1. Description of prototype functionality

1.1 Introduction

The application we have developed is based on a mobile app called QuizUp (https://www.quizup.com/en). The main function of this game is to randomly let two online players compete against each other in a quiz game. They will be asked a set of questions from a specific topic they choose and each player will be awarded a score for each correct answer they provide. Each correct answer will award the player 100 points which will contribute towards their final score at the end of the quiz. There are also some other features in the game, such as friend system, offline messaging and game history checking.

1.2 Flow of the game

See the following figure

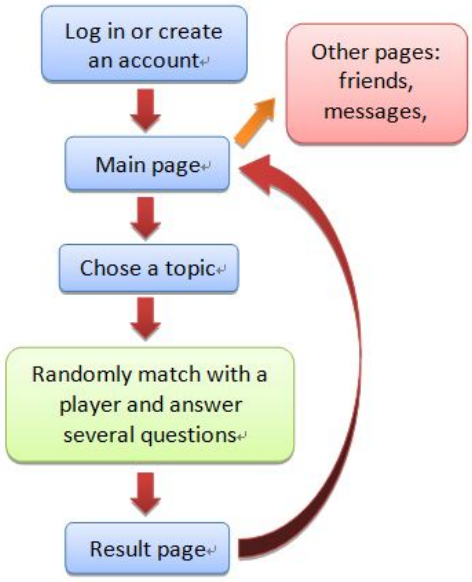

Figure 1: Flow diagram

1.3 Main function and features

- Main page: Link to all main functions pages.
- Friends: You can view your friends list and add/delete a friend.
- Messages: You can view your received messages and sent messages. You can also send messages to others by their accounts or names.
- Setting: Used to modify all of your personal information, including account, password, name, etc.
- Info: Show all your personal information. You can also search for other players information after you have played them in a quiz.
- Game history: Show the results of past games.
- Topics: You can choose the field you would like to play against others.
- Quiz game: Each player will answer a range of 5 questions (may have repeated questions) from their chosen topic. The 5 questions are predetermined before the game starts so both players answer the

same questions. Once a player has finished the quiz they will be sent to the processing page where they will wait for their opponent to finish the quiz. Each correct answer awards the player 100 points while and incorrect answer gives them 0 points.

- Game result: Shows the following results at the end of the game:
  1. Outcome of the game, whether you have won, lost or drawn.
  2. A players current experience after playing the game.
  3. A score breakdown where each player is shown where they were awards points for which question.
  4. A players current game level.

2. List of tools and techniques used

2.1 List of tools

- GitHub: It is a great platform for collaborating. Everyone can see the update of files instantly. It also has a great feature which is version control. So it is easy to recover previous versions from any mistake. All you need to do is to go back to the previous commit.
- Vim / Sublime Text3 / Notepad++/Aptana Studio 3: Everyone has their favorite text editor, and no editor is definitely better than the other. Vim has many hotkeys but is accompanied with a steep learning curve. Sublime text, Notepad++ and Aptana is easier to use and look fancier.
- Chrome / Firefox / Internet Explorer: When developing the app, we need to test our app frequently with the browser. One important point is that we have to make sure our app will work correctly in every kind of browsers.
- Google App Engine

2.2 Languages

- Python: We use Python2.7 in the back-end development. It is used to implement functions which handle all the database queries and data operations.
- HTML: We use HTML in the front-end development. It is used to show the layout of web pages. The user will interact directly with it, such as filling forms, clicking on buttons, etc.
- CSS: It is a style sheet language used for describing the appearance of web pages. We use CSS with HTML for the design of our application.
- JavaScript: We use JavaScript in the front-end development. It is used to dynamically interact with the user. In our application, it is used to work with the Google channel API.
- YAML: It is used for Python app configuration file. The content will specify how URLs correspond to request handlers, and other information, such as application ID, versions, libraries, etc.

2.3 How we co-work

- All the codes are stored in a GitHub repository.
- We created a Facebook group for posting announcement such as everyones progress and meeting time.
- We had regular meetings on campus during the semester.
- A Skype group was created for co-working over the winter holiday.

3. Relevant statistics

3.1 Line of codes (Use the tool LOC) 3.2 Commitments daily progress (Can get it from GitHub)
The following graph shows the commitment progress throughout the project. It shows that the development peaked in late November before dipping in early December (most likely due to the holidays) and finally rising in late December when the deadline was drawing nearer.

Figure 2: Progress chart

3.3 External libraries (Name + Link)

Tool	Source
Jinja2	http://jinja.pocoo.org/docs/dev/
Webapp2	https://webapp-improved.appspot.com/

4. Brief overview of design and implementation, including key design decisions

4.1 Overview of design
We use webapp2 as our main framework. Our application is mainly constructed with three parts:

- Views: Include all the handlers' Python codes, which deal with database queries and all other back end functions.
- Models: All classes of different objects are declared here. They will be used by the handlers when storing data, accessing data, or modifying data.
- Templates: All HTML codes are written here. These template files are used for the layout of web pages. They communicate with the handler. Some web pages return the information inputted by the users to the handlers. Other web pages receive the parameter from handler functions and output the correct information to users.

Besides these three parts, a file called urls.py specifies the corresponding relationships between view and templates. So it is clear that which template every function should handle.

4.2 Implementation of each part

- Jinja2: Jinja2 is like a bridge between handler functions and HTML. As stated above, some web pages need to receive the parameter from handler functions. But there is no way for a pure HTML webpage to receive information from handler functions. Jinja2 can help HTML to do this by setting some variables,

and the handler functions are able to send parameters to those variables. Then HTML can make use of these variables to output information. Jinja2 also has another great feature: inheritance. With this technique, we wrote some general templates, and they can be used by many other templates as bases. This reduced repetitive coding where some elements of the application were present on multiple web pages.

- Google DB Datastore: This application mainly functions by manipulating data a database. Since we are going to deploy the application on the GAE, it is very convenient to use Google Datastore as our database. Datastore is different from traditional relational databases. All queries are served by entities auto-built indexes. So some traditional queries cant be done, such as JOIN. The classes for objects save data as the properties provided by Datastore, such as integer, string, datetime, etc. When creating an entity, we will specify the entitys property values, e.g. name=Tom, password=12345. Then we can access an entity by filtering with property values or by specifying its index. After accessing an entity, we can use the method as creating entities to update an entity.

- Google channel API We use Google channel API to implement the main function of our app, which is the player-matching part. This technique can connect JavaScript clients and keep every user updated without polling. In our application, different players interact with JavaScript clients in their webpages. Then the JavaScript clients will connect to the channel created by the server. If two players are waiting for a game with the same topic, their JavaScript clients will link to the same channel and the game will start. While the game processes, the JavaScript clients listen to the channel for real-time updates.

5. Critical evaluation of the prototype submitted

5.1 Achievements

- Main function – The main function of this application is well implemented. We use Google channel API to implement it. Players can match with remote players and get the results of game simultaneously. This part is the main technical aspect and bugs occurred frequently. So a lot of time was spent implementing and testing it. Now the application is very robust and works well.

- Statistic functions - This includes game results, game histories, and experience systems. Though this part is not hard to implement, it is one of the most important part in every game application. People always gain the feeling of achievements by viewing these statistics.

- Social functions - This includes friends and messages. These functions let you socialize with other people. And the interfaces are simple and easy to use. We believe this kind of functions can make the application more interesting, not just answering questions with unknown people. Whats more, you can also view other peoples profiles and game histories by searching their accounts or names.

- Hidden functions - This includes web pages not accessible by direct links in our application such as add/topic and add/questions. These web pages allow us to populate the Google DB Datastore to fill it with topics and questions. More topics and questions can be added in a future update with ease from these hidden web pages

5.2 Further works

- Timer - A timer in the quiz could make the game more exciting, and could also prevent a player from taking too long to answer. This part should be our first priority if we still have time to improve our work.

- Appearance - Although we already used many CSS to make buttons and pull-down lists in the application, there are still some pages that look a little simple. Some parts are just plain text. If it is a real

commercial product, appearance should be a very important part. Due to time constraints, the design of the product was completed in a shorter time span while the main functions of the application were focused on more was allocated more time to develop.

- Other function - We think it will be more interesting if a user can add topic and question by themselves. However, implementing this feature would require checks to the questions and topics submitted by a user. One such check would be that the questions submitted are correct and are appropriate for the application.

5.3 Conclusion

We have successfully implemented the main application that was discussed at the beginning of the project with slight alterations. Additional features to the game were included in attempts to make the application more appealing and interesting. After developing the application, a lot of time was spent testing the game and fixing bugs, so it is quite a robust application. Further development may have allowed for a better theme/design of the application.