College of Computer Studies
De La Salle University
CCPROG2 Machine Project | Term 2 AY '22-'23
(from February 8, 2023 of 12 AM to April 11, 2023 of 8 AM)

Vaccination Registration with Chatbot App

Coronavirus (COVID-19) is an infectious disease that changed almost all aspects of our lives. According to the World Health Organization (2020), the COVID-19 pandemic has led to a dramatic loss of human life worldwide and presents an unprecedented challenge to public health, food systems and the world of work. Global efforts to control the spread of COVID-19 included travel bans, border closures, and severe public health regulations such as physical separation requirements and mask prohibitions. Complementing these strategies, several vaccines for SARS-CoV-2 have been developed, substantially mitigating the severity of COVID-19 and potentially reducing its transmissibility.

The best way to prevent and slow down transmission is to be well informed about the disease and how the virus spreads. Protect yourself and others from infection by staying at least a meter apart from others, wearing a properly fitted mask, washing your hands, or using an alcohol-based rub frequently and get vaccinated. Fortunately, we have found the vaccine, in fact, a good number of them. However, a lot of people have their own perceptions in getting the vaccine, the mere reason why there are people who opted not to be vaccinated. In the early months of 2021, COVID-19 vaccines began to roll out and the public was thrust into a vaccine registration system that aims to monitor the number of people who are already vaccinated to protect them from acquiring the virus.

For your project, you will create a program that facilitates the registration, schedule vaccination appointments, answer frequently asked questions and manage app data. Please be reminded that the program should manage at most 100 records. The following are descriptions of what the program should contain:

Main Features and Sub-features:

The project must have two primary menu options once your program has been launched. The first option is the ***Vaccination Registration Menu*** for the user who wants to register their vaccination details, set a vaccination appointment and ask questions using a basic chatbot. The other option is the ***Data Management Menu*** to access the administrator's menu to manage data for the Vaccination Registration with Chatbot App.

Vaccination Registration Menu

Vaccination registration menu is composed of four main sub-menu options: *User Registration, Vaccination Appointment, Chatbot FAQs and Exit.* For new applications, the user should choose the User Registration option and the Vaccination Appointment option is for users who have already created their user account and registered their vaccination.

1. ***User Registration -*** Once this option is chosen, the user will be asked to input all the necessary information about the user. For simplicity, the userID is also given by the user as input. However, this userID should be unique (that is, no other such userID exists in the list of users). After registering, the user is redirected back to the Vaccination Registration System menu. Don't forget to prompt the user of their Successful Registration.

   To Register as a user, your program should accept the following data: the UserID, password, name, address, contact, sex, first dose, first dose vaccine and first dose location. Other details such as second dose, second dose vaccine, second dose location, booster dose, booster dose vaccine and booster location are optional.

Data	Data Type	Example
userID	Numeric	1001
password	String(maximum of 10 characters)	******

name	String (maximum of 20 characters)	Juan Dela Cruz
address	String(maximum of 30 characters)	2401 Taft Ave, Malate, Manila
contact	Numeric	09159253949
sex	String(maximum of 10 characters)	Male/Female
first dose	String (maximum of 10 characters)	2023-02-16 (YYYY-MM-DD)
first dose vaccine	String (maximum of 10 characters)	Moderna
first dose location	String (maximum of 10 characters)	Manila
second dose (Optional)	String (maximum of 10 characters)	2023-02-16 (YYYY-MM-DD)
second dose vaccine (Optional)	String (maximum of 10 characters)	Moderna
second dose location (Optional)	String (maximum of 10 characters)	Makati
booster dose (Optional)	String (maximum of 10 characters)	2023-02-16 (YYYY-MM-DD)
booster dose vaccine (Optional)	String (maximum of 10 characters)	AstraZeneca
booster dose location (Optional)	String (maximum of 10 characters)	Makati

* Don't forget to apply proper validations to the required fields.

2. Vaccination Appointment

When a user registers successfully, they can log in with their userID and password. Your program should give the user three unsuccessful attempts before terminating the program. The Vaccination Appointment menu has two sub-menus: *Appointment Request* and *Manage Appointment* menu with the following specifications:

a. ***Appointment Request*** **–** The Appointment Request menu allows the user to select a schedule for their vaccination. Again, for simplicity, the appID is also given by the user as input. However, this appID should be unique (that is, no other such appID exists in the list of appointments). Your program should ask the user to select the location, vaccine brand, date and time for the covid 19 vaccination.

Your program should accommodate the booking of vaccination appointments by asking the user to enter the following details.

Data	Data Type	Example
appID	Numeric	1001
name	String (maximum of 20 characters)	Juan Dela Cruz
location	String(maximum of 10	Manila

	characters)	
vaccine	String (maximum of 10 characters)	Pfizer
date	String (maximum of 11 characters)	2023-02-16 (YYYY-MM-DD)
time	String(maximum of 6 characters)	14:00 (24-hour format)
dose	String (maximum of 10 characters)	2023-02-16YYYY-MM-DD)

a. ***Manage Appointment Menu–*** Furthermore, your program should allow the user to manage their appointment by choosing the **Manage Appointment** sub-menu if they want to cancel their appointment, reschedule the date and time or change the vaccination center location and their preferred vaccination brand. The program goes back to the Vaccination Appointment menu after saving the changes in the appointment.

b. ***Exit Menu-*** The exit option allows the user to quit the Vaccination Registration Menu. The program goes back to the Vaccination Registration Menu.

3. Chatbot (FAQs)

This menu allows the user to ask FAQs (Frequently Asked Questions) to a simple chatbot related to COVID 19. Your program will ask the user to enter a question as a string, then compares the string from the content of the chatbot text file. You can **compare the two strings lexicographically** using the strcmp() function from string.h, which simply means that it compares each character in sequence starting with the first character until the characters in both strings are equal or a NULL character is found. If a match is found, the answer will be displayed on the screen. However, if the match is not found, the user will be prompted with "Sorry, I don't know the answer. Please type another question". This will allow the user to enter another question or exit the chatbot feature and go back to the main menu. Note that format of the text files are listed under Exit menu of the Data Management Menu.

Refer to the table below for the question and answer specification of the chatbot feature:

Data	Data Type	Example
Question	String (maximum of 80 characters)	Is a vaccine mandatory?
Answer	String (maximum of 80 characters)	No, but highly recommended by the government.

Sample Run:

Hi, this is VacciBot, how may I help you?

>>Is vaccination mandatory?
bot>>No, but highly recommended by the government

>>What are the possible side effects of vaccination?
bot>>The possible side effects include pain, fever, fatigue and headache.

4. **Exit**
   The exit option will just allow the user to quit the Manage Data menu. Only those exported to files are expected to be saved. The program goes back to the Main Menu.

Data Management Menu

Data management menu is for the administrator who will manage the Vaccination Registration with Chatbot Application. To grant access to the module, the user must log-in using their ***userID*** and ***password*** (you should have all the characters to asterisks) . Further, the user is granted with three attempts to log-in, otherwise the program will be terminated. For verification, he is asked to input the password. Upon successful log-in, the following features are available:

1.0 User - This menu will allow the admin to do following:

1.1 **Add New User** – Add new user details that includes userID, name, address, contact, sex, first dose, first dose vaccine, first dose location, second dose, second dose vaccine, booster dose and booster dose vaccine. Your program will ask the admin if they want to add another user or go back to the User menu.

1.2 **View all Users.** View all the users arranged by user ID. The display should be in table format in the following sequence: userID, name, address, contact, sex, first dose, first dose vaccine, first dose location, second dose, second dose vaccine, booster dose and booster dose vaccine. The program goes back to the User menu after viewing the User details.

1.3 **Edit User Details** - Edit new user details that includes userID, name, address, contact, sex, first dose, first dose vaccine, first dose location, second dose, second dose vaccine, booster dose and booster dose vaccine. Your program will ask the admin if they want to edit another user details or go back to the User menu.

1.4 **Delete User** – Delete user details from the text file. Your program will ask the admin if they want to delete another user details or go back to the User menu.

1.5 **Exit** – The exit option allows the user to quit the User menu. The program goes back to the Data Management Menu.

2.0 Appointment - This menu will allow the admin to do following:

2.1 **Add New Appointment** – Add new appointment details that includes appID, name, location, vaccine, date and time. Your program will ask the admin if they want to add another appointment or go back to the Appointment menu.

2.2 **View all Appointments** - View all the appointments by the users. The display should be in table format in the following sequence: appID, name, location, vaccine, date and time. The program goes back to the Appointment menu after viewing the Appointment details.

2.3 **Edit Appointment** –Edit the details of the appointment that includes appID, name, location, vaccine, date and time. Your program will ask the admin if they want to edit another appointment details or go back to the Appointment menu.

2.4 **Delete Appointment** – Cancel an appointment by deleting it from the list of appointments. Your program will ask the admin if they want to delete another appointment or go back to the Appointment menu.

2.5 **Exit** – The exit option allows the user to quit the Appointment menu. The program goes back to the Data Management Menu.

3.0 Chatbot - This menu will allow the admin to do following:

3.1 **Add New Question and Answer -** Add vaccination FAQ question and answer in the chatbot text file. Your program will ask the admin if they want to add another question and answer or go back to the Chatbot menu.

3.2 **View all Questions and Answers** - This feature shows all the questions and answers for the chatbot. The program goes back to the Chatbot menu after viewing the Chatbot questions and answers.

3.3 **Edit Questions and Answers** – This will allow the admin to edit questions and answers in the chatbot text file. Your program will ask the admin if they want to edit another question and answer or go back to the Chatbot menu.

3.4 **Delete Questions and Answers** – This will allow the admin to delete questions and answers in the chatbot text file. Your program will ask the admin if they want to delete another question and answer or go back to the Chatbot menu.

3.5 **Exit** – The exit option allows the user to quit the Chatbot menu. The program goes back to the Data Management Menu.

4.0 **Export** – Your program should allow all data to be saved into a text file for future references. The admin is allowed to specify the filename. The filename have at most 30 characters including extension. If the file exists, the data will be overwritten. The sample content of the text files are given after the exit menu. Make sure to follow the format.

5.0 **Import** – Your program should allow the data stored in the text file to be added to the list of entries in the program. The user should be able to specify which file (input filename) to load. If there are already some entries added (or loaded previously) in the current run, the program shows one entry from the text file and asks if this is to be added to the list of entries (in the array). If yes, it is added as another entry. If no, this entry is skipped. Whichever the choice, the program proceed to retrieve the next entry in the file and asks the user again if this is to be included in the array or not, until all entries in the file are retrieved. The sample content of the text files are given after the exit menu. Make sure to follow the format.

6.0 **Exit Menu** - The program saves into text file all current data on set of users and set of schedules. Then the program terminates properly.

The users should be saved in **Users.txt** with the following format:

```
<user1 id><space><password><new line>
<name><new line>
<address><new line>
<contact number><new line>
<sex><new line>
<location1><space><date1><space><vaccine1><new line>
<location2><space><date2><space><vaccine2><new line>
<location3><space><date3><space><vaccine3><new line>
<new line>
<user2 id><space><password><new line>
<name><new line>
<address><new line>
<contact number><new line>
<sex><new line>
<location1><space><date1><space><vaccine1><new line>
<location2><space><date2><space><vaccine2><new line>
<location3><space>><date3><space><vaccine3><new line>
<new line>
...
<userN id><space><password><new line>
<name><new line>
<address><new line>
<contact number><new line>
<sex><new line>
<location1><space><date1><space><vaccine1><new line>
<location2><space><date2><space><vaccine2><new line>
<location3><space><date3><space><vaccine3><new line>
<new line>
<eof>
```

The appointment details should be saved in **Appointment.txt** with the following format: `<appointment1 id><space><user id><new line>`

```
<name><new line>
<location><new line>
<vaccine><new line>
<date><space><new line>
<time><new line>
<dose><new line>
```

```
<new line>
<appointment2 id><space><user id><new line>
<name><new line>
<location><new line>
<vaccine><new line>
<date><space><new line>
<time><new line>
<dose><new line>
<new line>
...
<appointmentN id><space><user id><new line>
<name><new line>
<location><new line>
<vaccine><new line>
<date><space><new line>
<time><new line>
<dose><new line>
<new line>
<eof>
```

The chatbot questions and answers should be saved in Chatbot.txt with the following format:

```
<question1><new line>
<answer1><new line>
<question2><new line>
<answer2><new line>
...
<questionN><new line>
<answerN><new line>
<eof>
```

Bonus

A **maximum** of **10 points** may be given for features **over & above** the requirements, like (1) Generating of pertinent reports (2) meaningful chatbot convo; or other features not conflicting with the given requirements or changing the requirements) subject to **evaluation** of the teacher. **Required features** must be **completed first** before bonus features are credited. Note that use of conio.h, or other advanced C commands/statements may **not** necessarily merit bonuses.

Submission & Demo

Final MP Deadline: April 11, 2023 (M), 0800 via Canvas. After the indicated time, no more submissions will be accepted, and grade will automatically be 0.

Requirements: Complete Program

- Make sure that your implementation has considerable and proper use of arrays, strings, structures, files, and user-defined functions, as appropriate, even if it is not strictly indicated.
- It is expected that each feature is supported by at least one function that you implemented. Some of the functions may be reused (meaning you can just call functions you already implemented [in support] for other features. There can be more than one function to perform tasks in a required feature.
- Debugging and testing was performed exhaustively. The program submitted has
    a. NO syntax errors
    b. NO warnings - make sure to activate -Wall (show all warnings compiler option) and that C99 standard is used in the codes.
    c. NO logical errors -- based on the test cases that the program was subjected to

Important Notes:

1. Use **gcc -Wall** to compile your C program. Make sure you **test** your program completely (compiling & running).
2. Do not use brute force. Use **appropriate conditional** statements **properly**. Use, **wherever appropriate**, **appropriate loops** & **functions properly**.
3. You **may** use topics outside the scope of CCPROG2 but this will be **self-study**. Goto *label*, exit(), break (except in switch), continue, global variables, calling main() are **not allowed**.
4. Include **internal documentation** (comments) in your program.
5. The following is a checklist of the deliverables:

Checklist:

❒ Upload via AnimoSpace submission:

- ❒ source code*
- ❒ test script**
- ❒ sample text file exported from your program

❒ email the softcopies of all requirements as attachments to **YOUR** own email address on or before the deadline

Legend:

Source code exhibit readability with supporting inline documentation (not just comments before the start of every function) and follows a coding style that is similar to those seen in the course notes and in the class discussions. The first page of the source code should have the following declaration (in comment) [replace the pronouns as necessary if you are working with a partner]:

```
/*************************************************************************************************
This is to certify that this project is my own work, based on my personal efforts in studying and applying the concepts
learned. I have constructed the functions and their respective algorithms and corresponding code by myself. The
program was run, tested, and debugged by my own efforts. I further certify that I have not copied in part or whole or
otherwise plagiarized the work of other students and/or persons.
                                                        <your full name>, DLSU ID# <number>
*************************************************************************************************/
```

Example coding convention and comments before the function would look like this:

```
/* funcA returns the number of capital letters that are changed to small letters
   @param strWord - string containing only 1 word
   @param pCount - the address where the number of modifications from capital to small are placed
   @return 1 if there is at least 1 modification and returns 0 if no modifications
   Pre-condition: strWord only contains letters in the alphabet
*/
int     //function return type is in a separate line from the
funcA(char strWord[20] ,    //preferred to have 1 param per line
      int * pCount)            //use of prefix for variable identifiers
{  //open brace is at the beginning of the new line, aligned with the matching close brace
      int    ctr;        /* declaration of all variables before the start of any statements –
                         not inserted in the middle or in loop- to promote readability */

   *pCount = 0;
   for (ctr = 0; ctr < strlen(strWord); ctr++)  /*use of post increment, instead of pre-
                                                    increment */
   {  //open brace is at the new line, not at the end
      if (strWord[ctr] >= 'A' && strWord[ctr] <= 'Z')
      {    strWord[ctr] = strWord[ctr] + 32;
           (*pCount)++;
```

```
        }
        printf("%c", strWord[ctr]);
    }

    if (*pCount > 0)
        return 1;
    return 0;
}
```

Test Script should be in a table format. There should be at least 3 categories (as indicated in the description) of test cases **per function. There is no need to test functions which are only for screen design (i.e., no computations/processing; just printf).

Sample is shown below.

Function	#	Description	Sample Input Data	Expected Output	Actual Output	P/F
sortIncreasing	1	Integers in array are in increasing order already	aData contains: 1 3 7 8 10 15 32 33 37 53	aData contains: 1 3 7 8 10 15 32 33 37 53	aData contains: 1 3 7 8 10 15 32 33 37 53	P
	2	Integers in array are in decreasing order	aData contains: 53 37 33 32 15 10 8 7 3 1	aData contains: 1 3 7 8 10 15 32 33 37 53	aData contains: 1 3 7 8 10 15 32 33 37 53	P
	3	Integers in array are combination of positive and negative numbers and in no particular sequence	aData contains: 57 30 -4 6 -5 -33 -96 0 82 -1	aData contains: -96 -33 -5 -4 -1 0 6 30 57 82	aData contains: -96 -33 -5 -4 -1 0 6 30 57 82	P

Test descriptions are supposed to be unique and should indicate classes/groups of test cases on what is being tested. Given the sample code in page 6, the following are four distinct classes of tests:

i.) testing with strWord containing all capital letters (or rephrased as "testing for at least 1 modification")
ii.) testing with strWord containing all small letters (or rephrased as "testing for no modification")
iii.) testing with strWord containing a mix of capital and small letters
iv.) testing when strWord is empty (contains empty string only)

The following test descriptions are **incorrectly formed**:

- Too specific: testing with strWord containing "HeLlo"
- Too general: testing if function can generate correct count OR testing if function correctly updates the strWord
- Not necessary -- since already defined in pre-condition: testing with strWord containing special symbols and numeric characters

6. Upload the softcopies via Submit Assignment in Canvas. You can submit multiple times prior to the deadline. However, only the last submission will be checked. Send also to your **mylasalle account** a **copy** of all deliverables.
7. Use **<surnameFirstInit>.c** & **<surnameFirstInit >.pdf** as your filenames for the source code and test script, respectively. You are allowed to create your own modules (.h) if you wish, in which case just make sure that filenames are descriptive.
8. During the MP **demo**, the student is expected to appear on time, to answer questions, in relation with the output and to the implementation (source code), and/or to revise the program based on a given demo problem. Failure to meet these requirements could result to a grade of 0 for the project.
9. It should be noted that during the MP demo, it is expected that the program can be compiled successfully and will run. If the program does not run, the grade for the project is automatically 0. However, a running program with complete features may not necessarily get full credit, as implementation (i.e., code) and other submissions (e.g., non-violation of restrictions evident in code, test script, and internal documentation) will still be checked.

10. The MP should be an HONEST intellectual product of the student/s.

    For this project, you are allowed to do this individually or to be in a group of 2 members only. Should you decide to work in a group, the following mechanics apply:

    Individual Solution: Even if it is a group project, each student is still required to create his/her INITIAL solution to the MP individually without discussing it with any other students. This will help ensure that each student went through the process of reading, understanding, solving the problem and testing the solution.

    Group Solution: Once both students are done with their solution: they discuss and compare their respective solutions (ONLY within the group) -- note that learning with a peer is the objective here -- to see a possibly different or better way of solving a problem. They then come up with their group's final solution -- which may be the solution of one of the students, or an improvement over both solutions. Only the group's final solution, with internal documentation (part of comment) indicating whose code was used or was it an improved version of both solutions) will be submitted as part of the final deliverables. It is the group solution that will be checked/assessed/graded. Thus, only 1 final set of deliverables should be uploaded by one of the members in the Canvas submission page. [Prior to submission, make sure to indicate the members in the group by JOINing the same group number.]
    Individual Demo Problem: As each is expected to have solved the MP requirements individually prior to coming up with the final submission, both members should know all parts of the final code to allow each to INDIVIDUALLY complete the demo problem within a limited amount of time (to be announced nearer the demo schedule). This demo problem is given only on the day/time of the demo, and may be different per member of the group. Both students should be present/online during the demo, not just to present their individual demo problem solution, but also to answer questions pertaining to their group submission.

    Grading: the MP grade will be the same for both students -- UNLESS there is a compelling and glaring reason as to why one student should get a different grade from the other -- for example, one student cannot answer questions properly OR do not know where or how to modify the code to solve the demo problem (in which case deductions may be applied or a 0 grade may be given -- to be determined on a case-to-case basis).

11. Any form of **cheating** (**asking other people not in the same group for help**, **submitting as your [own group's] work part of other's work**, **sharing your [individual or group's] algorithm and/or code to other students not in the same group, etc.**) can be punishable by a grade of **0.0** for the **course** & a **discipline case**.

Any requirement not fully implemented, or instruction not followed will merit deductions.

----------There is only 1 deadline for this project: April 11 0800 AM, but the following are **suggested** targets.---------- *Note that each milestone assumes fully debugged and tested code/function, code written following coding convention and included internal documentation, documented tests in test script.

1.) Milestone 1 : February 23
    a. Design for the Project Implementation
    b. Collection of data needed for the project
    c. Menu options
    d. Preliminary outline of functions to be created

2.) Milestone 2: March 9
    a. User Registration
    b. Add a User Record
    c. Edit User Details
    d. Delete a User Record
    e. Display User entries
    f. Add Appointment entries
    g. Book Appointment
    h. View Appointment entries

3.) Milestone 3: March 23
    a. Manage Appointment details
    b. Check Appointment Status
    c. Ask FAQs using chatbot
    d. Add entries in Chatbot questions and answers

e. View Confirmed Appointments

[+]For these features, do not clear the contents of the array of entries upon exit from Data Management menu IF you have not implemented the import and export yet.

4.) Milestone 4: March 30
   a. Export
   b. Import

5.) Milestone 5: April 4
   a. Integrated testing (as a whole, not per function/feature)
   b. Collect and verify versions of code and documents that will be uploaded
   c. Recheck MP specs for requirements and upload final MP
   d. [Optional] Implement bonus features