Proposed Architecture:

**1: Purpose**

This document outlines the overall architecturally significant requirements for this project. Based upon the main NFR’s and the system assumptions/dependencies this document will justify the decisions made and the constraints that come with them. It will also show the overall system view in many forms including class diagrams, activity diagrams and a brief logical view. Basic architectural frameworks are also mentioned to aid in visualizing how this project will be implemented.

**2: Game Design:**

**Overview / Vision Statement:**

Let’s Quiz is a trivia game designed to test your knowledge on a variety of different science fiction based topics. You’ll get points for getting things right but don’t take too long or you’ll run out of time. Compete online against other real people and try to become the top Let’s Quiz player.

**Level Abstract:**

* On round start a timer will be shown displaying the remaining answer time the player has for the round
* A question will be shown and a list of 4 possible answers.
* The user presses an answer and the result highlights green for correct or red for incorrect.
* A certain amount of points is awarded for a correct answer, this is displayed on the screen during the round.
* A new question and 4 possible answers are shown and the process repeats.
* Once the timer ends the round is over and a summary is shown displaying the points earned for that round.

There are no defined levels in Let’s Quiz. For each game there are 3 rounds of play. These rounds do not differ from each other except for the questions that are given. Each round has the same UI design.

**Difficulty**

The difficulty does not change for each round or as gameplay progresses. However, one of our goals is to eventually match players of similar skill levels. So as players earn more points they will play against others with similar points and game experience.

Different categories of questions will be offered to players so in some regards players might struggle in some areas over others. These categories are chosen by the user before a game begins.

Although this does not change, the player is always up against the clock. The more they can answer in a given time the more points they get. This factor should challenge players as they can choose to take it slow or go as fast as possible.

**Gameplay Elements:**

Let’s Quiz does not have a story line or characters. Its main gameplay element is that of the trivia itself. Players will be tested on their knowledge for the variety of topics. The more games a player plays, the more points they will gather. Every player will have access to the variety of global leaderboards to see where they stand against all other players. This factor will keep players coming back as they are up against real other people.

**3: Architectural goals and philosophies**

Overall goal is to create a server based, cross platform mobile application that will allow a multitude of users to be online simultaneously. An additional goal is to include a single player game that allows offline gameplay.

A smaller scope of this would be to see two players play online in a choice quiz game against each other. As well as having a global leaderboard and a multitude of available questions. The game will determine a winner and then update their individual and worldwide leader board. The game also allows players to be a part of multiple games simultaneously, allowing players to start new games or continue existing ones.

Based on the overall goals set out in the vision document the following are the high priority NFR’s. Usability, Reliability, Performance and Maintainability.

Usability is the most important NFR as without a usable system all other NFR’s can’t even be considered. The game is intended to be run on mobile devices and have a UI catered to those users. The game must be intuitive and self explanatory in all degrees of operation.

Reliability is important to be considered due to the game running off a server. The entire operation of the game requires a connection to the server to be reliable. Without a reliable connection, users will not be able to play any of the functionality of the application. (maybe add in redundancy if connection drops from user, will submit next time online)

Performance is still important in regards to this being a mobile application. The app must be tailored to mobile use and be quick to respond to inputs from the user. It must also have quick response times from the server as delays longer than 1-2 seconds is enough for mobile users to stop using the application. Mobile users expect everything to happen instantly.

Maintainability is important as the game must be able to be continued to be upgraded and bugs fixed without hindering the overall functionality of the game. Without being maintainable users could experience large downtimes. By focusing on making the game maintainable the server can be kept up for as much time as possible.

Current Non Functional Requirements:

* Usability
* Reliability
  + Availability
  + Accessibility
  + Stability
* Performance
  + Response Time
  + Throughput
* Maintainability
  + Scalability
* Compatibility
* Persistence/ Data Integrity
* Efficiency

**4: Assumptions and Dependencies**

Assumptions:

There are only a few assumptions in this early stage of the project as there are not many available options in which the product can be delivered. Below are the explicit ones that have been made:

* Our project will be released exclusively on Android and iOS. We are assuming the minimum system requirements of 4.4 (KitKat) for Android and 7.0 for iOS devices.
* In order to allow online gameplay, we are assuming the FTP server will be able to meet our needs using the MySQL database.

Dependencies:

As with any project there are dependencies on both the side of the user and in the project development phase with resources and team members. Below are our current project dependencies.

* Each user must have an internet connection in order to play a multi player game. Although users do not need to be online the entire time they are using the application, a connection to the server must be made at some point to download new game data and to upload results from current games.
* We are depending quite heavily on the availability of the server. If the server is unavailable at the end of a turn, then multiplayer will not function as intended. A backup system is in place to upload game data once an internet connection is made. In saying this our server provider boasts a close to 100% up time.
* Team member’s availability is a dependency which can limit our progress in this project. Each team member has stated their week to week time availability in the team charter. Based on this we have been able to set reasonable goals and stretch goals that should be achievable in the given time.
* Our project has a vast amount of different technologies in play including, Unity, C#, PHP, MySQL, Facebook and Google interaction. We are depending on our team members to deliver on their promises to learn how to use and create services using these technologies. To limit risk, we will deliver Technical Competency Applications to prove we can use required technologies.

**5: Architecturally Significant Requirements**

Listed below is our list of Functional Requirements.

1. The game must be implemented with C# script.

2. The game must be developed in Unity3d.

3. Game must be playable on different operating systems.

4. The game should allow 2/multiple players.

5. The game must display a login scene to the user.

6. Login screen must allow users to play as a guest.

7. Login screen must allow users to register to play

8. Login screen must allow users to login via Facebook.

9. Login screen must allow users to login in with google play services.

10. Once the user has logged in, the main menu screen will be presented.

11. Menu screen must allow player to start a game.

12. Menu screen should allow players to invite, share and like on Facebook.

13. Menu screen must allow player to enter settings menu.

14. Menu screen must allow player to access high scores.

15. The player should be able to start a new game at any time.

16. Questions must be presented to the player.

17. User must be able to choose an answer.

18. The application must be able to determine if the player has selected the correct answer.

19. Points are added for correct answers and deducted for incorrect answers

20. The score of each player must be recorded.

21. The application must have the ability to determine the winner.

22. A congratulatory message should be displayed to the winning player.

23. A list of correct answers will be displayed to the user when the game is over.

24. When the game ends the program should ask the player if a new game should be started.

25. A timer should limit the amount of time of each round.

26. Users can add people to a friends list (either by searching for their username or by adding them at the end of a round)

27. Players will have the ability to challenge people on their friends list

28. Users can vote on their favorite questions

Necessary Requirements for Architecture Realization:

* Game will be developed in Unity using C# and be delivered on iOS and Android in versions KitKat 4.4 and iOS 7.0. Both of these versions offer necessary API’s to complete the application.
* A log in screen allowing users to register, sign in with Facebook or Google Play will be implemented. SDK’s will be used to implement the Facebook and Google Play services which will provide the application with necessary user data to log them in.
* Once a multiplayer game has been created a connection to the FTP server must be established. If no connection possible the device will try again later, notifying the user.
* A MySQL database will be created on an FTP server. PHP script will be used to send commands from the user’s devices to the server to update the database.
* All logic inside the application must be complete. The game must be able to determine correct answers, calculate scores, determine game winners and display the correct answers at the end of game.

**6: Decisions, constraints & justifications**

Decisions:

* We will use Unity as our development environment.
* We will allow users to log in using either a Let’s Quiz account or their existing Facebook or Google accounts.
* The game will be available for both Android and iOS.
* A FTP server will be used to host the MySQL server.

Constraints:

* The game will require an internet connection to run.
* Most of the coding will be done using C# and PHP.
* The database will be in SQL.
* The user interface will have different levels of authentication handled by separate SDK’s.
* Team members individual time allocation for the project.

We are using Unity as our development environment as it is a program that all team members have used before and it is a suitable program for meeting the other decisions made. Unity allows us to create a cross platform application for Android and iOS.

By allowing users to log in using Facebook and Google Play users can authenticate themselves quickly and using services they are familiar. Users may feel safer doing this rather than signing up directly with our server.

None of the architecture being used is specific for either Android or iOS so we are able to develop the same application for both devices. The result of which widens our audience when it comes to delivering and gaining a user base.

We do not require users to join a lobby in order to allow multiplayer functionality. Rather the user plays locally for most of the gameplay except when it comes to updating a current round at its beginning and end. Therefore, we can use a MySQL server that just gets updated with current game information as required. This saves significant resources and skill requirements to allow multiplayer functionality.

Since we are allowing multiplayer we bring in the constraint of requiring an internet connection from the user. However as stated above this internet connection does not need to be consistent but just once at the beginning and end of a round. We will also require functionality to store data that needs to be sent to the server if for some reason an internet connection is not available when required.

Inside of Unity we will use the language of C# to code our application. We need to specify a language to be used and we have chosen this off of the skills of the team members. PHP will be used to communicate commands from the user to the server for the same reason.

Similar to above we have chosen to use a MySQL server due to the fact that team members all have previous experience in implementing.

We require authentication of all users entering into multiplayer. IN order to handle with we will implement both Facebook and Google Play SDK’s to assist in gathering user data. These are the two most popular services that allow authentication and are services that most users still use.

It is known that all team members have many other commitments over the course of this project. All of these constraints have been laid out in the Team Charter. Team members will be held to their promise of time commitment, if not then action will be taken.

**7: Architectural Mechanisms**

Play Game: Two options from this button push. The user can either start a new game or select from an ongoing one. Both options require a connection to the internet.

Submit Question:

Users will be able to submit their own questions to become part of the normal game question rotation.

Multiplayer:

Refers to the multi player functionality allowing players to play against each other. Currently being implemented through a MySQL database.

User Login:

Login using either an email address or the Facebook/Google Play SDK.

GUI:

We will use Unity to design the GUI for display on mobile devices. Will follow standard design principles and not be too dissimilar to other applications of the same nature.

**8: Layers or Architectural framework**

The majority of the application will follow a mobile architecture pattern. Meaning it will be designed specifically for mobile. The GUI will be relevant and use touch input to move through the application. The application will also be optimized to run on a mobile device.

The other part of the application will use a server-oriented architecture model. This model describes application components connecting together through a communication protocol over a wireless network.

**9: Architectural Views**

## Use Case Descriptions

### Use Case: End Goal: Launching Game

When the user

Wants to start the application they click the application icon on their device

So that the application opens to show the login screen

### Use Case: End Goal: Login

When the user

Wants to login, they must choose login option

So that that the application allows login and displays pregame screen

### Use Case: Login Registered users-Sub-function End Goal: Login

When the user

Wants to play by logging in, they then press login

So that that the application opens to the login screen

### Use Case: Login with Facebook-Sub-function End Goal: Login

When the user

Wants to play by logging in with Facebook, they then press Facebook login

So that that the application connects to the Facebook Authentication server and allows login

### Use Case: Login with Google Play Services-Sub-function End Goal: Login

When the user

Wants to play by logging in with Google Play Services, they then press Google Play Services login

So that that the application connects to the Google Play Services Authentication server and allows login

### Use Case: Play as Guest-Sub-function End Goal: Login

When the user

Wants to play without logging in or first registering they press play as guest

So that the application opens to the pre-game screen

### vii) Use Case: End Goal: User Submit Question

When the user

Wants to submit a question they will press the submit question button

So that the application opens to the submit question scene

### viii) Use Case: End Goal: Start a Game

When the user

Wants to start a new game they will press the start new game button

So that the application will either start a new game or join an existing game

### ix) Use Case: Choose Game Mode-Sub-function End Goal: Start a Game

When the user

Wants to Choose game mode they close the game mode by pressing Corresponding mode

So that the application opens the correct game state

### x) Use Case: Continue an existing game-Sub-function End Goal: Start a Game

When the user

Wants to take their turn in a previously started game they press the games description

So that the application opens the correct game state

### xi) Use Case: End Goal: Answer question

When the user

Wants to answer a question they select the correct answer

So that the game can check the answer for correctness

### xii) Use Case: End Goal: Facebook share

When the user

Wants to share game they click share on face book button/link

So that the application connects to the Facebook server and allows sharing

### xiii) Use Case: End Goal: Facebook Challenge/Invite

When the user

Wants to Challenge/Invite they click the challenge button

So that so that the application connects to the Facebook server and sends invitation

### xiv) Use Case: End Goal: Check Leader board

When the user

Wants to check the leader board scores they press the leader board button

So that the application connects to the Google Play Services server to display the leader board

### xv) Use Case: End Goal: Check Achievements

When the user

Wants to check their achievements they press the achievements button

So that the application connects to the Google Play Services server to display their achievements

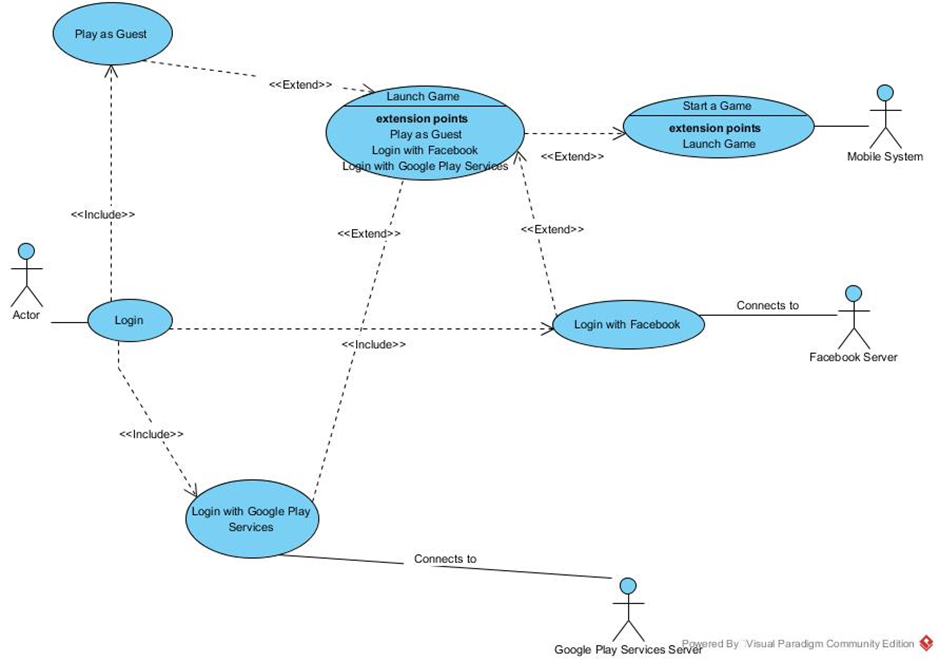
### xvi) Use Case: End Goal: Exit application

When the user

Wants to exit the application they press the back button on android and home button on IOS

So that the application closes down

### Use Case: End Goal: Login



### Use Case: End Goal: Start a Game



### Use Case: End Goal: Answer question



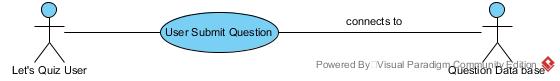
### Use Case: End Goal: Facebook Share and Challenge/Invite



### Use Case: End Goal: Check Leader board/Achievements

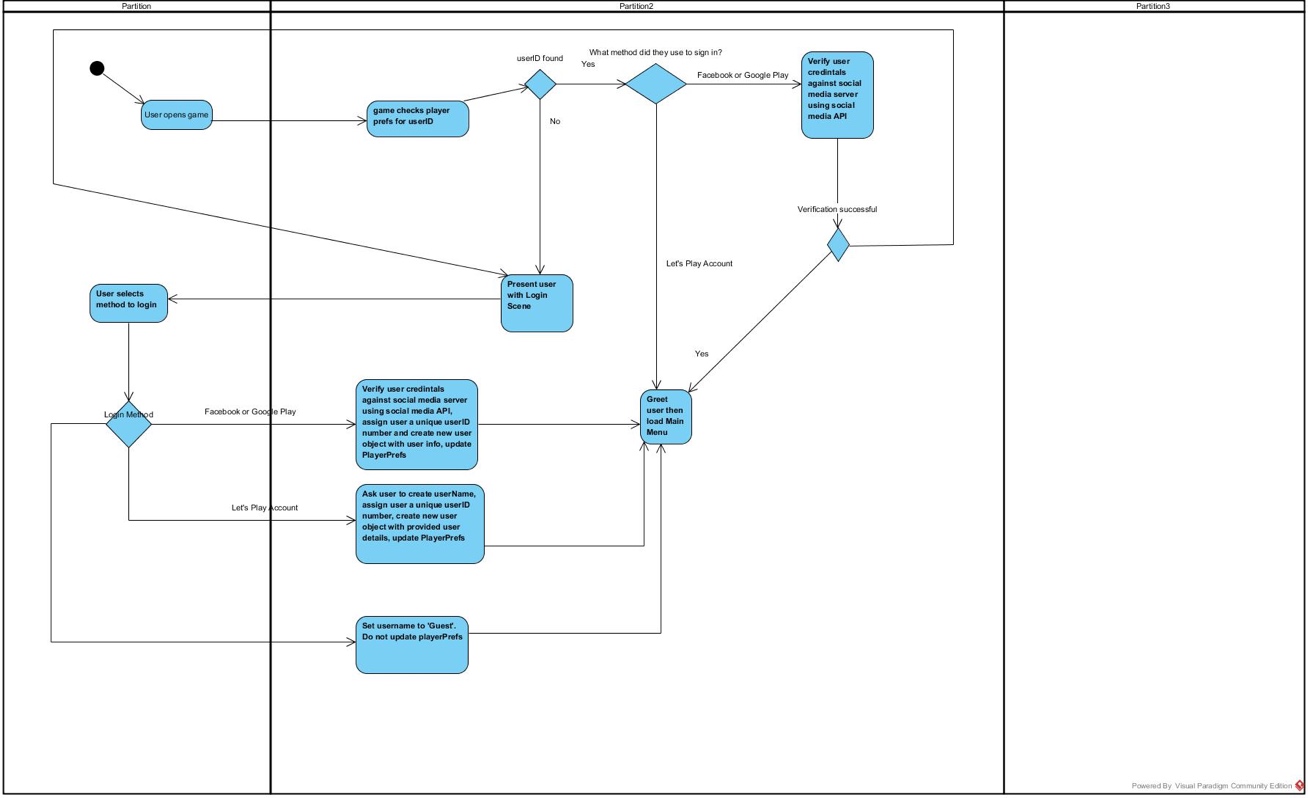


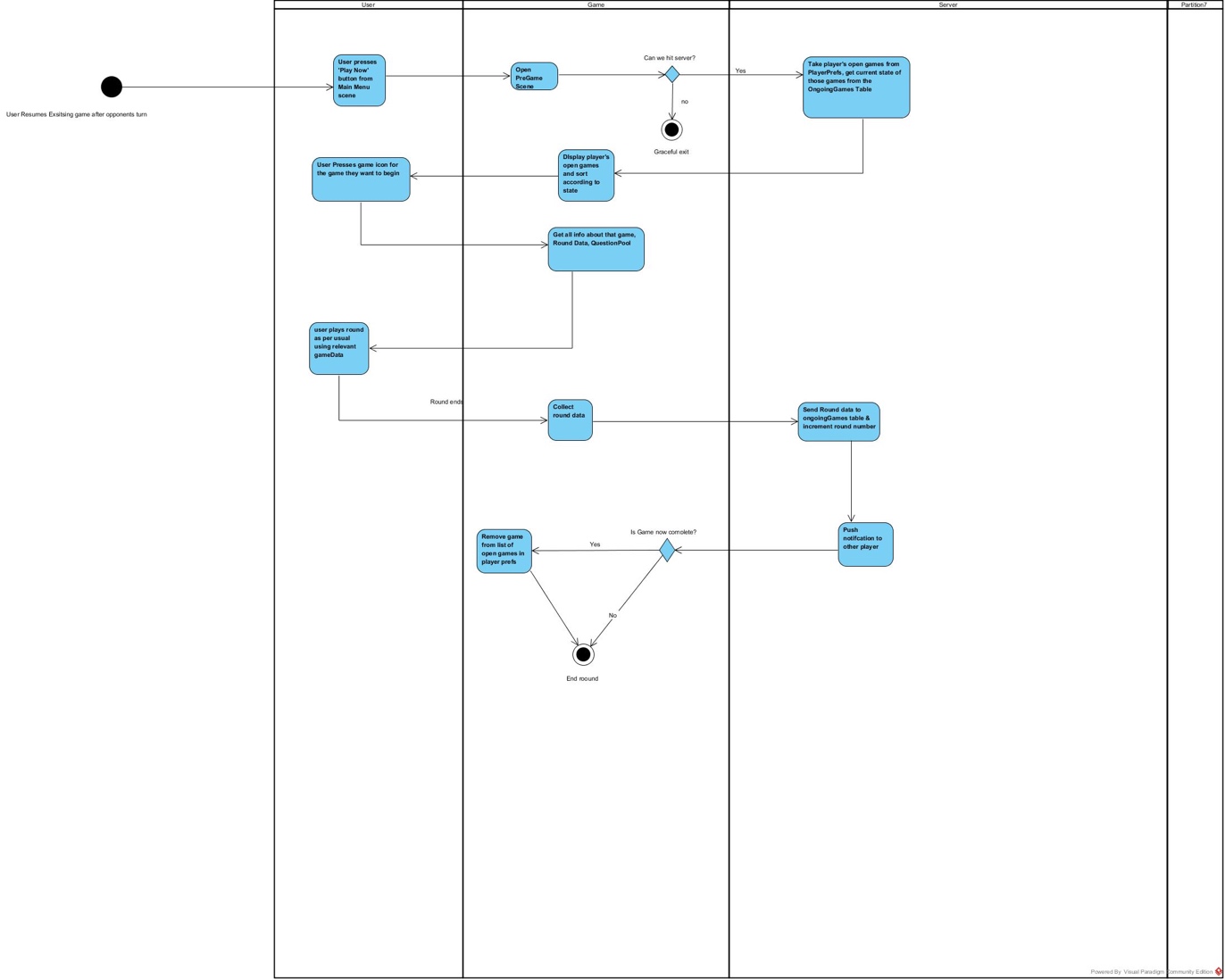
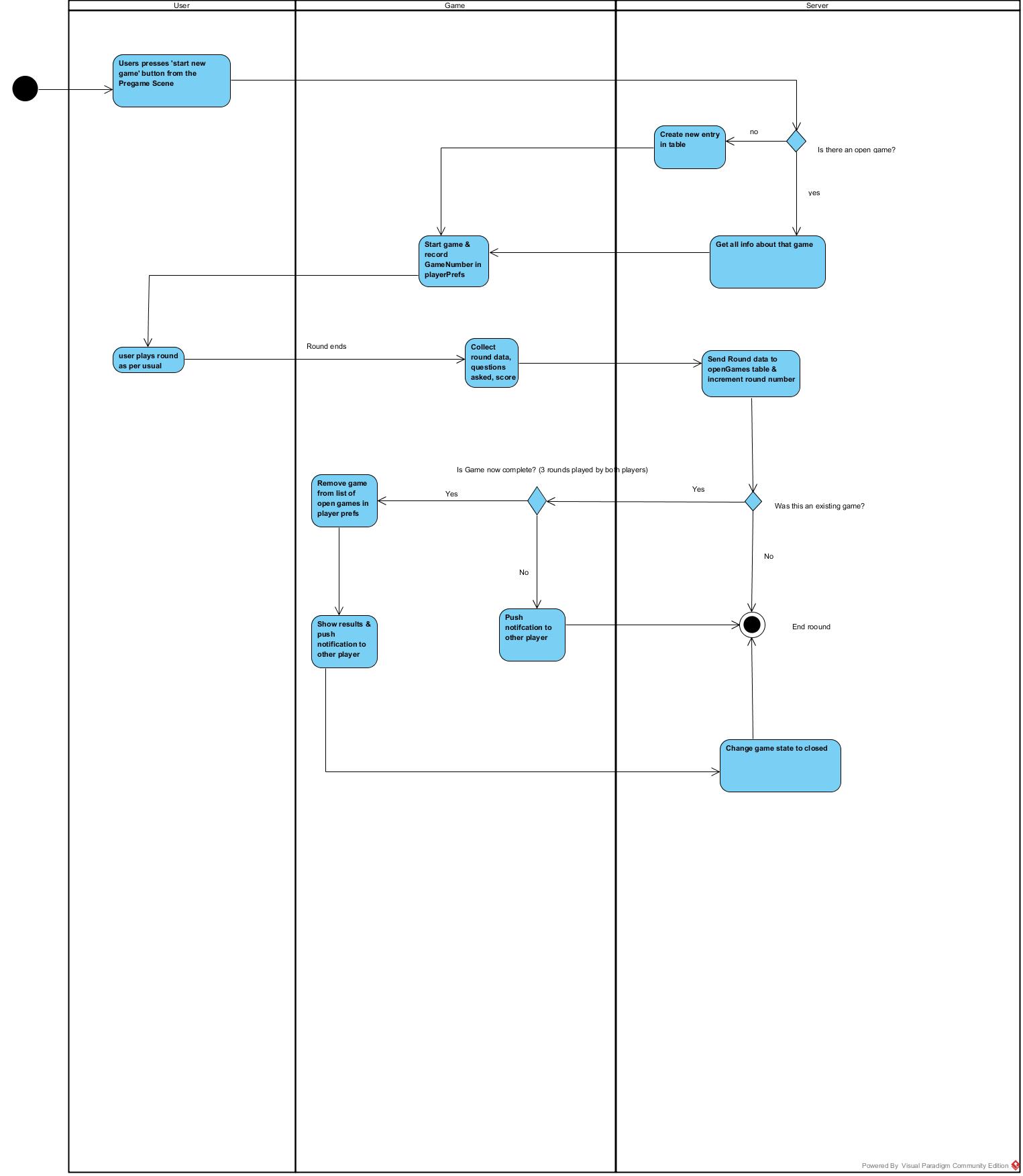
### vi) Use Case: End Goal: User Submit Question



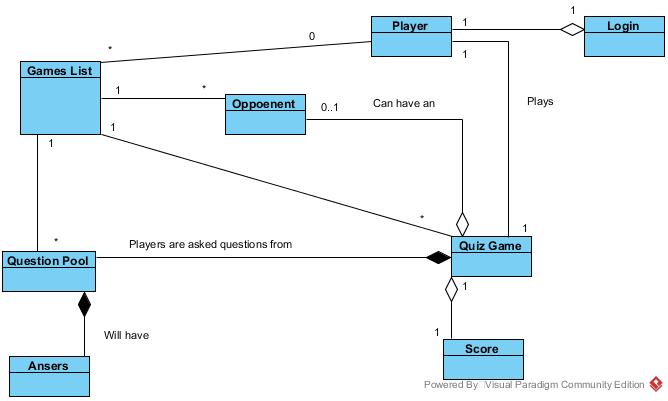
## Activity Diagrams

Open App Activity:



User Resumes Game Activity:User Starts New Game Activity:

## Domain Model



## Logical View

**Important Classes:**

The player class holds the name, email and login id for every user. A player can login, through the authentication services provided (Facebook, Google Play or Let’s Quiz account) or they can skip and play as a guest. Once authenticated a player can play a quiz game. The quiz game here is the multiplayer quiz game. The functionality for a single player game will be slightly different as users will not require an internet connection.

The quiz game becomes part of the games list held on the MySQL database. During gameplay the Quiz game will request a question from the question pool. The current game status after every question will be sent to the MySQL database. The question pool also has a list of answers to all questions which will be made available at the end of the game to both players. The quiz game will create a score class to keep track of player score and high score data. Once a game is finished it is removed from the current games list on the database.

Users will require an internet connection for all updates to the database. This happens at the beginning of a game and at the beginning and end of any question. It is required at the beginning of a question in order to retrieve the result of the other player’s answer. It is required at the end in order to update the database with the player’s response to the question.