BY: HANNAH FAUS

CHATBOT - CSCE 240

SCOPE

SC District Number: 115

Coding Language Used: C++

This chatbot consisted of five programming projects (P1-P5) and one assembling project (P6). Throughout this presentation I will be giving an overview on my project as well as answering any questions that you may have.

DATA

What data is available and what is retrieved from the program?

1. From the inputted text file (input.txt), the program sorts through the information to retrieve items such as name, address, etc. These items are then outputted to an organized text file (output.txt) with the rep's relevant information.

2. From the command line, the program takes in a user's utterance and matches it to the information in output.txt. The matches are then outputted to the user in the command line. This process continues until the user quits.

CODE ORGANIZATION

These are some of the highlights from my project:

1. In my middle projects P2-P4 I changed from a non dynamic multi-class string search program to a matching program that matched the user utterance with a text file.

2. Most of my code was combined and blended together so P6 didn’t require a lot of code combining. I only had to add the actual file read in portion from P1.

3. In order to dynamically store data I used a lot of vectors throughout my program, especially in P4-P6.

QUERIES SUPPORTED

Mandatory:

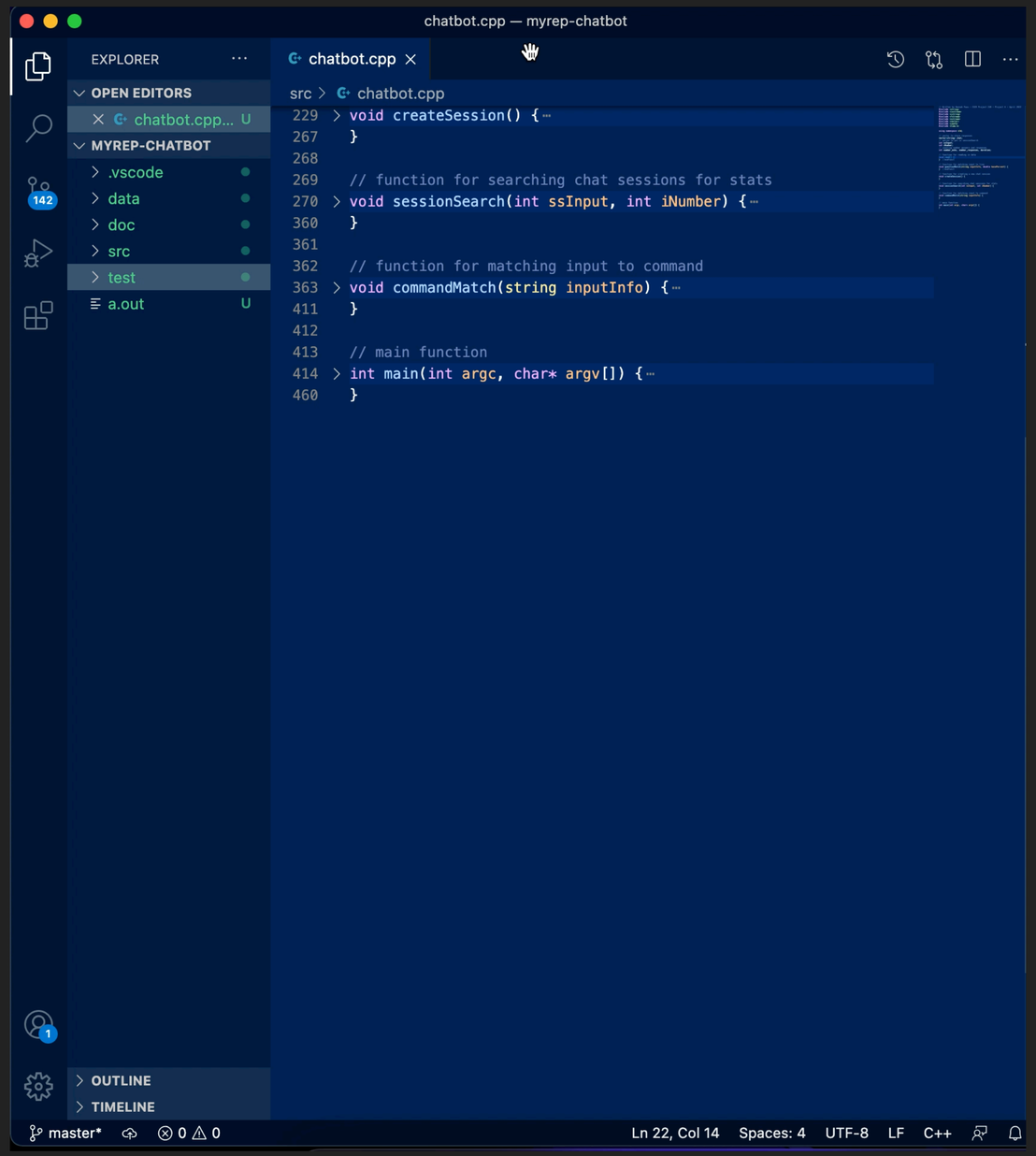
Quit, quit, q
Tell me about the representative
Where does the rep live
How do I contact my rep
What committees is my repo on
Tell me everything
What district do you support for Q/A
<any other text>
-summary
-showchat-summary 2
-showchat 2
-showchat 200

Extra:

Is my rep married
What is their birthday
Where did they go to college

Etc. (based on personal info extracted)

Link to recording:



EXPERIENCE IMPLEMENTING THEN TESTING

Experience with implementing:

All in all, it was difficult learning a new coding language and doing an entire project in it. But, my background in Java helped me to know what I wanted to accomplish it just took a bit more time to execute it.

Experience with testing:

With my unfamiliarity with C++ I did a lot of guess and checking. Luckily, I used block testing techniques so most of the time my testing went smoothly and I was able to accurately pinpoint errors.

QUESTIONS?

P6 read() function

```
21  // function for reading in data
22  void read() {
23      string input_filename = "data/input.txt";  // name of file to be parsed
24      string output_filename = "data/output.txt";  // name of output file
25      string input;  // word that comes in from file
26      string firstName, lastName, nickName;  // parts of name
27      string party, districtNum, districtCounty;  // policital information and affiliation
28      string homeAdd, colaAdd, cellPhone, busPhone;  // addresses and phone numbers
29      string personal;  // paragraph of personal information
30      string jobHistory, residence, repParents, college, family, birthday, unknownPersonal, committee;  // for the different personal info parts
31
32      // beginning of stream - taking in info
33      ifstream in_myfile (input_filename);
34      if (in_myfile.is_open()) {…
105     } else {
106         cout << "Cannot open file." << endl;
107     }
108
109     // beginning of stream - outputting info
110     ofstream out_myfile (output_filename);
111     if (out_myfile.is_open()) {…
129     } else {
130         cout << "File cannot be opened." << endl;
131     }
132 }  //endfunct
```

P6 questionMatch() function

```
134  // function for matching input to line
135  void questionMatch(string inputInfo, double basePercent) {
136      string input_filename = "data/output.txt";  // name of file to be parsed
137      string matchInfo;  // line from input file, output for the end
138
139      // beginning of stream - taking in info
140      ifstream in_myfile (input_filename);
141      if (in_myfile.is_open()) {
142          double maxPercent;  // the current highest match percentage
143          string output = "I do not understand. Please rephrase and try again!";
144          while (getline(in_myfile, matchInfo)) {
145              string response, promptString, inputString; // holds response, placeholder for prompt, placeholder for input
146              double matchCount = 0.1;  // number of words that are in both inputVector and promptVector
147              double wordCount = 0.1; // number of words in inputVector
148              double actPercent = 0.0;  // match percentage = matchCount/wordCount
149
150              vector<string> promptVector;  // holds "buzz" words
151              vector<string> inputVector;  // holds the words from inputInfo
152
153              // looping through matchInfo to separate into promptVector and response
154  >           for (int i = 0; i < matchInfo.size(); i++) {…
177
178              // looping through inputInfo to put into inputVector
179  >           for (int k = 0; k < inputInfo.size(); ++k) {…
193
194              // looking for matches between inputVector and promptVector
195  >           for (int l = 0; l < inputVector.size(); ++l) {…
204
205              // calculate the actPercent (match percentage)
206              actPercent = matchCount / wordCount;
207  >           if (actPercent >= basePercent) {  // if the match percent is above base…
215          } //endwhile
216          // close file and return output, add to chat vector
217          cout << output << endl;
218          chat.push_back(output);
219          ++number_responses;
220          in_myfile.close();
221      } else {
222          cout << "file cannot be opened." << endl;
223      }
224  }  //endfunct
```

P6 createSession() function

```
226  // function for creating a new chat session
227  void createSession() {
228      int seqNumber = 1;
229      bool isCreated = false;
230
231      // creating a new session
232      while (isCreated == false) {
233          string newFileName = "data/chat_sessions/chatSession" + to_string(seqNumber) + ".txt";
234
235          fstream in_myfile (newFileName);
236          if(in_myfile) {
237              ++seqNumber;
238          } else {
239              isCreated = true;
240              ofstream newSession (newFileName);
241              if (newSession.is_open()) {
242                  // looping through chat vector and adding info to new chat session
243                  for (int i = 0; i < chat.size(); ++i) {
244                      string ph = chat[i];
245                      newSession << ph << endl;
246                  }
247              } else {
248                  cout << "New chat session cannot be created." << endl;
249              }
250              newSession.close();
251          }
252      }
253
254      // adding session to .csv file
255      string csvFileName = "data/chat_statistics.csv";
256      ofstream newCsvSession;
257      newCsvSession.open(csvFileName, ios::app);
258      if (newCsvSession.is_open(), ios::app | ios::binary) {
259          newCsvSession << seqNumber << "," << number_asks << "," << number_responses << "," << duration << "\n";//<< "," << number_asks << "," << number_responses << endl;
260      } else {
261          cout << "CSV file cannot be opened." << endl;
262      }
263      newCsvSession.close();
264  }
```

P6 sessionSearch() function

```
266  // function for searching chat sessions for stats
267  void sessionSearch(int ssInput, int iNumber) {
268      // open the chat_statistics.csv
269      string input_filename_csv = "data/chat_statistics.csv";
270      string input_filename_session = "data/chat_sessions/chatSession" + to_string(iNumber) + ".txt";
271      string token;
272      string line, word;
273      vector<string> result;
274      string sSession, sAsks, sResponses, sDuration;  // string version
275      int iSession, iAsks, iResponses, iDuration;  // int version
276      // initializing to 0
277      int tSession = 0;
278      int tAsks = 0;
279      int tResponses = 0;
280      int tDuration = 0;
281      // initializing to empty string
282      string one_output = "";
283      vector<string> searchVector;
284
285      // entering file
286      fstream in_myfile_csv (input_filename_csv);
287      if (in_myfile_csv.is_open()) {
288          while (getline(in_myfile_csv, line)) {…
316          if (ssInput == 1) {…
326          } else if (ssInput == 2) {…
335          } else if (ssInput == 3) {…
351          ++number_responses;
352          in_myfile_csv.close();
353      } else {
354          cout << "csv file could not be opened." << endl;
355      }
356  }
```

P6 commandMatch() function

```
358  // function for matching input to command
359  void commandMatch(string inputInfo) {
360      // variables for command and chat number
361      string sNumber, ph;
362      int iNumber;
363      int numSummary = 0;
364      int numShowchat = 0;
365      vector<string> commandVector;
366
367      // putting string into vector
368      for (int i = 1; i < inputInfo.length(); ++i) {…
380
381      // figuring out what is being asked
382      for (int j = 0; j < commandVector.size(); ++j) {…
392
393      // different command word combos
394      if ((numSummary == 1) && (numShowchat == 1)) {…
397      } else if ((numSummary == 1) && (numShowchat == 0)) {…
401      } else if ((numSummary == 0) && (numShowchat == 1)) {…
404      } else {
405          cout << "Invalid command entered. Please try again." << endl;
406      }
407  }
```

P6 main() function

```
409  // main function
410  int main(int argc, char* argv[]) {
411      // for time duration
412      time_t start, end;
413
414      // read function
415      read();
416
417      // ask for what the desired information is
418      cout << "Hello! Welcome to the district chatbot!" << endl;
419      cout << "If you would like to exit the chatbot at any time please type Quit, quit or q" << endl;
420      cout << "What can I help you with today?" << endl;
421
422      // what the desired information is
423      string inputInfo;
424      double basePercent = 0.20;
425      getline(cin, inputInfo);
426      time(&start);
427
428          // infinite loop for user query
429  >       while (inputInfo != "Quit" && inputInfo != "q" && inputInfo != "quit") {…
445          // closing remark
446          cout << "Thank you for using the chatbot. Have a nice day!" << endl;
447
448          time(&end);
449          double time_taken = double(end - start);
450          setprecision(5);
451          duration = time_taken;
452
453          createSession();
454  }
```