**Requirements Specifications**

**P01: QuizLink**

**<team member names & ids>**

|  |  |
| --- | --- |
| **Student ID** | **Name** |
| **24100175** | **Abdul Muiz** |
| **24100277** | **Bisma Nawaz** |
| **24100127** | **Abdur Rafae Haroon** |
| **24100308** | **Hafsa Ahmed** |

|  |  |  |
| --- | --- | --- |
| **Content** | **Totals** | **Obtained** |
| Introduction & system actors | 5 | 5 |
| Use case diagram | 10 | 8 |
| Use case descriptions | 20 | 17 |
| Class diagram | 20 | 10 |
| Sequence diagram | 20 | 10 |
| State diagram | 5 | 5 |
| Non-functional requirements | 5 | 5 |
| Who did what | 5 | 5 |
| Review checklist | 5 | 5 |
| Overall formatting/template | 5 | 5 |
| Late submission penalty | -20 |  |
| Github folder structure | -5 | -5 |
| **Total** | **100** | **70** |
| Review | 20 |  |
| **Grand Total** |  |  |

**Table of Contents**

[1.](#_gjdgxs) Introduction 3

[2.](#_30j0zll) System Actors 4

[3.](#_1fob9te) Use Cases 5

[3.1 Use Case Diagrams 5](#_3znysh7)

[3.2 Description of Use Cases 5](#_2et92p0)

[3.2.1 Withdraw cash 5](#_tyjcwt)

[3.2.2 Transfer funds 6](#_3dy6vkm)

[4.](#_1t3h5sf) Class Diagram 7

[4.1 Diagram 7](#_4d34og8)

[4.2 Description 7](#_2s8eyo1)

[5.](#_17dp8vu) Sequence Diagrams 8

[5.1 Use case Name e.g., Withdraw cash 8](#_3rdcrjn)

[5.2 Use case Name e.g., Transfer funds 8](#_26in1rg)

[6.](#_lnxbz9) State Diagrams 9

[6.1 Diagram details 9](#_35nkun2)

[6.2 Diagram 9](#_1ksv4uv)

[7.](#_44sinio) Non-functional Requirements / Quality Attributes 10

[8.](#_2jxsxqh) Who Did What? 11

[9.](#_z337ya) Review checklist 11

# Introduction

The proposed project is a user-hosted trivia web application. This application aims to create an interactive quiz-based learning experience, taking its inspiration from the popular game-based learning platform, ‘Kahoot!’. Designed for students, teachers, trainers or simply anyone who wants to challenge their friends to a quiz, the extent of the web application’s features allow anyone to quickly create and host a quiz-based session and let others join in and participate. Thus, within a session, there will exist a host user who creates the trivia while the participants joining voluntarily act as players. The players will use the quiz ID given by the host to join a particular session.

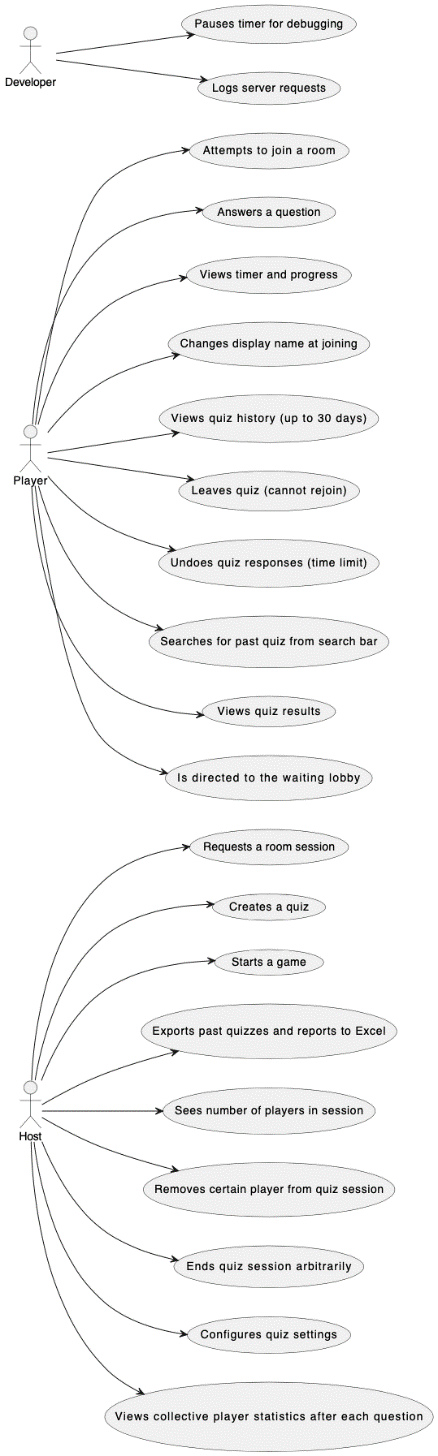
# System Actors

|  |  |
| --- | --- |
| **Actor Name** | **Description** |
| Host | A user with Host status who can create quizzes and host games for other Players to join. Hosts can comprise teachers, students or any other experts looking to test students on a specific subject/topic. |
| Player | A user who can join in any game/quiz made by a Host user. It can include students, teachers or anyone looking to participate in interactive learning activities. |
| Developer | A developer can be either a Player or a Host in developer mode. The developer has access to the advanced tools for the sake of debugging/ development. |
| Database Manager/Admin | Overlooks all the users registered from the app. |

# Use Cases

## Use Case Diagrams

* Why use case diagram was not place here and pasted the link of diagram in document?
* You have four actors in above section and your use case diagram contain 3 actors only.]



[**Use Case diagram**](https://drive.google.com/file/d/1G2fST2df9xVLIulh6pKjZToKR942_zEh/view?usp=share_link)

## Description of Use Cases

[See comments in some selected use cases below.]

3.2.1 Host requests a room session

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-001 |
| **Purpose** | | To provide space for the Host and the Players on the server and give a unique id for Players to join a specific host |
| **Pre-conditions** | | User is recognized as a Host, Host creates a quiz use case is implemented [Which use-case number you are referring to? The post-condition of mentioned use case can become the pre-condition here instead of referring to use-case] |
| **Post-conditions** | | Host is given a unique id for a room and the state is switched to waiting |
|  | | |
| **Step #** | **Typical Course of Action** | |
|  | Host selects the request room id option | |
|  | System asks Host to import a new quiz or choose an existing quiz | |
|  | Host chooses a quiz option | |
|  | System asks Host to give the number of participants that will be in room | |
|  | Host enters the number of participants | |
|  | If the number of participants is valid, the System sends a request to the backend | |
|  | System backend checks availability and sends back a unique room id if space is available | |
|  | Host receives a room id | |
|  | Use case ends | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
|  | Host can cancel room request and go directly to step 9 | |
| **Step #** | **Exception Paths** | |
|  | Step 6 will give an error if the number of participants are not in the bound | |
|  | Step 7 will give an alert and no room id if the space is not available for participants | |

3.2.2 Player attempts to join a room

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-002 |
| **Purpose** | | To allow Players to join to a room created by a Host |
| **Pre-conditions** | | Host requests a room use case is implemented, User is recognized as a Player |
| **Post-conditions** | | Player joins a room on the server of a specific Host |
|  | | |
| **Step #** | **Typical Course of Action** | |
|  | Player selects the option to join a room | |
|  | System asks for a room id from the Player | |
|  | Player provides a room id | |
|  | System takes the request to backend | |
|  | System verifies the room id and sends a response back | |
|  | On valid room id, Player is taken to a room with the host and other Players | |
|  | Use case ends | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
|  | On step 3, Player can choose to cancel and go directly to step 7 | |
| **Step #** | **Exception Paths** | |
|  | On step 5, If the room id is not valid, an error message is sent to the player informing that the room id is invalid | |

3.2.3 Host creates a quiz

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-003 |
| **Purpose** | | Allow host to create a quiz which can be used to host a game |
| **Pre-conditions** | | User is recognized as a Host |
| **Post-conditions** | | Host successfully creates a quiz and is saved on their device, saved in the database or exported by the Host |
|  | | |
| **Step #** | **Typical Course of Action** | |
|  | Host selects the option to create a quiz | |
|  | System asks the Host to create or import a quiz | |
|  | Host selects the create option | |
|  | System asks the Host how many questions they want to have | |
|  | Host enters a valid amount of questions | |
|  | For each question, System asks the Host the question, the four options and the correct option | |
|  | Host fills all the questions | |
|  | System saves the quiz | |
|  | Use case ends | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
|  | In step 2, Host selects the import option and the quiz is imported from their device in the .CSV format | |
|  | In steps 2,5,7, Host has the option to cancel and go directly to step 9 | |
| **Step #** | **Exception Paths** | |
|  | In steps 2, 5, 7, if Host is not complying with the quiz format, an alert message is displayed until the Host complies or cancels the quiz creation | |
|  | In alternate step 2, if the quiz is not in the correct format, an error message is displayed | |

3.2.4 Host starts a game

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-004 |
| **Purpose** | | Allow Host users initiate state to game start state |
| **Pre-conditions** | | Host requests a room is implemented, Player joins a room is implemented, Host has a room and at least two Players in the room |
| **Post-conditions** | | Host enters the game start state |
|  | | |
| **Step #** | **Typical Course of Action** | |
|  | Host selects the option to start the game | |
|  | Systems asks the backend if the players are ready | |
|  | System backend sends a response if the players are ready | |
|  | Host is taken to the game start state | |
|  | Use case ends | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
|  | In step 1, the Host selects the option to cancel the session | |
|  | System sends the request to the backend | |
|  | System backend clears the room and inform all the Users including Host | |
|  | Host is taken back to the default state | |
| **Step #** | **Exception Paths** | |
|  | In step 3, if the players are not ready, the Host is given an alert | |

3.2.5 Player answers a question

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-005 |
| **Purpose** | | Allow Players to submit their answers for a question |
| **Pre-conditions** | | Player joins a room and the Host starts a game session use cases are implemented. The player must be in a room and the game state is in Play |
| **Post-conditions** | | An answer is submitted by the Player |
|  | | |
| **Step #** | **Typical Course of Action** | |
|  | The player is in the game session with the Host and other Players | |
|  | System displays four options to the user | |
|  | Player selects one of the four options for the answer | |
|  | System sends the selected option to the backend | |
|  | System backend send the answer to the Host and informs the Player that the answer is submitted if it is in time | |
|  | Player options are grayed out, making it so the player can not select any more option | |
|  | Use case ends | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
|  | In step 5, if the answer is not submitted on time or is given late, the Player is awarded a score of zero | |
| **Step #** | **Exception Paths** | |
|  |  | |

3.2.6 Players can view timer and their progress.

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-006 |
| **Purpose** | | This use case describes how players can monitor the timer for the current question and track their progress during the quiz question (i.e. question number or total questions remaining or answered so far). |
| **Pre-conditions** | | * The player is logged onto a quiz session. * A quiz session is ongoing. * The player is currently in the process of answering questions. |
| **Post-conditions** | | * The player has the real-time visibility of the timer for the current question. * The player is aware of the progress within the quiz (i.e. number of questions answered so far). |
|  | | |
| **Step #** | **Typical Course of Action** | |
| **1.** | The player joins the quiz session hosted by the host. | |
| **2.** | The quiz session starts and the first question is displayed. | |
| **3.** | The player can see the timer on the screen showing the remaining time for the current question. | |
| **4.** | The player can see the question number and the total number of questions in the quiz, allowing the players to track their progress during the quiz. | |
| **5.** | The player can choose the answer within the time limit for each question. | |
| **6.** | After answering, the response of the player is submitted. | |
| **7.** | The player can see the next question displayed. | |
| **8.** | The above steps are repeated for each question. | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
|  | The next questions will be displayed if the player is unable to submit the answer within the time limit. | |
| **Step #** | **Exception Paths** | |
|  | None. | |

3.2.7 Players can change their display name at the time of joining the quiz.

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-007 |
| **Purpose** | | The use case outlines how players can modify their display names when joining a quiz session. |
| **Pre-conditions** | | * The player is about to join a quiz session. * The player chooses to change the displayed name for the specific session. |
| **Post-conditions** | | * The player’s new display name is visible to other players and the host within the specific quiz session. |
|  | | |
| **Step #** | **Typical Course of Action** | |
| **1.** | The player wants to join a quiz session hosted by a host. | |
| **2.** | The player initiates the join process by entering the ID shared by the host. | |
| **3.** | The system prompts the player to choose the displayed name for the current quiz session. | |
| **4.** | The player confirms the selection of the unique displayed name. | |
| **5.** | The system records the player’s chosen displayed name for the current quiz session. | |
| **6.** | The player successfully joins the quiz with the chosen displayed name. | |
| **7.** | The player participates in the quiz session with the newly changed displayed name (all other players will be able to see the chosen displayed name). | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
|  | If the user chooses the name which is already chosen by another player, the system asks the user to choose another name. | |
| **Step #** | **Exception Paths** | |
|  | None | |

3.2.8 Developers can pause the timer for debugging purposes.

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-008 |
| **Purpose** | | This use case outlines how the developers, while working on specific quiz features or debugging, can pause the timer during the quiz question to examine and test timer-related functionalities. |
| **Pre-conditions** | | * The developer is logged into the development environment. * The developer is working on a feature related to quiz timers. |
| **Post-conditions** | | * The developer successfully pauses and starts the timer during a quiz question for testing and debugging purposes. |
|  | | |
| **Step #** | **Typical Course of Action** | |
| **1.** | The developer accesses the development environment of the application. | |
| **2.** | The developer is working on a feature related to quiz timers. | |
| **3.** | During development or debugging, the developer encounters a situation where they need to pause the timer to examine specific behavior. | |
| **4.** | The developer initiates the process to pause the timer. | |
| **5.** | The system pauses the timer, effectively pausing it at the current countdown value. | |
| **6.** | Once the developer has completed the examination and testing, they initiate the process to start the timer. | |
| **7.** | The system successfully resumes the timer from the point where it was paused. | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
|  | None | |
| **Step #** | **Exception Paths** | |
|  | None | |

3.2.9 Players can view quiz history for up to 30 days

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-009 |
| **Purpose** | | This use case describes how players can access and view the quiz history for quizzes taken within the past 30 days. |
| **Pre-conditions** | | * The player is logged into the account. * The player has participated in one or more quizzes within the past 30 days. |
| **Post-conditions** | | * The player can view and analyze the quiz history for quizzes taken in the past 30 days. |
|  | | |
| **Step #** | **Typical Course of Action** | |
|  | The player logs into the account. | |
|  | The player navigates to the Quiz history section where they can access the past quiz records. | |
|  | The player can see a list of quizzes that he has participated in the last 30 days. | |
|  | The player can select a specific quiz from the list to view the detailed information. | |
|  | The player can see the quiz date, ranking, scores, and the individual quiz performance for the selected quiz. | |
|  | The players can analyze their performance for the chosen quiz. | |
|  | The player can exit the quiz history section at any time and return to the landing page. | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
|  | None | |
| **Step #** | **Exception Paths** | |
|  | None | |

3.2.10 Players can leave the quiz at any time but cannot rejoin the quiz.

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-010 |
| **Purpose** | | The use case outlines how players have the option to exit a quiz session at any point during the quiz, but the player cannot rejoin the same quiz once they have left. |
| **Pre-conditions** | | * The player is currently participating in a quiz. |
| **Post-conditions** | | * The player successfully exits the quiz session and cannot rejoin the same quiz. |
|  | | |
| **Step #** | **Typical Course of Action** | |
| **1.** | The player is currently participating in a quiz session hosted by a host. | |
| **2.** | At any point during the quiz, the player decides to leave the session. | |
| **3.** | The player initiates the exit process by selecting the Leave quiz option. | |
| **4.** | The system confirms the player’s decision to leave the quiz. | |
| **5.** | The system removes the player from the quiz session, thereby ending their participation in the quiz. | |
| **6.** | The player can no longer rejoin the same quiz session once left. | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
|  | None | |
| **Step #** | **Exception Paths** | |
|  | None | |

3.2.11 Players can undo quiz responses during the time limit of the current question.

|  |  |  |
| --- | --- | --- |
| **Identifier** | | U-011 |
| **Purpose** | | The use case outlines how players can undo their responses to a quiz question while the question’s time limit is not over yet (the timer is still active). |
| **Pre-conditions** | | * The player is currently participating in a quiz session. * The player is in the process of answering a quiz question. * The question’s time limit has not expired yet. |
| **Post-conditions** | | * The player successfully undoes the response for the current quiz question. |
|  | | |
| **Step #** | **Typical Course of Action** | |
| **1.** | The player is currently in a quiz session and is presented with a quiz question. | |
| **2.** | The player reads the quiz question and submits an initial response. | |
| **3.** | After submitting the response, the player reconsiders the answer. | |
| **4.** | While the time limit for the current question is still active, the player initiates the undo process to their previous response. | |
| **5.** | The system confirms the player’s decision to undo the response for the current question. | |
| **6.** | The system removes the initial response of the player, allowing the player to choose a different answer. | |
| **7.** | The player selects a new answer within the remaining time limit for the question. | |
| **8.** | The system records the player’s updated response. | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
|  | None | |
| **Step #** | **Exception Paths** | |
|  | None | |

3.2.12 Host exports past quizzes and reports to Excel

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-012 |
| **Purpose** | | The Host can export past quizzes that he has hosted and their reports to excel. The report can show statistics like how many people got a particular question right. |
| **Pre-conditions** | | * The host should have hosted at least one quiz * There is no existing session going on that the host is hosting. |
| **Post-conditions** | | The Host successfully gets the report of all the past quizzes in excel on his device. |
|  | | |
| **Step #** | **Typical Course of Action** | |
| **1.** | The host is logged in to his account. | |
| **2.** | From the main menu he selects the quiz history option | |
| **3.** | He then gets a list of all the quizzes he hosted along with the session id, date etc. He clicks on the export quiz reports to excel option. | |
| **4.** | The host can specify for which quiz he wants the reports or he can select all of them so host specifies which quiz sessions he wants reports for | |
| **5.** | If the host has hosted at least one quiz session and is not currently hosting a session, the app proceeds to export the reports of the quizzes to excel | |
| **6.** | The app opens an excel file that contains the reports of the quizzes the host has chosen. | |
| **7.** | The use case ends. | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
| 1. | In steps 2,3,4 the host can cancel this use case and proceed to step 7. | |
| **Step #** | **Exception Paths** | |
| 1 | In step 5 if one of the conditions is false then the host will get an error message stating which condition was false and execution proceeds to step 7 | |

3.2.13 Admin edits the data of users

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-013 |
| **Purpose** | | This use case outlines how the admin can edit the data of users |
| **Pre-conditions** | | The admin has valid credentials. |
| **Post-conditions** | | The data of users is successfully edited by the admin |
|  | | |
|  | **Typical Course of Action** | |
| **1.** | The admin logs in using the special credentials for admin | |
| **2.** | The Admin dashboard is visible and the Admin selects the database management option | |
| **3.** | The admin selects whether he wants to edit the table of host or players. | |
| **4.** | The admin performs a query to the database on the backend | |
| **5.** | The desired entries are edited in the database | |
| **6.** | The use case ends | |
|  | | |
|  | **Alternate Courses of Action** | |
|  | None | |
|  | **Exception Paths** | |
|  | None | |

### 3.2.14 Host removes a certain player from his quiz session

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-014 |
| **Purpose** | | This use case outlines how a host can remove a certain players from his quiz session. |
| **Pre-conditions** | | The host must be currently hosting a quiz session and atleast one person has joined the session. |
| **Post-conditions** | | The player has been successfully removed and the quiz proceeds normally with the remaining players. |
|  | | |
| **Step #** | **Typical Course of Action** | |
| **1.** | The host is logged into his account and has started a quiz session with at least two player | |
| **2.** | At the bottom of the screen he selects the participants option | |
| **3.** | The host gets a list of all the participants along with their names | |
| **4.** | The host right clicks on the player he wants to remove | |
| **5.** | Other options appear along with remove player option | |
| **6.** | The host selects the remove player option. | |
| **7.** | The player is removed by the system and the quiz proceeds normally with remaining players. | |
| **8.** | The use case ends | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
| 1. | In steps 2,3,5 the host can reconsider and cancel his act of removing the player | |
| **Step #** | **Exception Paths** | |
|  | None | |

### 3.2.15 Host ends a quiz session arbitrarily

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-015 |
| **Purpose** | | This use case outlines how the host can end a quiz session before the timer has finished |
| **Pre-conditions** | | The host must be currently hosting a quiz session. |
| **Post-conditions** | | The quiz session is successfully ended by the system |
|  | | |
| **Step #** | **Typical Course of Action** | |
| **1.** | The host is logged in and is currently hosting a quiz session [is this precondition or course of action?] | |
| **2.** | On the bottom of the screen he selects the end quiz option | |
| **3.** | The application asks him to confirm his action | |
| **4.** | The host selects the confirm button. | |
| **5.** | The system ends the quiz session and makes this quiz and its report available in the quiz history section. | |
| **6.** | The use case ends | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
|  | In steps 2, 3, the host can reconsider and let the quiz session go on instead of quitting | |
| **Step #** | **Exception Paths** | |
|  | None | |

### 3.2.16 Player searches for a past quiz from the search bar

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-016 |
| **Purpose** | | This use case outlines how the player can search for a past quiz (by id or name) he has attempted in the search bar of the quiz history section |
| **Pre-conditions** | | The player is logged in and has attempted that particular quiz he is searching for |
| **Post-conditions** | | The player successfully sees the particular past quiz along with what questions he got right if the host has given access to the right answers. |
|  | | |
| **Step #** | **Typical Course of Action** | |
| **1.** | The player is logged in and navigates to the Quiz History Section | |
| **2.** | The quiz history section opens up that displays the list of past quizzes | |
| **3.** | The user goes to the search bar and types in the quiz name or id | |
| **4.** | If the quiz exists and he has attempted that quiz then the report of that quiz shows up on the screen | |
| **5.** | The use case ends. | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
| 1. | In steps 2,3 the player can reconsider and cancel this use case by doing something else | |
| **Step #** | **Exception Paths** | |
| 1. | In step 4 if the quiz does not exist or the player has not attempted that quiz then an error message is displayed on the screen which says quiz not available. | |

### 3.2.17 Host configures quiz settings.

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-017 |
| **Purpose** | | A Host configures quiz settings before starting a quiz session. |
| **Pre-conditions** | | The Host has created a room/ quiz session and is now in the phase of setting some parameters before broadcasting it to the participants. |
| **Post-conditions** | | The quiz session is configured according to the Host’s preferences. |
|  | | |
| **Step #** | **Typical Course of Action** | |
| **1.** | The Host clicks on the option of “Quiz Settings” after creating a quiz. | |
| **2.** | The Host is directed to another page where they can set quiz parameters such as the time allocated to each question in the quiz, whether skips are enabled for questions or not, and whether the quiz answers will be displayed after each question or towards the end of the quiz. | |
|  | The Host sets each parameter according to their objectives for the quiz. | |
|  | The Host clicks on “Apply” to save all configured settings. | |
|  | The web app updates the quiz settings according to the Host’s choice. | |
| **6.** | The Host clicks on “Publish Quiz” to make it available for joining with the newly set parameters. | |
| **7.** | The use case ends. | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
|  | In steps, 2 and 3, the Host can just select “Default Settings” to go directly to step 4. | |
| **Step #** | **Exception Paths** | |
|  | None. | |

### 3.2.18 A developer logs server requests.

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-018 |
| **Purpose** | | A developer logs all incoming requests made to the server for debugging and monitoring performance. |
| **Pre-conditions** | | The developer has access to the server’s logging functionality and the server is operational. |
| **Post-conditions** | | All requests made to the server are logged and stored for performance analysis and debugging. |
|  | | |
| **Step #** | **Typical Course of Action** | |
| **1.** | The developer access the server’s logging system through some developer tool. | |
| **2.** | The developer navigates to the request logging option and enables it. | |
| **3.** | The developer begins logging all incoming request to the server, for instance, those for path, timestamp, method, parameters etc. | |
| **4.** | The server stores the logged requests in a file or database, which the developer can review later. | |
| **5.** | The use case ends. | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
|  | None. | |
| **Step #** | **Exception Paths** | |
|  | None. | |

### 3.2.19 Host views collective statistics of players after each question.

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-019 |
| **Purpose** | | A Host is able to monitor the quiz results in real time by looking at the quiz statistics of all participants after each question. |
| **Pre-conditions** | | A room session is ongoing i.e participants are answering questions from the Host’s quiz. |
| **Post-conditions** | | A Host is informed of the number of participants still present, the number of correct answers, average time spent on the question and other useful statistics. |
|  | | |
| **Step #** | **Typical Course of Action** | |
| **1.** | A Host is monitoring a quiz session. | |
| **2.** | After the timer for each question ends and participants submit their answers, the web app calculates statistics from the available data. | |
| **3.** | The web app then displays it on the Host’s dashboard. | |
| **4.** | The Host notes the number of players still present in the session, the number of correct answers, the percentage of players selecting each option, and the average time spent on a particular question, allowing them to understand how participants are performing. | |
| **5.** | The Host clicks on “Next” and presents the next question to the participants. | |
| **6.** | The use case ends. | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
| 1. | After step 4, the Host might engage with the participants and offer feedback in real-time such as in the case that the quiz is taking place within a classroom. | |
| **Step #** | **Exception Paths** | |
|  | None. | |

### 3.2.20 Players view quiz results.

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-020 |
| **Purpose** | | Each player is able to view his/ her results after the quiz has ended i.e. they receive immediate feedback after completing a quiz. |
| **Pre-conditions** | | The quiz session has concluded for all participants after the timer for the quiz ends. [does this also check if the use is logged-in or not?] |
| **Post-conditions** | | Each player can see their relative standing, time taken on each question, and overall score. |
|  | | |
| **Step #** | **Typical Course of Action** | |
| **1.** | The quiz session ends, and the player is directed to a “Results” screen. | |
| **2.** | The screen displays a summary of each player’s quiz results including their relative standing, number of correct answers, and time spent on each question. | |
| **3.** | The player can use these statistics to analyze their individual performance in comparison to other participants. | |
| **4.** | The player can navigate to the “Share” button to get a link to share their quiz results with others. | |
| **5.** | The player can click on “Save/ Download Results” to download a copy of their result in CSV format. | |
| **6.** | The player can then click on “Exit” to be directed back to the landing page/ | |
| **7.** | The use case ends. | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
|  | None. | |
| **Step #** | **Exception Paths** | |
| 1. | A player might leave an ongoing quiz session without completing it in which case, they will not be able to view their results. | |

### 3.2.21 A player is directed to the waiting lobby.

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-021 |
| **Purpose** | | When the specified minimum number of participants for a quiz has not been reached, all the participants who have requested to join the quiz will be directed to a waiting lobby until the cap for the room/ quiz session is filled. |
| **Pre-conditions** | | A quiz session has been initiated by a host.  A player attempts to join that quiz session but the minimum participant requirement has not been met yet. |
| **Post-conditions** | | The player waits in the lobby until the minimum participant requirement is met. |
|  | | |
| **Step #** | **Typical Course of Action** | |
| **1.** | The player wants to join a quiz session hosted by a host. | |
| **2.** | The player initiates the join process by entering the ID shared by the host, and clicking on “Join Session”. | |
| **3.** | The web app checks the participant count and compares it to the specified number of participants required to start a session. | |
| **4.** | If the participant count is less than the required minimum, the web app directs the player to another page. | |
| **5.** | The player is directed to the waiting lobby which displays the message: “Waiting for more players to join the session”. | |
| **6.** | The web app continues to monitor the player count and once the minimum requirement is met, the waiting player is notified and directed to the quiz page as the Host starts the session. | |
| **7.** | The use case ends. | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
| 1. | After step 5, the player decides to exit the waiting lobby instead of waiting for more players to join. | |
| 2. | The player goes on to join some other quiz session. | |
| **Step #** | **Exception Paths** | |
|  | The player might face connectivity issues while in the waiting lobby and have to rejoin the quiz. | |

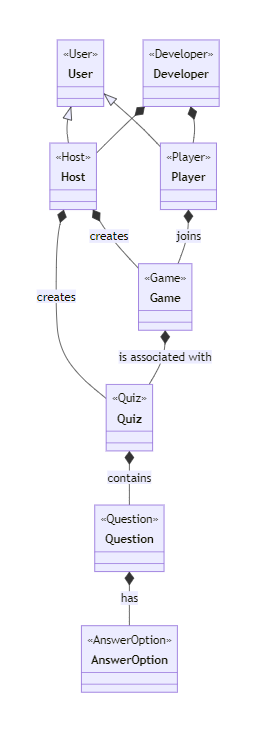
# 

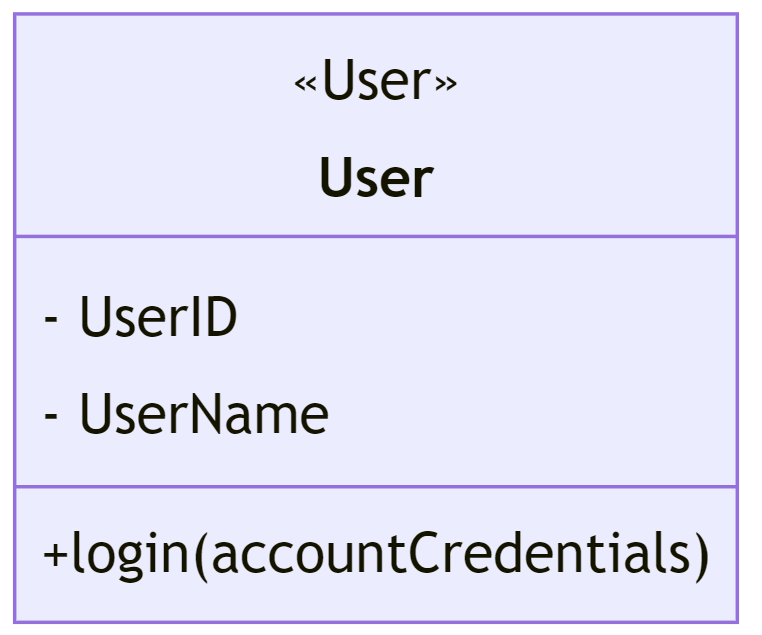
## 3. Class Diagram

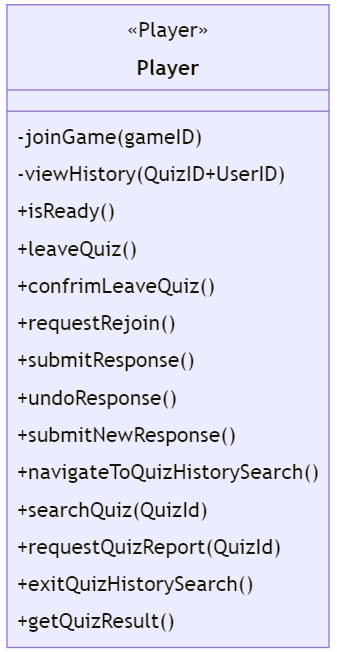
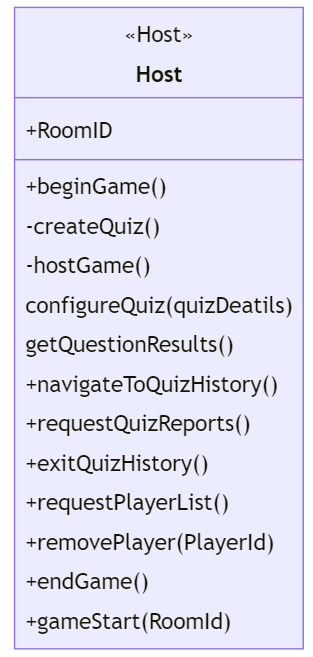
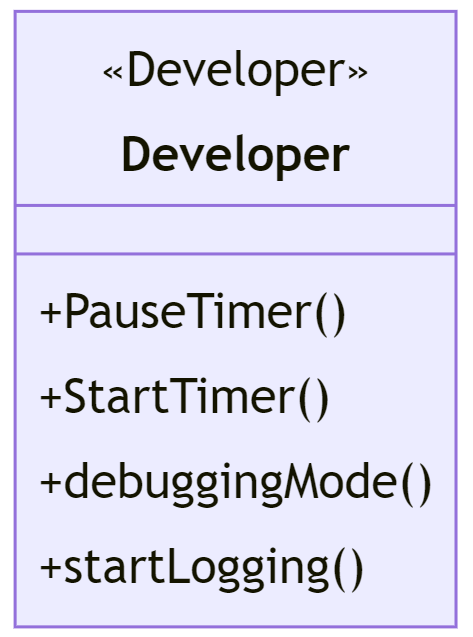
## Diagram

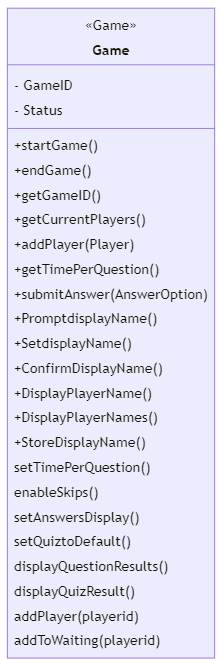
[Class descriptions are part of class diagram. The current diagram is not showing the complete picture of the system.

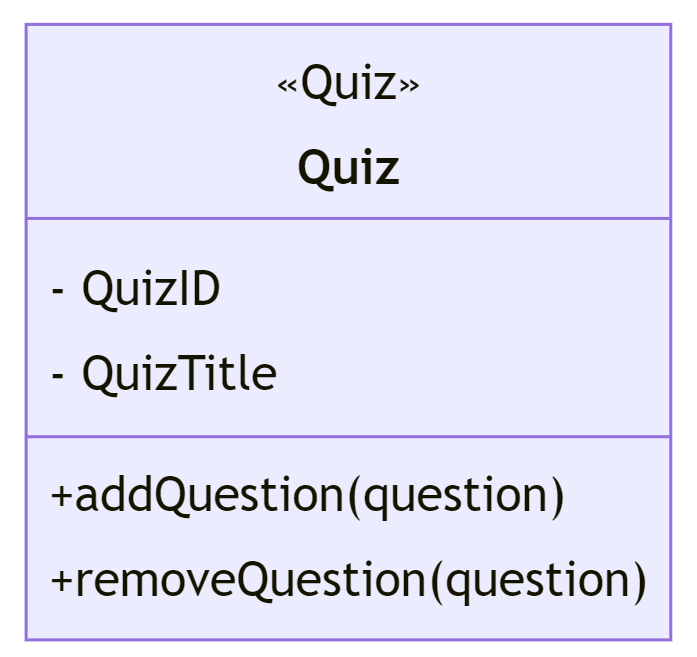
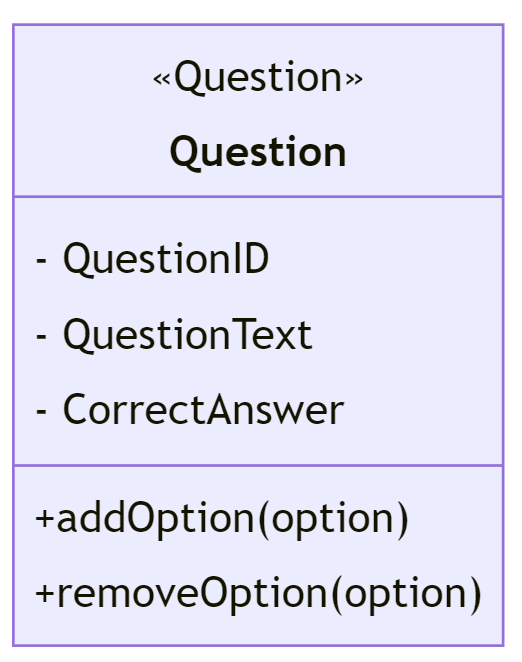
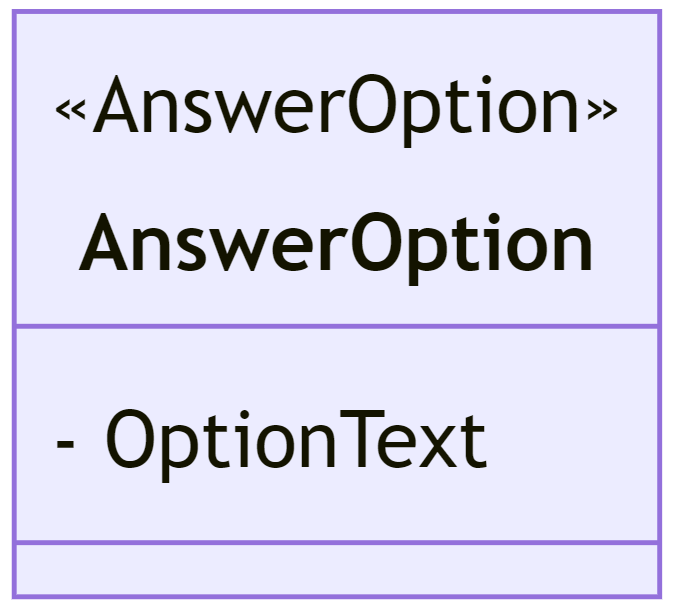
You need classes for payments, user access control.]

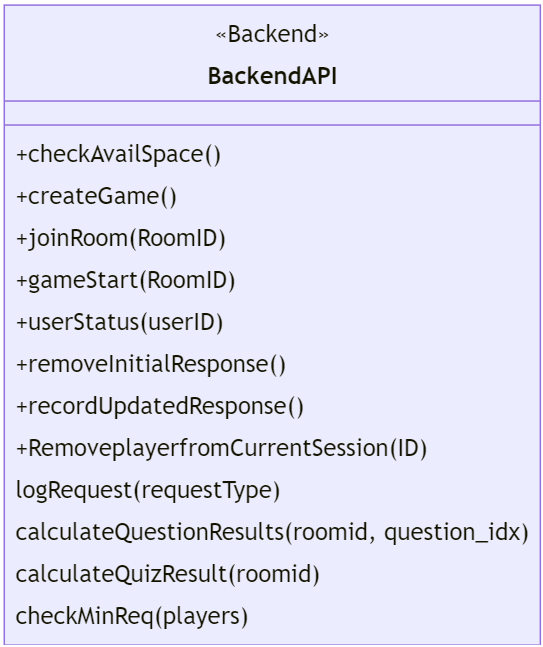












## Description

**User**: Represents a user of the system. It contains attributes such as UserID and UserName. Users can either be Hosts or Players.

**Host**: A subtype of User, responsible for creating quizzes and hosting games. It has methods to create a quiz and host a game.

**Developer**: A subtype of User, likely responsible for system development and debugging. It has a method for enabling debugging mode.

**Player**: Another subtype of User, responsible for joining games and viewing their history. It has methods to join a game by specifying a gameID and view quiz history by specifying a combination of QuizID and UserID.

**Game**: Represents a game instance with attributes like GameID and Status. It has methods to start and end the game.

**Quiz**: Represents a quiz with attributes like QuizID and QuizTitle. It has methods to add and remove questions from the quiz.

**Question**: Represents a question within a quiz with attributes like QuestionID, QuestionText, and CorrectAnswer. It has methods to add and remove answer options for the question.

**AnswerOption**: Represents an answer option for a question. It contains the OptionText attribute.

**Relationships and associations:**

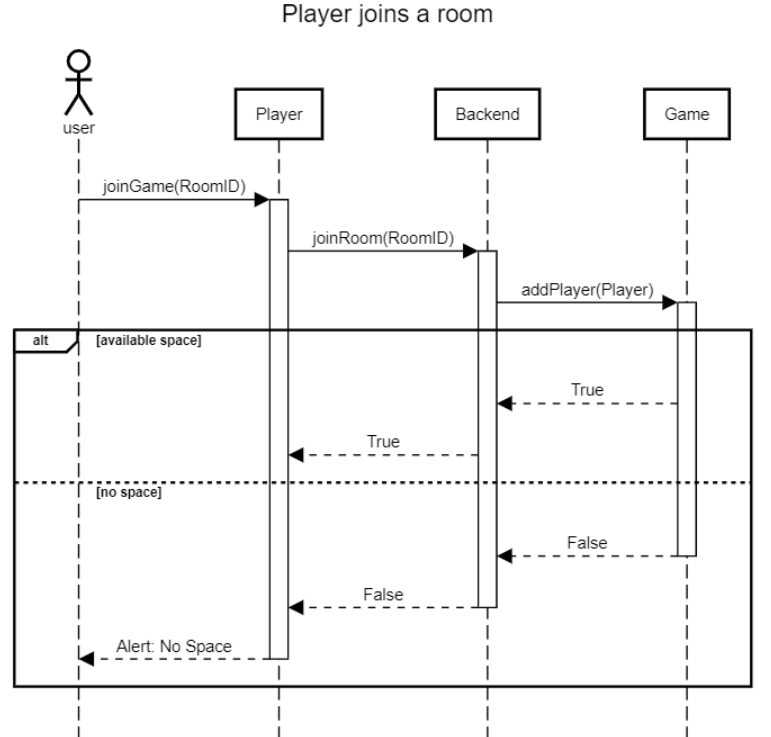
* **User** can be a **Host** or a **Player**, indicating the inheritance relationship.
* **Host** creates **Games**, and **Player** joins Games, showing a many-to-many relationship between Hosts, Players, and Games.
* **Developers** can be associated with both **Host** and **Player**, indicating that they can perform development tasks for both roles.
* **Host** creates **Quizzes**, and **Games** are associated with Quizzes.
* **Quizzes** contain **Questions**, and Questions have **AnswerOptions**.

# Sequence Diagrams

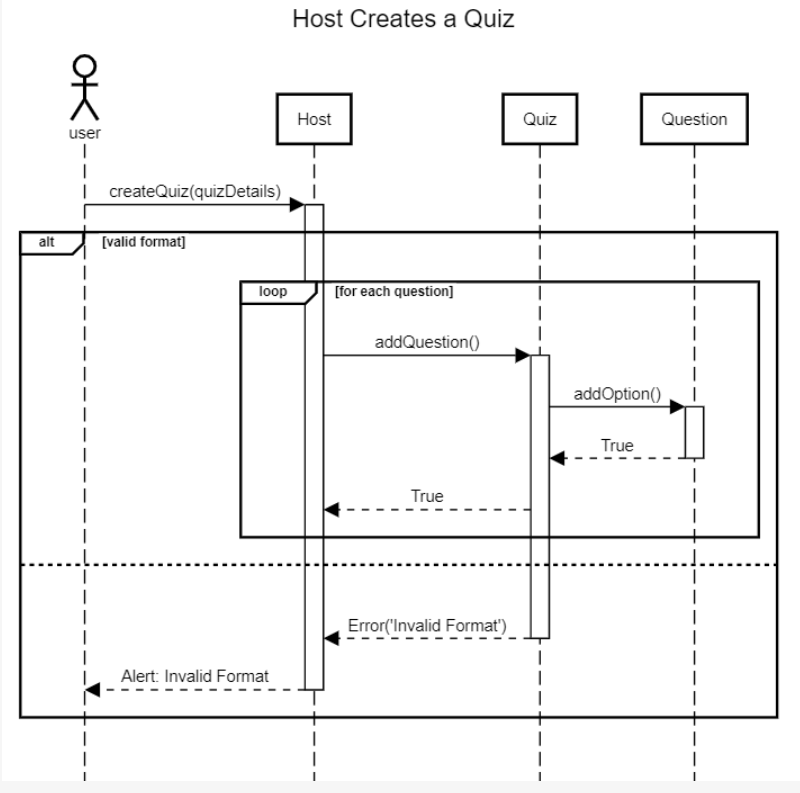
[The sequence diagrams must have classes from the Class diagram you created earlier in this document. These diagrams must show the interactions among the system classes. Some of the sequence diagrams that you have created, there are extra classes from your class model, e.g., Backend. If sequence diagram is incomplete without these classes, you should include these in your class diagram as well. Make your sequence diagrams consistent. These diagrams are not complete and you should complete these along with class-diagram.]

* 1. **Host requests a room session [Missing]**

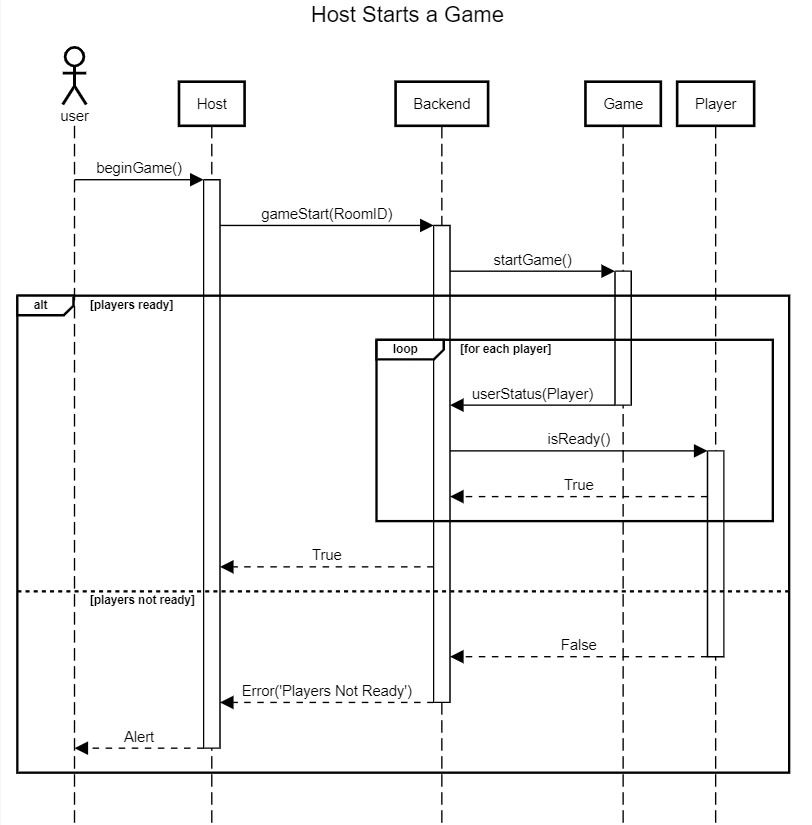
## Player attempts to join a room



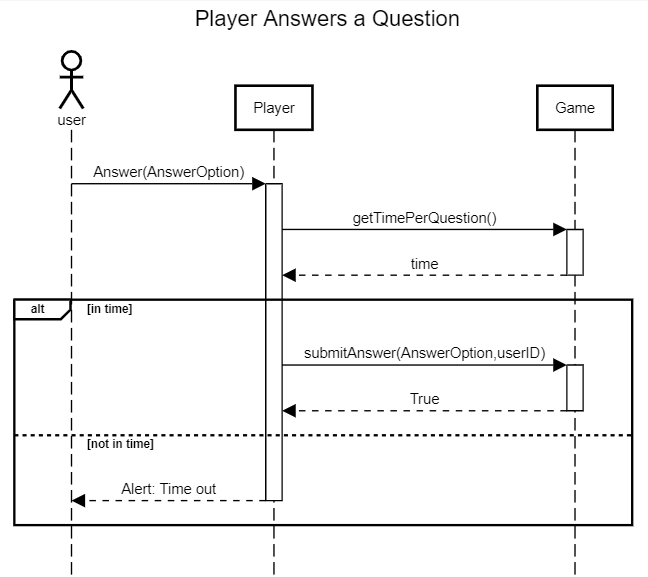
## Host creates a quiz



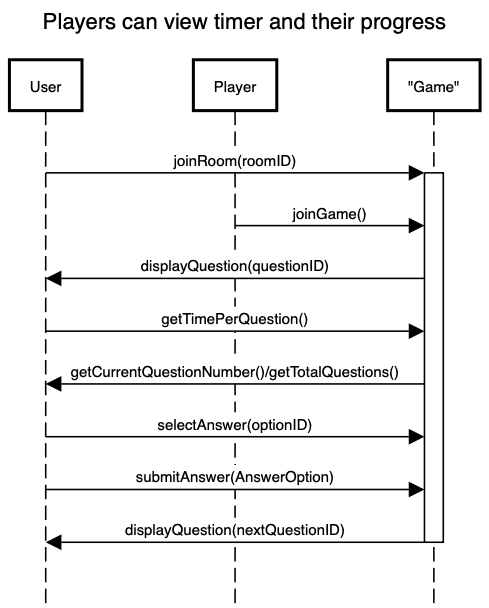
## Host starts a game



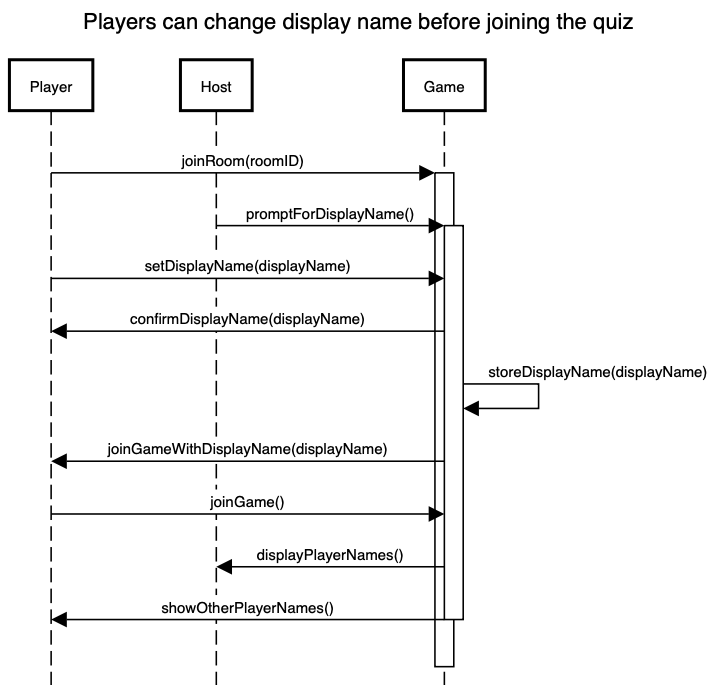
## Player answers a question



* 1. **Players can view timer and their progress.**

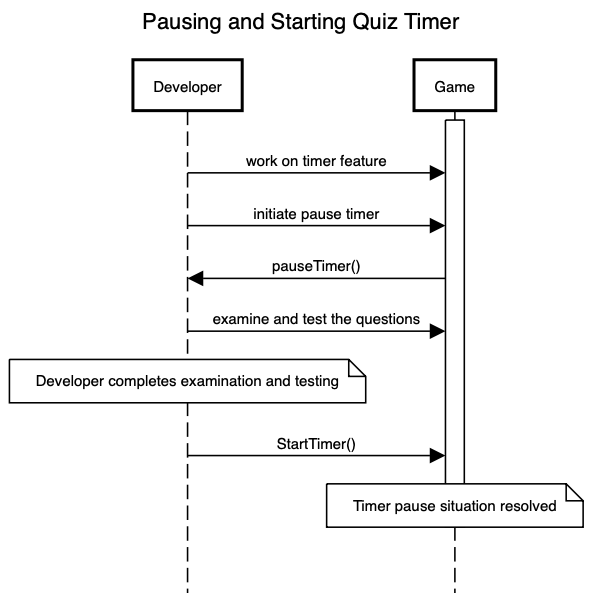
****

* 1. **Players can change their display name at the time of joining the quiz.**

****

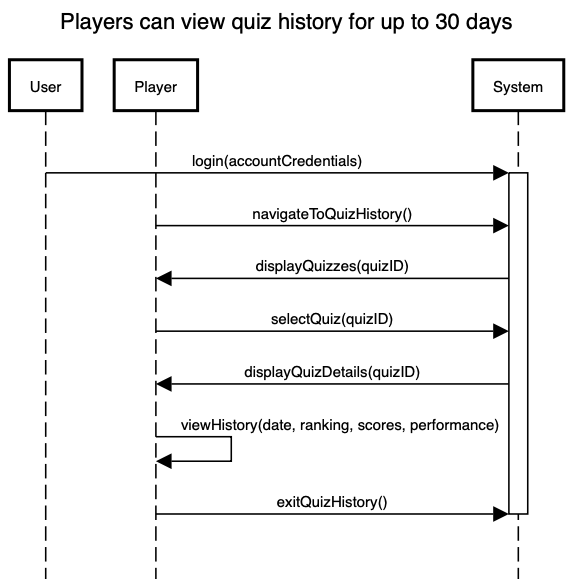
* 1. **Developers can pause for debugging purposes.**

**[Who is the actor here? Some of the sequence diagrams are inconsistent to others, please see request and response here.]**

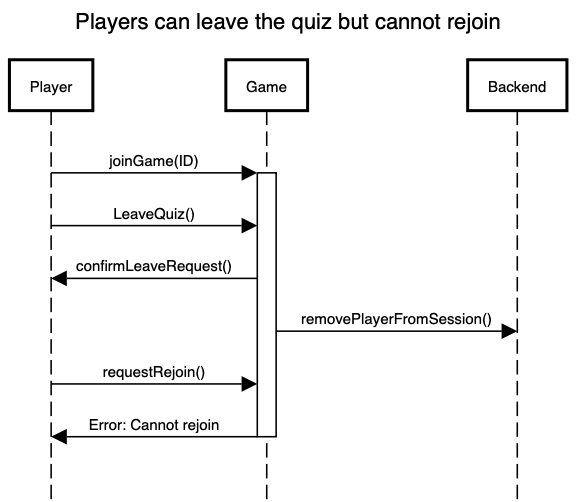
****

* 1. Players can view quiz history for up to 30 days.

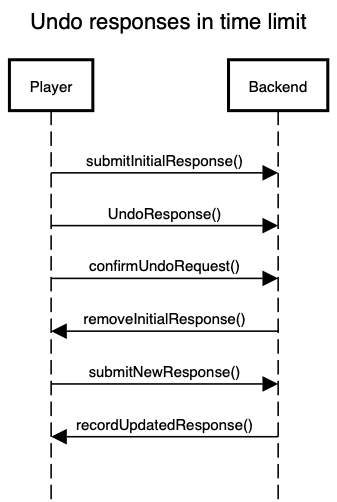
[Does login get any response from the system? Is user getting any feedback]



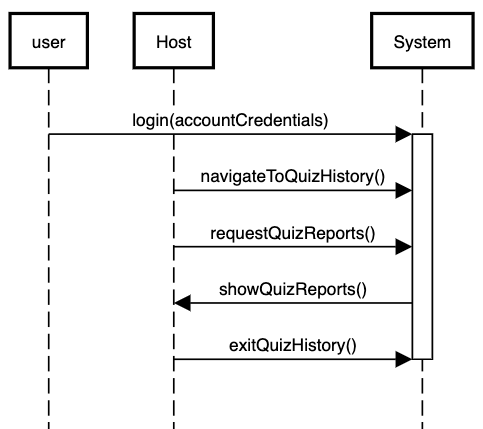
* 1. Players can leave the quiz at any time but cannot rejoin the quiz.



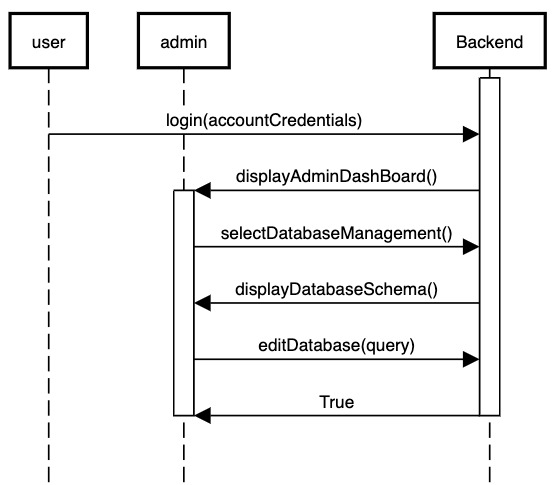
* 1. Players can undo quiz responses under the time limit



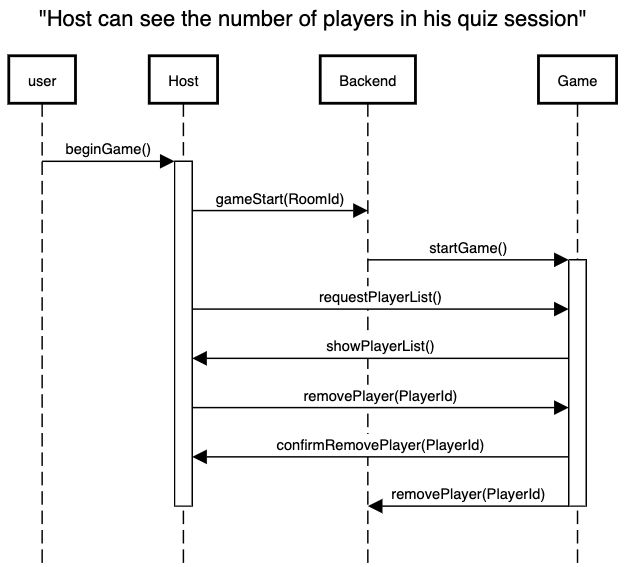
**3.12 Host exports past quizzes and reports to Excel**



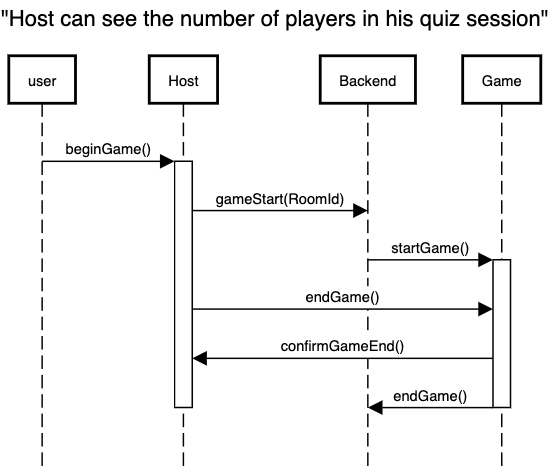
**3.13 Admin edits the data of users**

****

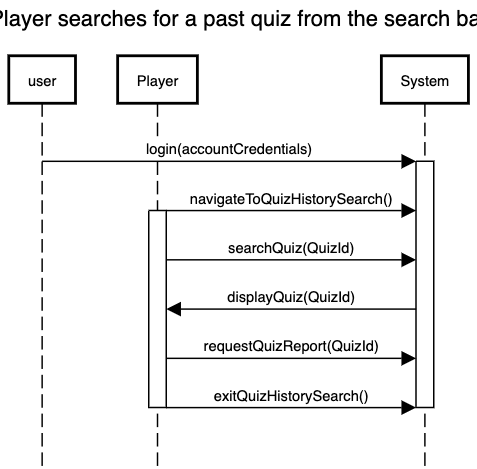
**3.14** **Host removes a certain player from his quiz session**



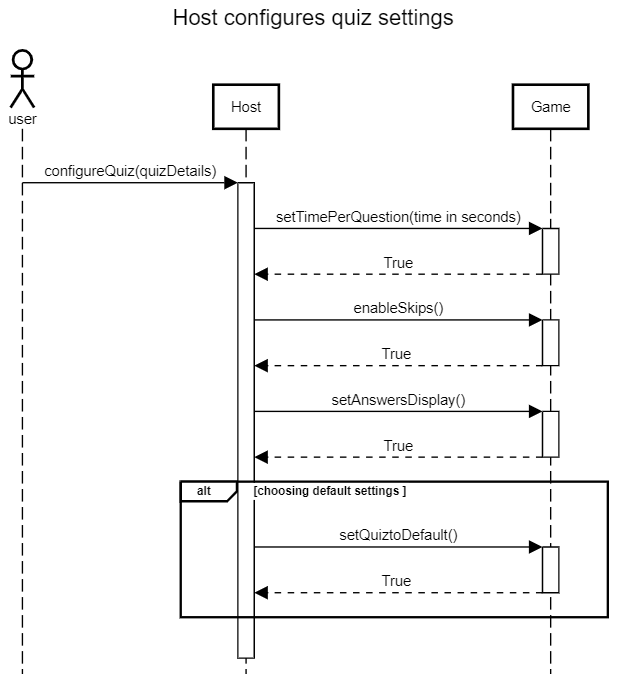
**3.15 Host ends a quiz session arbitrarily**



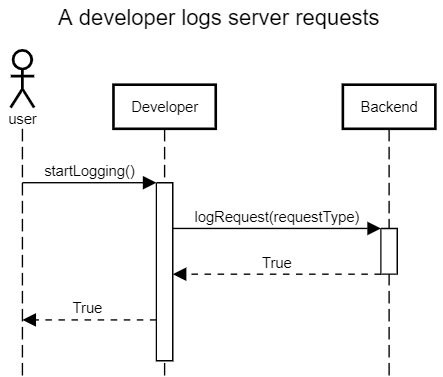
**3.16** **Player searches for a past quiz from the search bar**



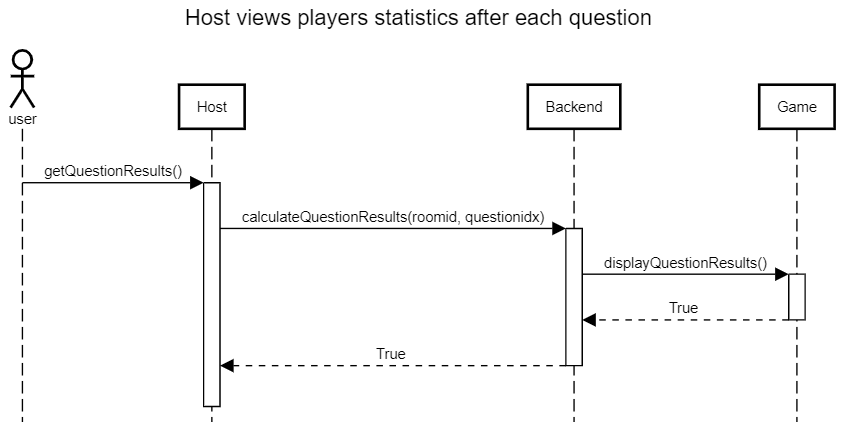
5.17 Host configures quiz settings.



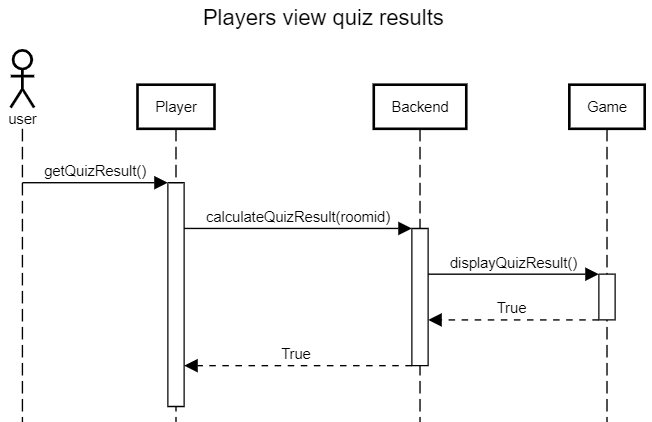
5.18 A developer logs server requests.



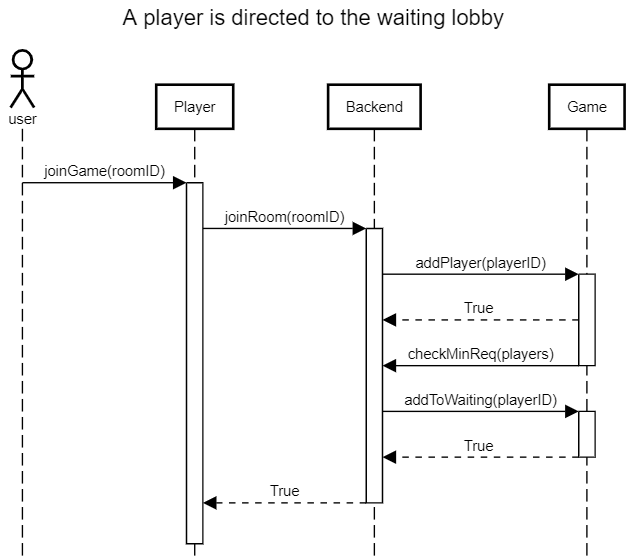
5.19 Host views collective statistics of participants after each question.



5.20 Players view quiz results



5.21 A player is directed to the waiting lobby.



# State Diagrams

## Diagram details

**Lobby**: The game session starts in this state. Players gather and wait for the host to start the game.

**InProgress**: When the host starts the game, the session moves to the "InProgress" state, indicating that the game is actively running.

**Question**: From the "InProgress" state, the game proceeds to the "Question" state, which represents the current question being asked.

**DisplayQuestion**: Inside the "Question" state, there is a sub-state called "DisplayQuestion." This is where the question and answer choices are displayed to all players.

**ShowingQuestion**: After displaying the question, the game transitions to the "ShowingQuestion" state, where players have a chance to read and understand the question.

**Answering**: Players select their answers while in the "ShowingQuestion" state.

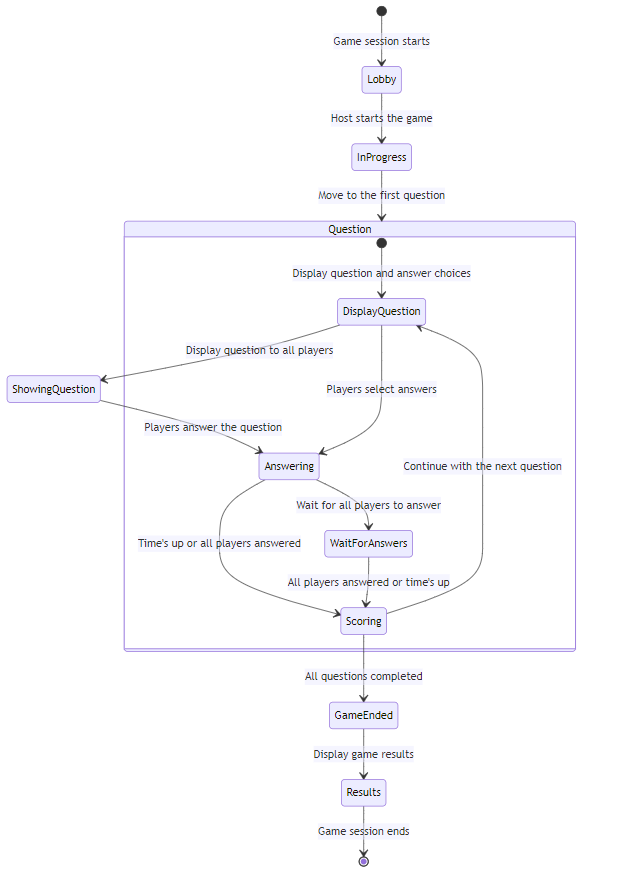
**Scoring**: The game moves to the "Scoring" state when either the time for answering is up or when all players have answered the question. Here, the answers are evaluated.

**GameEnded**: If all questions have been completed, the game transitions to the "GameEnded" state, indicating that the game has finished.

**Results**: In the "GameEnded" state, the game results are displayed to the players

The "Question" state has its own substates, representing the flow of a single question within the game. It starts by displaying the question and answer choices, then transitions to players selecting answers, waiting for all players to answer, and finally, scoring the question. After scoring, the game continues with the next question or ends if all questions have been asked.

## Diagram

****

# Non-functional Requirements / Quality Attributes

| **Sr#** | **Requirements** |
| --- | --- |
| 1 | A Player's response to a question should not take more than 2 seconds to be visible to the Host. |
| 2 | It should not take more than 10 seconds for a Player to join a quiz hosted by a particular host. |
| 3 | The latency should be below 500 ms for question loading and scoring. |
| 4 | The application must support upto 10 simultaneous groups on its initial build. |
| 5 | The web application should work on the latest versions of chrome and safari. [Only?] |
| 6 | The service should be available 90% of the time (0.10 \* 8760 = 876 hours of allowable downtime per year). |
| 7 | The data (quiz questions for host and marks/history for the players) should be retained for at least 30 days on each users’ devices. |
| 8 | The web application should automatically resize to fit onto any small, medium, large screens (i.e. phones, tablets, or laptops) |
| 9 | The application should be scalable such that it should be able to support at least 300 users at the same time (this includes all the hosts and players) without significant degradation of performance. |
| 10 | It should not take more than 5 seconds for the landing page of the website to be visible to a user. |
| 11 | Server should send out responses to Hosts and Players every 1-2 seconds when the question is being answered to ensure the timer for the question for each User is synced |
| 12 | The system can not let more than the MAXIMUM and less than the MINIMUM players in a session. The number can be modified by the Host but a fixed MINIMUM of 2 and a MAXIMUM of 20 is set. |
| 13 | Permanent storage for quizzes and scores is done by registering as a user. The MAXIMUM limit of quizzes is set to 5 per user and scores are recorded to last 10. |
| 14 | Each quiz must have 4 options to the question, one of them being correct. |
| 15 | Quizzes are formatted in CSV form for exporting and importing purposes, with a column for question, columns for all 4 options and the correct answer. |
| 16 | The web application should have fault tolerance such that error in one component should not cause the application to completely crash. User authentication and quiz creation should operate independently, ensuring that issues in one area do not disrupt the core functionality of hosting and participating in quiz sessions. |
| 17 | No Player should be able to enter a quiz session once it starts. |
| 18 | A fallback central server should always be present to allow immediate accessibility. |

# Who Did What?

|  |  |
| --- | --- |
| **Name of the Team Member** | **Tasks done** |
| Hafsa Ahmed | Use Cases 13, 17 to 22, Introduction, Actors |
| Bisma Nawaz | Use cases 6 to 11, sequence diagrams 6 to 11, introduction, actors, use case diagram |
| Abdul Muiz | Use cases 1 to 5 and Sequence Diagram 1 to 5, Class diagram, description and State diagrams, descriptions |
| Abdur Rafae Haroon | Use case 12 to 16, sequence diagrams 12 to 16 |

# Review checklist

|  |  |
| --- | --- |
| **Section** **Title** | **Reviewer Name(s)** |
| Hafsa Ahmed | Use Cases, Intro, Actors |
| Bisma Nawaz | Introduction, actors, state and class diagrams and descriptions, non-functional requirements |
| Abdul Muiz | Use cases 6 to 20, Sequence Diagrams 6 to 20 |
| Abdur Rafae Haroon | Use Cases, Intro, Sequence diagrams |