Component Design

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)

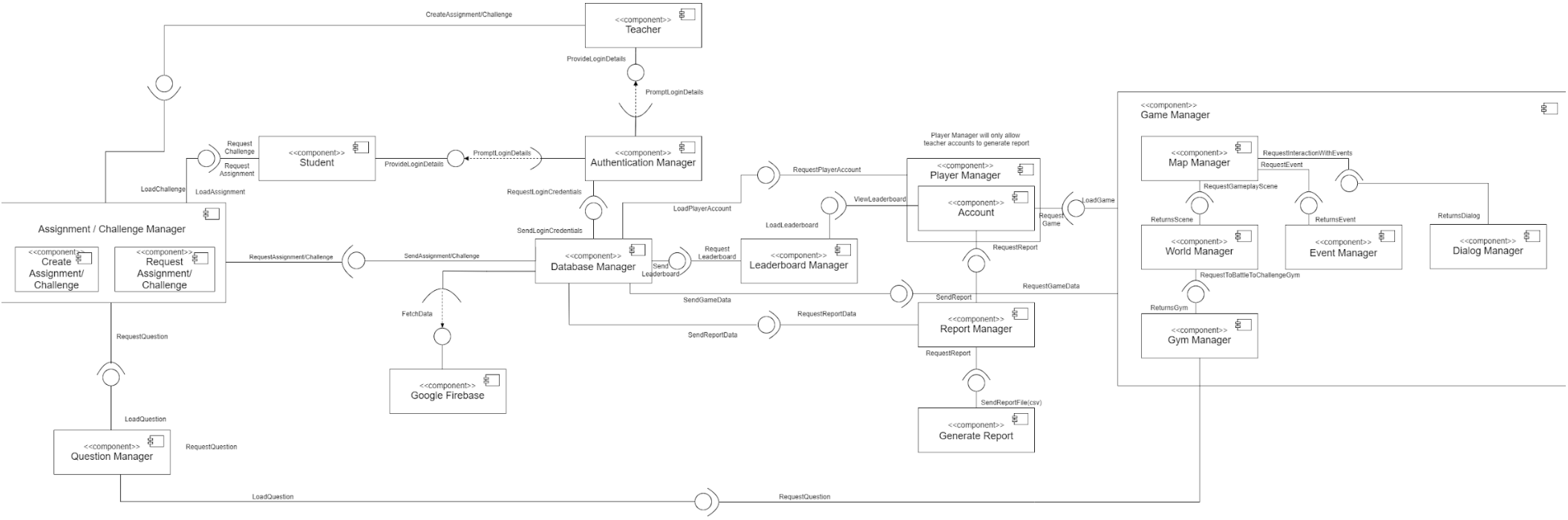
14/11/2021

Table of Content

Revision History

Name	Date	Reason for change	Version
Everyone	04/10/2021	Initial draft	0.1
Everyone	31/10/2021	For submission	1.0

Component Diagram



Description

The component diagram is designed based on a call and return architecture. Every component / manager will request from or send data to the database manager. The database manager will then fetch or store the data on the hosted firebase server.

Students / teachers provide login credentials to the authentication manager, the authentication manager will then fetch the login credentials of the same user from the database manager for validation. The database manager will then return the login credentials to the authentication manager to validate with the inputs from the student / teacher. The authentication manager will then return true if the credentials are correct, else return false if the credentials are not.

Once login is successful, the users have access to the other functions of the application such as game play, report generation and leaderboard viewing etc. This is subject to the type of candidate the users are.

Player manager handles the view for the different users using the application. Teachers are able to create assignments, challenges and generate reports while students are able to play the game and view the leaderboard to check their own rankings.

The player manager will then request game scenes from the game manager if the user is a student, to play the game. Game manager will then request the game data from the database manager. The database manager will then return the game data to the game manager and the game manager will then return the game scenes back to the students. Within the game manager are subsystem components which handle the game logic / flow.

Once the students battle against the gym leaders, their scores will be tabulated and included into the leaderboard when the leaderboard manager requests for the leaderboard and leaderboard data will be sent to the leaderboard manager from the database manager. Students then can request to view the leaderboard via player manager and leaderboard manager will return the leaderboard data back to the player manager for viewing.

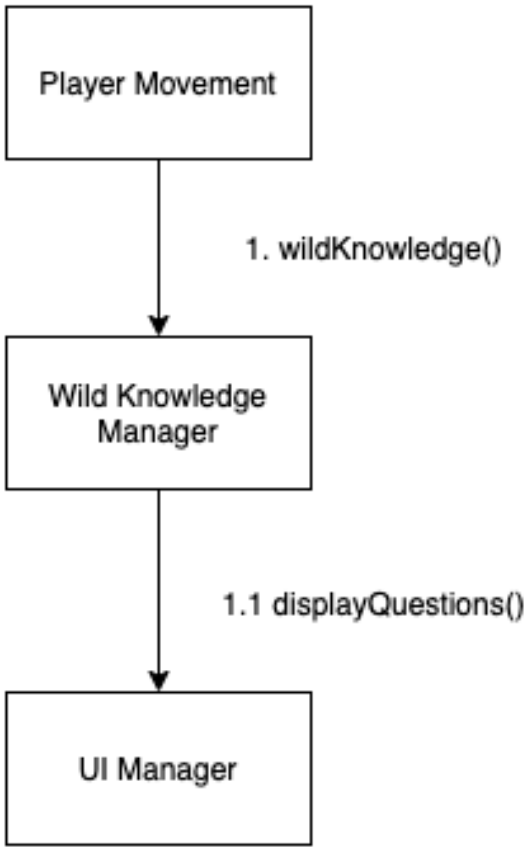
Teachers can generate reports by requesting from the report manager. The report manager will in turn request from the generated report component. The generated report will send the generated report to the report manager and then to the teacher.

Teachers are entitled to create assignments and challenges for the students to work upon. Upon creation, the created assignments and challenges will be passed to the assignment and challenge manager. In turn the challenge manager will pass the created questions to the database manager for storage.

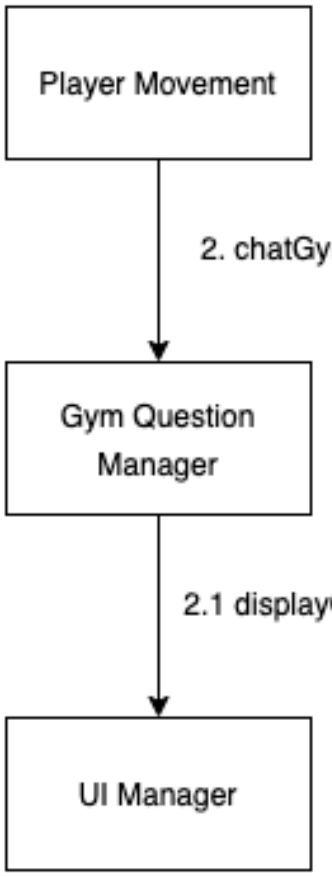
The question manager then handles the fetching of questions from the assignment / challenge manager and returning these questions to the gameplay manager (gym manager).

Communication Diagrams

Player walking in tall grass



Challenge Gym Mast



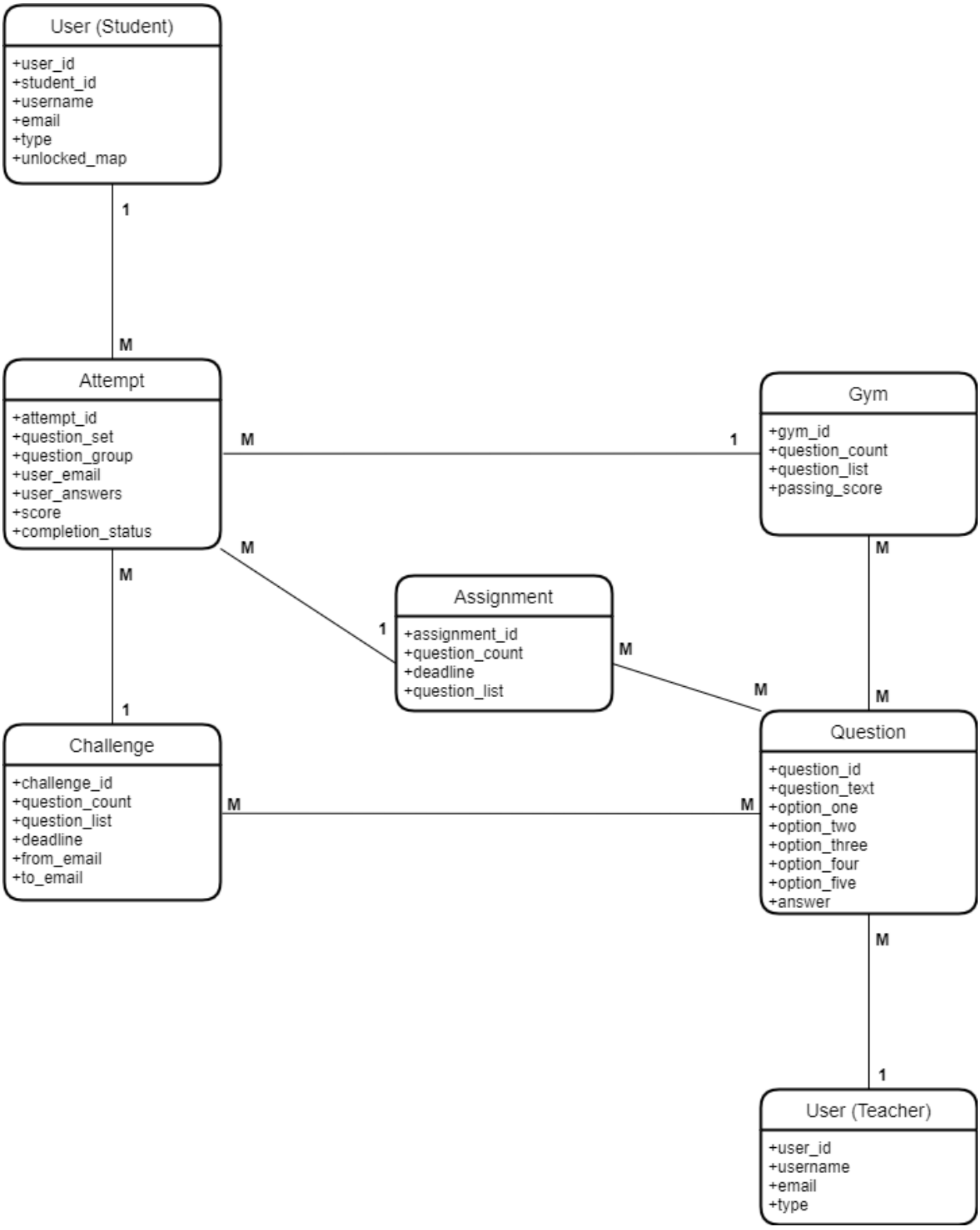
Description

In the above diagrams, we have listed 2 communication diagrams based on the scenarios in Edumon.

The first scenario we have picked out is depicted in the ‘Player walking in tall grass’ diagram. In Edumon, there will be tall grasses stationed on the map, and when a player decides to walk on the tall grasses, there is a chance that the player will encounter “wild knowledge” which will provide the player tips and knowledge for the upcoming challenges. Upon encountering a “wild knowledge”, the Wild Knowledge Manager will display the Wild Knowledge UI through the UI Manager. These “wild knowledge” are designed to assist our players in strengthening their knowledge and serve as practices for their upcoming gym challenges.

The second scenario happens when a player tries to challenge the gyms. In every map, there will be a gym for the player to challenge in order to clear the map and leave his or her name on the leaderboard. Through player movement, the player enters the gym. To trigger the start of the gym questions, the player will chat with the gym master in order to receive the challenge. The player will then complete the challenge within the stipulated time frame.

Data Persistence Diagram



Description

The above diagram describes the relationship between the different tables stored in the database.

The Assignment, Challenge and Gym tables contain an element called question_list, which is a list of question_id from the Question table. This links the Question table to the Assignment, Challenge and Gym tables. Questions can be repeated across different Assignment, Challenge or Gym.

The Attempt table contains an element called question_set, which is an id value from the Assignment (assignement_id), Challenge (challenge_id) or Gym (gym_id) tables. This links the Attempt table to the Assignment, Challenge and Gym tables. The Attempt stores an attempt on one set of questions.

A list of API routes to perform database actions on each table is presented below:

Assignment	**GET /assignment** Get list of all assignments **GET /assignment/{id}** Get an assignment matching the assignment_id **POST /assignment** Create an assignment **PATCH /assignment/{id}** Update an assignment matching the assignment_id **DELETE /assignment/{id}** Delete an assignment matching the assignment_id
Challenge	**GET /challenge** Get the list of all challenges **GET /challenge/{id}** Get a challenge matching the challenge_id **GET /challenge/email/{email}** Get a list of challenges belonging to a user **POST /challenge** Create a challenge

	DELETE /challenge/{id} Delete the challenge matching the challenge_id
Gym	**GET /gym** Get the list of all gyms **GET /gym/{id}** Get a gym matching the gym_id **POST /gym** Create a gym **PATCH /gym/{id}** Update a gym matching the gym_id **DELETE /gym/{id}** Delete a gym matching the gym_id
Question	**GET /question/{id}** Get a question matching the question_id **POST /question** Create a question. **PATCH /question/{id}** Update a question matching the question_id **DELETE /question/{id}** Delete a question matching the question_id
Attempt	**GET /attempt/email/{email}** Get all submissions by user_email **GET /attempt/{id}** Get an attempt matching attempt_id **POST /attempt** Submit an attempt **PATCH /attempt/{id}** Update answers of an attempt by attempt_id

	DELETE /attempt/{id} Delete an attempt matching attempt_id
Account	**GET /account** Get all accounts **GET /account/{id}** Get an account matching uid **GET /account/email/{email}** Get an account matching the email **POST /account** Submit an account **POST /account/username** Update account by id **POST /account/map** Add a map name to unlocked_map for an account based on its id