4DN4: Advanced Internet Communications
Lab 2: Online Grade Retrieval Application

Instructor: Dr. Zhao

Demoed: Feb 26, 2020 at 7:10-7:30pm

Devin Jhaveri - jhaverid - 400030315
Rey Pastolero - pastoler - 400020159
Vaibhav Ariyur - ariyurv - 400012535

As a future member of the engineering profession, the student is responsible for performing the required work in an honest manner, without plagiarism and cheating. Submitting this work with my name and student number is a statement and understanding that this work is my own and adheres to the Academic Integrity Policy of McMaster University and the Code of Conduct of the Professional Engineers of Ontario. Submitted by [Devin Jhaveri, jhaverid, 400030315]

As a future member of the engineering profession, the student is responsible for performing the required work in an honest manner, without plagiarism and cheating. Submitting this work with my name and student number is a statement and understanding that this work is my own and adheres to the Academic Integrity Policy of McMaster University and the Code of Conduct of the Professional Engineers of Ontario. Submitted by [Rey Pastolero, pastoler, 400020159]

As a future member of the engineering profession, the student is responsible for performing the required work in an honest manner, without plagiarism and cheating. Submitting this work with my name and student number is a statement and understanding that this work is my own and adheres to the Academic Integrity Policy of McMaster University and the Code of Conduct of the Professional Engineers of Ontario. Submitted by [Vaibhav Ariyur, ariyurv, 400012535]

Server

- Reading Data from CSV
  To read the data from the provided CSV file, the class DictReader from the builtin module csv was used. To store this data, a dictionary was used. This dictionary would contain another dictionary for each row of the csv containing all the details for that student. For the keys of each dictionary entry, the unique hash of the student's ID and password were used. These are created before any client requests are handled.

- Handling Average Requests
  There are various commands that the server waits to receive from the client (i.e. GMA, GL1A, etc.). Each command corresponds to a column in self.data.values (which is a dictionary that contains a dictionary). The for loop under the calculate_average function accesses columns containing the command as the header, adds it to the total, and averages the total over the amount of csv rows (corresponding to each student). The result is then sent back to the client, who will make meaning from the result it received from the client. This means that the server doesn't tell the client "Lab 1 average was X", rather, the server sends result X back to the client and the client knows that it corresponds to lab 1's average based on its previous request.

- Handling Grade Requests
  When the received bytes do not match the bytes of any of the main commands of the server, the server assumes it is a hash and tries to use the received bytes to fetch an entry from the server's dictionary. If the dictionary fails to return an entry for the hash received, the hash then must have an invalid ID or password and thus returns an error message to a client. If the dictionary does return an entry, then the grades are formatted and are sent to the client

Client

- Setup of Client
  The client has 3 main methods for initialization: get_socket(), connect_to_server(), and send_console_input_forever(). Get_socket initializes the iPv4 port to handle TCP connections. Connect_to_server() initializes the connection to the Grade Retrieval Server. Finally, send_console_input_forever() keeps the connection alive. Unlike the server, the client doesn't need a special port allocated to initialize the connection, any random port will be chosen to communicate with the server.

- Handling User Input
  The client has 2 main methods for handling the user input, the first one is get_console_input() that will loop until it gets a non blank input from the user. Once it gets an input, the client will then determine whether it has a "GG" input or another input, using the connection_send() function. From there the client will either go to the normal_send() function for any input besides "GG" or it will go to get_grades_send() for "GG". In normal_send the client will print out which mark in particular its fetching from the server. In get_grades_send() the client will ask for the ID and password of the user and

then encode the ID and password into a hash to send to the server

- Server Communication
  Once the client figures out what input it receives and does the necessary printing and encoding, it will then send off the input to the server and await the server's response. The client will then use the connection_receive() function to wait for the server's response and then print out the result as long as it is not zero bytes, which signals a connection close.