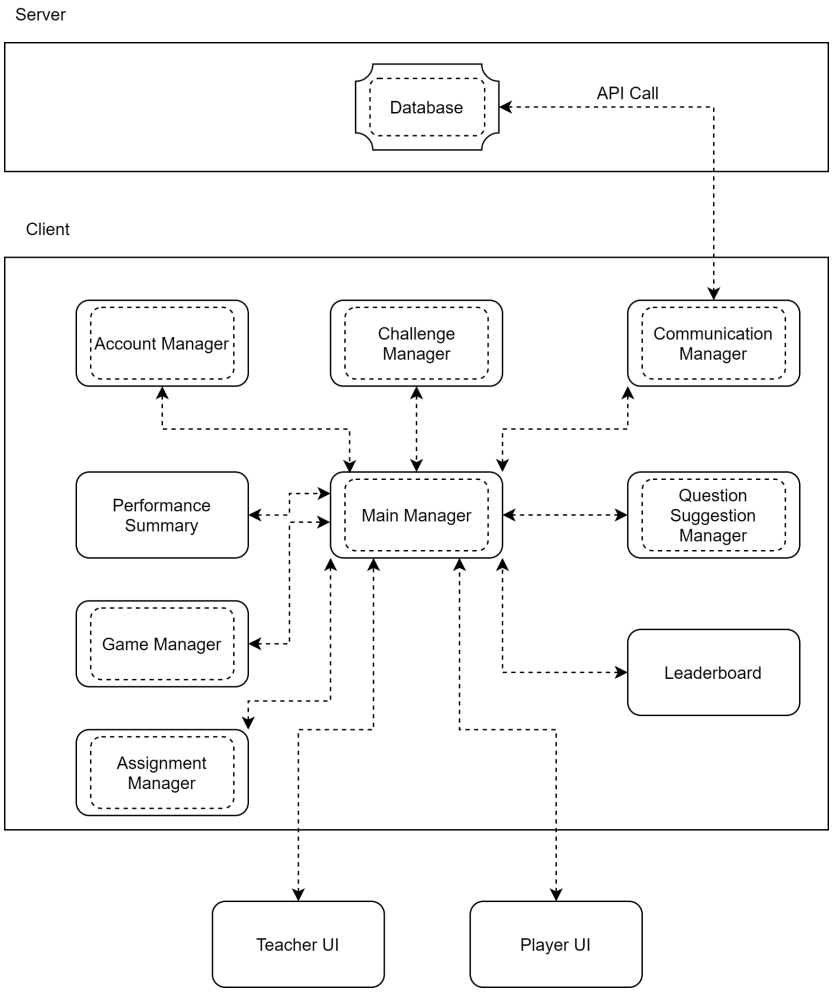
Data Flow Architecture

The second type of architecture has defined data flow within the architecture. From the diagram shown, it is easy to read with icons to differentiate between databases and subsystems. It is also able to show the functions of the different types of users in the game clearly. However, the type of data passing from one component to another is not defined clearly and there is a lack of control flow between the components.



Main Program and Subroutine Architecture

The third type of architecture allows new features to be easily added in the future without affecting a major part of the system. Security of the system can be achieved easier due to access being only granted to authorized clients. Backup and recovery of data can be achieved easier with this architecture. However, there are complicated protocols required in order to manage the connection between the client and the server. If there are too many client requests, the server could be overloaded due to a congestion in the traffic in the connection. It is also difficult to implement the “Main Manager” in the client side due to the number of components it must communicate with.



Conclusion

With the analysis of the different architectures, we have decided to go with the layered architecture as shown in the first diagram. The layers are separated, preventing the user interface from accessing the database directly which allows new features or components to be easily added without affecting existing components in the system.

The team will also be able to easily split the workload to work on multiple components of the system at the same time without depending on the progress of the other components. Future updates to the system can also be easily implemented without affecting other components, being able to reuse the lower layer components to implement new upper layers. Testing of the system is also easier due to the separated layers.