Proposed Architecture:

* All goals and philosophies well explained, and completely consistent with high priority NFR’s
* All critical architecturally significant requirements correctly identified and implications explained
* A comprehensive list of decisions and constraints identified. Decisions completely consistent with goals and philosophies, sensible, and well justified with reference to specific needs of project
* A comprehensive list of architectural mechanisms identified. Mechanisms related to architecturally significant requirements.
* Framework/architectural style sensible and completely appropriate to project. All NFR’s addressed.

**Purpose:**

**Architectural goals and philosophies (consistent with NFR’s):**

Overall goal is to create a server based, cross platform mobile application that will allow a multitude of users to be online simultaneously.

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.

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
* Security
* Compatibility
* Persistence/ Data Integrity
* Efficiency

Current 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 add 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

**Assumptions and Dependencies:**

Obvious assumptions. This application is being explicitly built for mobile use on iOS and Android. The backbone of which will be a PHP server. Currently the application is being built using Unity and should be noted as an assumption.

The server is going to be a crucial part of this project, currently we are looking at a personal server from among the team. If this does not suit our needs, then we will rent out a sufficient server.

Skill of team:

Availability of team:

**Architecturally significant requirements:**

I guess this links to the dependencies, so running the PHP server, developing in Unity using C#, developing for both iOS and Android.

**Decisions, constraints & justifications:**

We have chosen Unity as our builder since it uses a language that all of the team are confident with, allows us to create cross platform applications easily and is purpose built for designing the type of application we wish to create.

The decision of going cross platform.

The decision of using a personal server.

Time constraint of team members

**Architectural Mechanisms:**

**Key abstractions:**

* Quiz Up (think that was the name)
* ‘mobile friendly’, quick, low CPU use, pleasing, entertaining
* (what else?)

**Layers or Architectural framework:**

**Architectural Views:**

* Use cases will be provided to describe the main functions of the system. Containing all architecturally significant properties.
* Maybe an overall class diagram?
* Logical view