COMP30023 Computer Systems 2019

Project 1: Image Tagger
Due date: 11:59pm on 29 April (Monday), 2019

Prepared by Junhao Gan and Lachlan Andrew

Background. Nowadays, search engines have been an essential part of our lives, e.g., Google and Bing, which have significantly improved our productiveness. There are various types of searching services, among which keyword search is the dominant one, namely searching by keywords. While keyword search works well for documents, pages and websites, it faces a big challenge when searching for images. This is because linking images with keywords is typically difficult for computers as it requires understanding on the semantic of images. For this purpose, tagging images with keywords is one of the most effective ways to help computers understand images' semantic, and this is also the first step to help machines learn (by telling them which is correct and which is wrong). Unfortunately, tagging a large number of images is boring and requires a substantial amount of human power. In this project, you will make the boring tagging process more funny by implementing a **network game server** that tags images as people play. (The idea was invented by the inventor of re-Captcha and Duo-lingo; Google bought the company and shut down the game.)

The Rules of the Game. The game consists of two players and one server, where each of the players can *only* communicate with the server (but the other player). At the beginning, when the two players log on to the server, the server sends both of them a same image. The game then starts; the players submit words or phrases to the server, one at a time, without being told what the other player has inputed. The goal for both the players is to enter a word or a phrase that has been submitted by the other as soon as possible. Obviously, in order to maximise the chance of getting a match, the two players would better to enter words that describe the image well, since it is the only information shared by both of the two. (The image is, therefore, tagged.) Once the goal is achieved, the game ends; the server sends a web page indicating that the game is completed and prompting the players to play again. If both the players agree to play again, the process repeats with a new image.

Project Basic Functionalities. In this project, you are asked to write a program to implement (by **socket programming in C**) the aforementioned **game server** supporting two players from *two different browsers* over HTTP. The program should allow users to configure both the *server IP address* and the *port number*. Specifically, you need to implement the following ***basic*** functionalities of the server, where *the html source files of all the pages will be provided.*

- *The Welcome Page.* The server should be able to accept persistent TCP connections from the players and return a **Welcome Page** (as shown in Figure 1) upon the establishment of the connection. Specifically, the page consists of:
  - a title, i.e., Image Tagger Game,
  - a welcome image,
  - a welcome phrase,
  - a textbox for players to enter their name, and
  - a button with text "submit", by pressing which the name entered in the textbox will be sent to the server by the *http POST* method;

When a player name is submitted, the server should be able to record the name and return the **Main Menu Page** (see Figure 2).

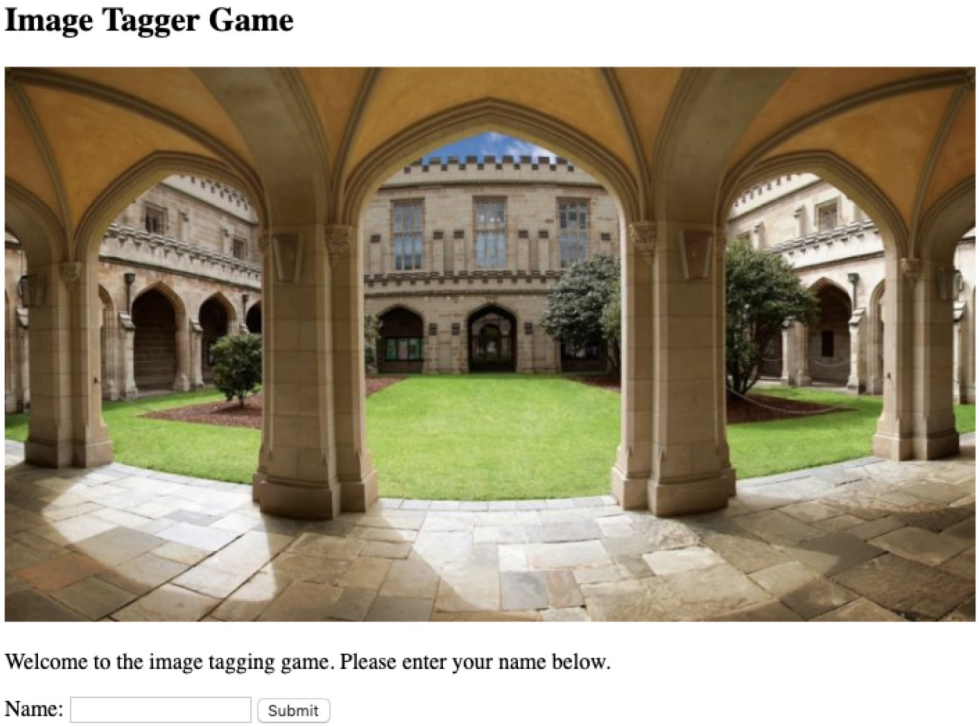

Figure 1: welcome.html

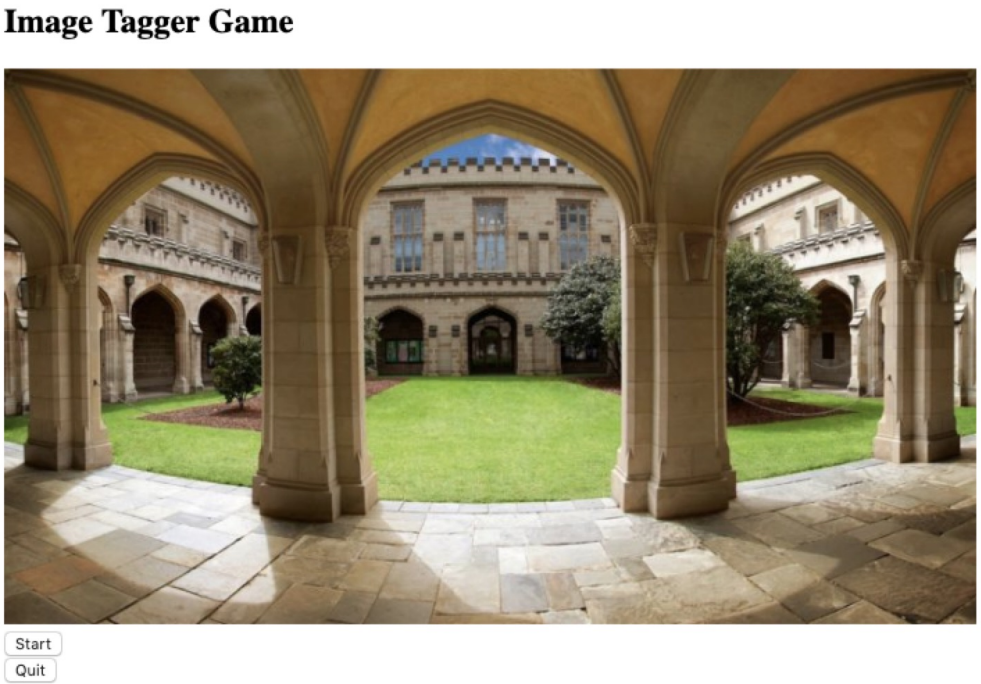

Figure 2: start.html

- *The Main Menu Page.* The Main Menu Page (as shown in Figure 2) has two buttons right below

the image: (i) the "Start" button, and (ii) the "Quit" button.

  - When the "Start" button is clicked, an HTTP GET request is sent to the server. Upon receiving the request, the server returns the **Game Playing Page** (see Figure 3).
  - When the "Quit" button is clicked, the current player quits the game and a POST request is sent to the server. Upon receiving this POST request, the server returns the **Game Over Page** (i.e., Figure 7) to the current player, and to the other player when he/she submits their next keyword.

- *The Game Playing Page.* The **Game Playing Page** (as shown in Figure 3) contains a keyword textbox which is for players to type keywords as their attempts. When the button "Guess" is clicked, the keyword in the textbox is sent to the server (by a POST request). The server should then perform the following actions:

Step 1. Check whether the other player is ready to play. If so, go to Step 2; otherwise, return the **Keyword Discarded Page** (i.e., Figure 4) to the current player indicating that the other player is not ready yet and the keyword just inputed is discarded.

Step 2. Check whether or not the submitted keyword has ever been submitted by the other player. If so, the server returns the **Game Completed Page** to the current player and to the other player when he/she submits the next guess. Otherwise, the server returns the **Keyword Accepted Page** (i.e., Figure 5) to the current player and the game continues.

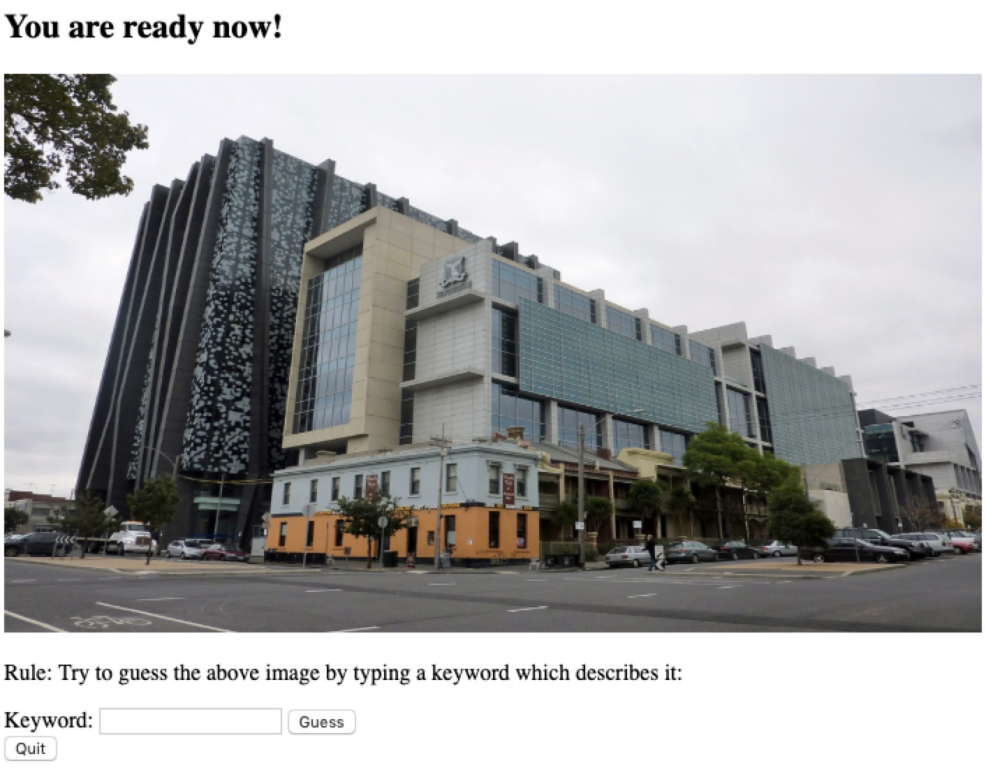

Figure 3: first_turn.html

- *The Keyword Discarded Page.* In the **Keyword Discarded Page** (as shown in Figure 4), when the "Guess" button is clicked, the server performs exactly the same actions as what it does in the **Game Playing Page**.

- *The Keyword Accepted Page.* In the **Keyword Accepted Page** (as shown in Figure 5), when the "Guess" button is clicked, the server performs the same actions as what it does in the **Game Playing Page**, except that the server does not need to check whether the other player is ready or not.
- *The Game Completed Page.* For the **Game Completed Page**, the server performs the same actions as what it does in the Main Menu Page.
- *The Game Over Page.* This page shows "Game Over!". The server closes the TCP connection.

Advanced Functionalities. In order to implement the following advanced functionalities, you will need to "generate" the html files *dynamically* rather than simply using the static html files provided. More specifically, you will need to insert or change some contents in the html files such that it can show information dynamically according to what information the server has received.

- *Advanced Functionality 1: Showing the Inputed Keyword List.* In the **Keyword Accepted Page**, the page should show the list of keywords that have been successfully submitted by the player so far.
- *Advanced Functionality 2: Identifying Players by Cookie.* When a player connects to the server for the first time, the server should be able to create a cookie for the player. Afterward, when a player tries to connect to the server for the second time with a cookie, the server should be able to identify the player by cookie and return directly the **Main Menu Page** *showing the player's name* (therefore, you will need an html file that can change dynamically) without asking the player to submit his/her name in the **Welcome Page**.

Specific Requirements. Below are some specific requirements for the project:

- The game server *must* be implemented by socket programming in C. Programs in other languages will not be accepted and will result in *zero mark*.
- The server *must* use HTTP protocol to communicate with the client browsers.
- A makefile *must* be provided along with your code for compilation, and the compiled executable binary *must* be named as **image_tagger**.
- The **image_tagger** program *must* accept two parameters: (i) a specified server IP address, and (ii) a specified port number.
- When the program is started, it *must* print out the information of the IP address and the port number of the server, as shown in Figure 8.

Hints. In this project, you can assume that everything in the test environment is friendly, namely, there are no adversaries aiming to break down your program, neither unexpected connection drops. In addition, below are some key knowledge that you may need to know to complete the project:

- communicating with a client over a persistent TCP connection by socket programming in C;
- sending an html file to the client browser;

Keyword Discarded. The other player is not ready yet.

Rule: Try to guess the above image by typing a keyword which describes it:

Keyword: Guess

Quit

Figure 4: discarded.html

Keyword Accepted! Keep trying more.

Rule: Try to guess the above image by typing a keyword which describes it:

Keyword: Guess

Quit

Figure 5: accepted.html

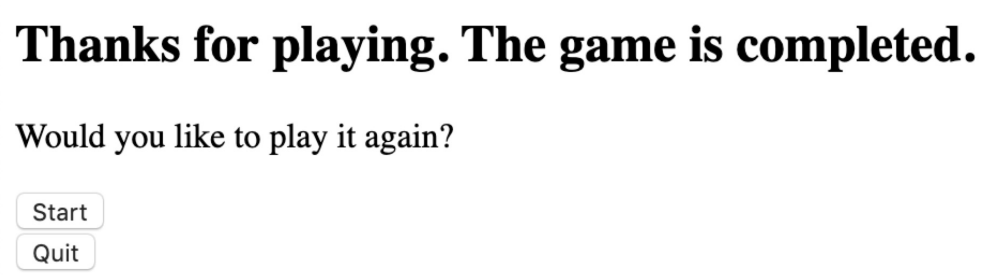

Figure 6: endgame.html

Figure 7: gameover.html

```
$ image_tagger <server_ip> <port_number>
image_tagger server is now running at IP: <server_ip> on port <port_number>
```

Figure 8: Example execution instance. Note: $ is the CLI prompt; <*server_ip*> and <*port_number*> are input parameters.

- *parsing* an http message received from a client browser such that the server can extract the http method (whether it is a GET or a POST) and the contents in the message;
- supporting two connections with two client browsers simultaneously, namely, multiplexing & demultiplexing (some useful materials and good examples can be found here[1]).

An important note: You will have a try on some of the above techniques in the labs in Week 5 and Week 6. You may not want to miss them.

Marking Scheme. This project worths 15% of the subject. The marking scheme is as follows:

[1] http://www.beej.us/guide/bgnet/html/multi/advanced.html

Marks	Task
1	Configurable IP addresses and port numbers
1	HTTP over Persistent TCP Connections
1	Response to the player's name submission
2	Response to the clicking the "Start" button
1	Response to the clicking the "Quit" button
3	Response to the clicking the "Guess" button
2	Displaying the keyword list
1	Identifying player by cookie
1	The use of Gitlab for version control
1	Code Quality
1	Build Quality (e.g., providing a makefile)

Questions. Any questions or doubts should be raised in the LMS Discussion Forum — Project 1. Do not share implementation-specific source code on the discussion forum.

Submitting Your Source Code. The due date is at 11:59pm on 29 April (Monday) 2019. You must submit, to *both* **GitLab** and **LMS**, your source code (with a makefile) in a .zip file with a filename in the the format of *<your_username>*_comp30023_2019_project-1, e.g., junhaog_comp30023_2019_project-1. Any failure to follow the filename format will result in a **2-mark deduction**. Any missing submission from either Gitlab or LMS or both will be considered a submission failure.

Late Submissions. There will be heavy penalties on late submissions. Any late submission will be *deduced* **20% from the total marks** *in the first day*, and *an* **extra 1-mark deduction** *applies to each day passing the deadline.*

Zero Tolerance for Cheating. While you are allowed to discuss the project with your classmates and search online to learn some related techniques to complete the project , you are required to implement the program by your own. This means that every single line of the code must be written by yourself. **All submissions will be checked for plagiarism.** Any confirmed case will receive a 0 mark for the project outright, and be reported to the school for disciplinary actions.