TEST SCRIPT

Name: SANTOS, Miko L. **Course: CCPROG2**

Section: S20 **Ms. Jennifer Contreras**

Function	#	Descriptiton	Sample Input Data	Expected Output	Actual output	P/F
void **displayMain**(struct User userRegistrant[], struct Appointment userAppointment[], struct Chatbot FAQ[], int* i, int* j, int *k);	1	Test if the function displays the main menu and allows the user to select from the given options.	nChoice = 2	displayDataManMenu function should be called	displayDataManMenu function was called	P
	2	Test if the function correctly handles an invalid user input.	nChoice = 4	displayMain should display the main menu again and prompt the user to choose a valid option	displayMain displayed the main menu again and prompted the user to choose a valid option	P
	3	Test if the function exits properly when the user chooses to exit.	nChoice = 3	exitPrompt function should be called and the program should exit	exitPrompt function was called and the program exited	P
void **displayVaccRegMenu**(struct User userRegistrant[], struct Appointment userAppointment[], int* i, int* j);	1	Testing if the function properly displays the vaccine registration menu and if the user can select the option to **register as a user.**	userRegistrant = [], userAppointment = [], i = 0, j = 0 Input: 1	-Display the header and the vaccine registration menu options. -Prompt the user to choose a command and input 1 to register as a user. -Direct to register_ user() function.	-Displayed the header and the vaccine registration menu options. -Prompted the user to choose a command and input 1 to register as a user. - Directed to register_user() function.	P

	2	Testing if the function properly displays the vaccine registration menu and if the user can select the option to set a **vaccination appointment**.	userRegistrant = [], userAppointment = [], i = 0, j = 0 Input: 2	-Display the header and the vaccine registration menu options. -Prompt the user to choose a command and input 2 to register as a user. -Direct to displayVaccAppointment() function.	-Displayed the header and the vaccine registration menu options. -Prompted the user to choose a command and input 2 to register as a user. - Directed to displayVaccAppointment() function.	P
	3	Testing if the function properly displays the vaccine registration menu and if the user can select the option to view the **chatbot FAQ.**	Sample Input Data: userRegistrant = [], userAppointment = [], i = 0, j = 0 Input: 3	-Display the header and the vaccine registration menu options. -Prompt the user to choose a command and input 3 to register as a user. -Direct to chatbotFAQ() function.	-Displayed the header and the vaccine registration menu options. -Prompted the user to choose a command and input 3 to register as a user. - Directed to chatbotFAQ() function.	
void **register_user**(struct User userRegistrant[], int* i);	1	Test for valid input	userID = 1234 password = Test1234 fullName = John Doe address = 123 Main St contact = 1234567890	No errors, user data saved in userRegistrant array	No errors, user data saved in userRegistrant array	P
	2	Test for invalid userID	userID = 123 password = Test1234 fullName = John Doe address = 123 Main St contact = 1234567890	Error message "User ID must be 4 digit numbers" displayed, user data not saved	Error message "User ID must be 4 digit numbers" displayed, user data not saved	P

	3	Test for existing	userIDuserID = 1001 (assuming userID 1001 already exists in userRegistrant array) password = Test1234 fullName = John Doe address = 123 Main St contact = 1234567890	Error message "This userID already exists. Please type another." displayed, user data not saved	Error message "This userID already exists. Please type another." displayed, user data not saved	P
void **displayVaccAppointment**(struct User userRegistrant[], struct Appointment appointmentRequest[], int* i, int* j);	1	Test if the function displays the appointment menu correctly when option 1 is chosen.	nChoice = 1, userRegistrant[], userAppointment[], i=0, j=0	Display the appointment menu with option 1 selected and call the apptReq function.	Displayed the appointment menu with option 1 selected and called the apptReq function.	P
	2	Test if the function displays the appointment menu correctly when option 2 is chosen.	nChoice = 2, userRegistrant[], userAppointment[], i=0, j=0	Display the appointment menu with option 2 selected and call the manageAppointment function.	Displayed the appointment menu with option 2 selected and called the manageAppointment function.	P
	3	Test if the function exits the menu loop correctly when option 3 is chosen.	nChoice = 3, userRegistrant[], userAppointment[], i=0, j=0	Exit the menu loop and return to the calling function.	Exited the menu loop and returned to the calling function. Pass/Fail: Pass	P
int **loginVaccAppointment**(struct User userRegistrant[], int* i);	1	Test with a valid user ID and a valid password	userID = 1234, password = "password123"	loginStatus = 1 (successful login)	LoginStatus = 1	P
	2	Test with an invalid user ID and a valid password	userID = 999, password = "password123"	loginStatus = 0 (unsuccessful login)	loginStatus = 0	P

	3	Test with a valid user ID and an invalid password	userID = 5678, password = "12345678901"	loginStatus = 0 (unsuccessful login)	loginStatus = 0	P
int **CheckUserCredentials**(struct User userRegistrant[], int userID, char password[], int *i);	1	Test with valid userID and password that matches a registered user in the system	userRegistrant[] = [{userID: 1234, password: "password1"}, {userID: 5678, password: "password2"}] userID = 1234 password = "password1" i = 1	Output message "Status: Login successful." Return value 1	Status: Login successful. Return value 1	P
	2	Test with valid userID and incorrect password	userRegistrant[] = [{userID: 1234, password: "password1"}, {userID: 5678, password: "password2"}] userID = 1234 password = "incorrectpassword" i = 1	Output message "Status: Incorrect password. Try again." Return value 0	Status: Incorrect password. Try again. Return value 0	P
	3	Test with invalid userID that does not exist in the system	userRegistrant[] = [{userID: 1234, password: "password1"}, {userID: 5678, password: "password2"}] userID = 9999 password = "password3" i = 1	Output message "Status: Account does not exist. Try again." Return value 0	Status: Account does not exist. Try again. Return value 0	P
void **apptReq**(struct User userRegistrant[], struct Appointment userAppointment[], int *i, int* j);	1	Test for a valid appointment sign-up, all inputs are correct and within the valid range.	User Login: [{userID: 1234, password: testpass] Appointment List: [{appID: 1000, customerName: Matilda '', string: ['Manila', 'AstraZeneca', '2023-05-01'], time: '10:00'}]	No error messages. UserAppointment[0] should contain the following: appID: 1000 customerName:”Matil	No error messages. UserAppointment[0] contains the following: appID: 1000 customerName:”Matilda' string: ['Manila',	P

			i = 1 j = 0	da' string: ['Manila', 'AstraZeneca', '2023-05-01'] time: '10:00']	'AstraZeneca', '2023-05-01'] time: '10:00']	
	2	Test for an invalid appointment ID, which is not a 4-digit number.	User Login: [{userID: 1234, password: testpass] Appointment List: [{appID: 12345, customerName:Matilda ", string: ['Manila', 'AstraZeneca', '2023-05-01'], time: '10:00'}] i = 1 j = 0	Error message: "Appointment ID must be 4 digit numbers".	Error message: "Appointment ID must be 4 digit numbers".	P
	3	Test for an invalid appointment ID, which is already taken	User Login: [{userID: 12345, password: testpass}] Appointment List: [{appID: 1000, customerName: Matilda", string: ['Manila', 'AstraZeneca', '2023-05-01'], time: '10:00'}] i = 1 j = 1	Error message: "This userID already exists. Please type another.".	Error message: "This userID already exists. Please type another."	P
void **manageAppointment**(struct User userRegistrant[], struct Appointment userAppointment[], int* i, int* j);	1	Testing if the function properly displays the Manage Appointment Manu and if the user can	nChoice = 1	-Display the header and the appointment menu options. -Prompt the user to choose a command and input 1 to register	-Display the header and the appointment menu options. -Prompt the user to choose a command and input 1 to register as a user. -Direct to cancelAppt()	P

		select the option to view the cancelAppt()		as a user. -Direct to cancelAppt() function.	function.	
	2	Testing if the function properly displays the Manage Appointment Manu and if the user can select the option to view the reschedDT()	nChoice = 2	-Displayed the header and the appointment menu options. -Prompted the user to choose a command and input 1 to register as a user. -Directed to reschedDT() function.	-Displayed the header and the appointment menu options. -Prompted the user to choose a command and input 1 to register as a user. -Directed to reschedDT() function.	P
	3	Testing if the function properly displays the Manage Appointment Manu and if the user can select the option to view the changeVLVT()	nChoice = 3	-Display the header and the appointment menu options. -Prompt the user to choose a command and input 1 to register as a user. -Direct to changeVLVT() function.	-Display the header and the appointment menu options. -Prompt the user to choose a command and input 1 to register as a user. -Direct to changeVLVT() function.	P
void **cancelAppt**(struct User userRegistrant[], struct Appointment userAppointment[], int* i, int* j);	1	Test canceling an appointment that exists in the list of appointments for the user.	userRegistrant: an array of struct User objects where userRegistrant[0].userID=1234 anduserRegistrant[0].password="password" userAppointment : an array of struct Appointment objects where userAppointment[0]. appID=1000, userAppointment[0]. customerName="Juan Dela Cruz", userAppointment[0].	The appointment with ID 1000 should be deleted from the userAppointment array and the output should say that the appointment has been canceled. Status: Appointment	Status: Appointment ID 1000 has been cancelled	P

			time="10:00", userAppointment[0].string[0]="Quezon City", userAppointment[0].string[1]="Manila", userAppointment[0].string[2]="Taguig", and userAppointment[0].string[3]="Makati" i: a pointer to an integer with value 0 j: a pointer to an integer with value 0	ID has been cancelled		
	2	Test canceling an appointment that does not exist in the list of appointments for the user.	userRegistrant: an array of struct User objects where userRegistrant[0].userID=1234 and userRegistrant[0].password="password" userAppointment: an array of struct Appointment objects where userAppointment[0]. appID=1000, userAppointment[0].customerName="Juan Dela Cruz", userAppointment[0]. time="10:00", userAppointment[0]. string[0]="Quezon City", userAppointment[0]. string[1]="Manila", userAppointment[0]. string[2]="Taguig", and userAppointment[0]. string[3]="Makati" i: a pointer to an integer with value 0 j: a pointer to an integer with value 0	Status: Appointment ID does not exist.	Status: Appointment ID does not exist.	P
	3	Test canceling an appointment with an invalid password.	userRegistrant: an array of struct User objects where userRegistrant[0].userID=1234 and userRegistrant[0].password=	Status: Incorrrect Credentials. Try again	Status: Incorrrect Credentials. Try again	P

			"password" userAppointment: an array of struct Appointment objects where userAppointment[0].appID=1, userAppointment[0].customerName="Juan Dela Cruz", userAppointment[0].time= "10:00", userAppointment[0].string[0]="Quezon City", userAppointment[0].string[1]="Manila", userAppointment[0].string[2]="Taguig", and userAppointment[0].string[3]="Makati" i: a pointer to an integer with value 0 j: a pointer to an integer with value 0			
void **reschedDT**(struct User userRegistrant[], struct Appointment userAppointment[], int* i, int* j);	1	Test with valid user ID, password, and appointment ID. The user wants to reschedule the appointment and enters the new date and time in the correct format.	User ID: 1234 Password: abcd1234 Appointment ID: 5678 New Date: 2023-05-01 New Time: 13:00	Message: Login successful. Display the appointment details with the updated date and time.	Message: Login successful. Display the appointment details with the updated date and time.	P
	2	Test with valid user ID, password, and appointment ID. The user wants to reschedule the appointment and enters an invalid date format.	User ID: 2345 Password: qwer5678 Appointment ID: 6789 New Date: 20230501 New Time: 12:30	Message: Login successful. Message: (Error) *indicating that the entered date format is invalid.*	Message: Login successful. Message: (Error)	P

	3	Test with invalid user ID and password. The user cannot log in. Sample Input Data:	User ID: 1001 Password: asdfghjkl	Status: Incorrrect Credentials. Try again.	Status: Incorrrect Credentials. Try again.	P
void **swapStrings**(char *str1, char *str2);	1	Test swapping two strings "old appointment date and time" and "new appointment date and time"	char old_date_time[] = "April 10, 2023, 10:00 AM"; char new_date_time[] = "April 11, 2023, 11:00 AM";	old_date_time = "April 11, 2023, 11:00 AM"; new_date_time = "April 10, 2023, 10:00 AM";	old_date_time = "April 11, 2023, 11:00 AM"; new_date_time = "April 10, 2023, 10:00 AM";	P
	2	Test swapping two strings "old location" and "new location"	char old_location[] = "Quezon City"; char new_location[] = "Makati City";	old_location = "Makati City"; new_location = "Quezon City";	old_location = "Makati City"; new_location = "Quezon City";	P
	3	Test swapping two strings "old vaccination dose type" and "new vaccination dose type"	char old_vaccdose[] = "AstraZeneca"; char new_vaccdose[]="Pfizer ";	char old_vaccdose[] = "Pfizer"; char new_vaccdose[]= "AstraZeneca ";	char old_vaccdose[] = "Pfizer"; char new_vaccdose[] = "AstraZeneca ";	
void **changeVLVT**(struct User userRegistrant[], struct Appointment userAppointment[], int* i, int* j);	1	User enters valid user ID and password, and a valid appointment ID, and inputs a new location and	User ID: 1001 Password: pass123 Appointment ID: 1 New Location: Manila New Vaccine Brand: Pfizer	The program displays the updated schedule with the new location and new vaccine brand.	The program displays the updated schedule with the new location and new vaccine brand.	P

		vaccine brand				
	2	User enters the correct login details and an incorrect appointment ID.	User ID: 1001 Password: pass123 Appointment ID: 100	Message: Login successful. Message: (Error) The program should display an error message.	Message: Login successful. Message: (Error) The program should display an error message.	P
	3	User enters the wrong password three times.	User ID: 1001 Password: wrongpass (entered 3 times)	Status: Try again later	Status: Try again later	P
int **apptIDCheckingDuplication**(struct Appointment userAppointment[], int newAppID, int* j);	1	Test if the function can correctly identify a duplicated appointment ID in the given array of appointments.	Appointment array: [{appID: 1234, name: "Juan", date: "2023-05-01"}, {appID: 4567, name: "Maria", date: "2023-05-02"}, {appID: 7890, name: "Pedro", date: "2023-05-03"}] New appointment ID: 4567	The appointment ID already exists. Please try again."	The appointment ID already exists. Please try again."	P
	2	Test if the function can correctly identify a new appointment ID in the given array of appointments.	Appointment array: [{appID: 1234, name: "Juan", date: "2023-05-01"}, {appID: 4567, name: "Maria", date: "2023-05-02"}, {appID: 7890, name: "Pedro", date: "2023-05-03"}] New appointment ID: 9999	0 (No duplicated appointment ID)	0 (No duplicated appointment ID)	P
	3	Test if the function can correctly identify a duplicated appointment ID in the given array of	Appointment array: [{appID: 1234, name: "Juan", date: "2023-05-01"}, {appID: 456, name: "Maria", date: "2023-05-02"}, {appID: 7890, name: "Pedro", date:	Message: (Error) 1	Message: (Error) 1	

		appointments	"2023-05-03"}] New appointment ID: 7890			
int **userIDCheckingDuplication**(struct User userRegistrant[], int newUserID, int* i);	1	Test to check if the function detects a duplicate user ID.	userRegistrant = [{userID: 1234, name: "John Doe", address: "Manila"}, {userID: 4567, name: "Jane Doe", address: "Cebu"}], newUserID = 1234, i = 2	"The user ID already exists. Please try again."	"The user ID already exists. Please try again."	P
	2	Test to check if the function returns 0 for a new user ID.	userRegistrant = [{userID: 123, name: "John Doe", address: "Manila"}, {userID: 456, name: "Jane Doe", address: "Cebu"}], newUserID = 7890, i = 2	Returns 0	Returns 0	P
	3	Test for a new user ID that does not already exist in the user registrant array	User registrant array: [{userID: 1234, name: "John Doe", address: "Manila"}, {userID: 456, name: "Jane Smith", address: Baguio: }] New user ID: 7890 i: 2	Returns 0	Returns 0	P
void **chatbotFAQ**()	1	Test the function when the user asks whether vaccination is mandatory	"Is vaccination mandatory?"	Prints “No, but highly recommended by the government.” Prompts “Do you have any more questions?Type your response:”	Printed “No, but highly recommended by the governmen”. Prompted: “Do you have any more questions?Type your response:”	P
	2	Test the function	"What are the approved COVID-19	Prints “Approved	Printed “Approved	P

		when the user asks about the available COVID-19 vaccines	vaccines?"	COVID-19 vaccines, including Pfizer-BionTech, AstraZeneca, and Sinovac". Prompts "Do you have any more questions?Type your response:"	COVID-19 vaccines, including Pfizer-BionTech, AstraZeneca, and Sinovac". Prompted "Do you have any more questions?Type your response:"	
	3	Test the function when the user asks about the side effects of the vaccine	"What are the possible side effects of the COVID-19 vaccine?"	Prints "Possible side effects: pain, fever, fatigue, headache, and muscle aches.". Prompts ""Do you have any more questions? Type your response:"	Prints "Possible side effects: pain, fever, fatigue, headache, and muscle aches.". Prompts ""Do you have any more questions? Type your response:"	P
int **loginDataMananagement**(struct User userRegistrant[], int *i);	1	User enters valid user ID and correct password on the first attempt.	User ID: 5678 Password: 4321	Display header -Prompt for user ID and password -Display Display "Status: Login successful."	Display header -Prompt for user ID and password -Display Display "Status: Login successful."	P
	2	User enters valid user ID and incorrect password on the first attempt.	User ID: 5678 Password: 4321	-Display header -Prompt for user ID and password -Display "Status: Incorrect password.	-Display header -Prompt for user ID and password -Display "Status: Incorrect password. Try again."	P

				Try again."		
	3	User enters invalid user ID on the first attempt.	User ID: 9999 Password: 1234	Expected Output: -Display header -Prompt for user ID and password -Display "Status: Account does not exist. Try again.	-Display header -Prompt for user ID and password -Display "Status: Account does not exist. Try again."	P
void **displayDataManMenu**(struct User userRegistrant[], struct Appointment userAppointment[], struct Chatbot FAQ[], int* i, int* j, int* k);	1	Test the function with the user selecting option 1 (userManagement Menu).	nChoice = 1	The userManagementMenu function() should be called with userRegistrant() and i as input parameters.	The userManagementMenu function() should be called with userRegistrant() and i as input parameters.	P
	2	Test the function with the user selecting option 5	nChoice = 5	The importMenu() function should be called with userRegistrant, userAppointment, FAQ, i, j, and k as input parameters.	The importMenu() function should be called with userRegistrant, userAppointment, FAQ, i, j, and k as input parameters.	P
	3	Test the function with the user selecting option 6 (Exit the program).	nChoice = 6	The repeat variable should be set to 0 to exit the while loop.	The repeat variable should be set to 0 to exit the while loop.	P
void **DataManMenu**(struct User userRegistrant[], struct Appointment	1	Testing userManagement Menu() option	nChoice :1	Direct to userManagement Menu() function	Directed to userManagement Menu() function	P

userAppointment[], struct Chatbot FAQ[], int *i, int* j, int* k);						
	2	Testing export data menu option Sample Input Data:	nChoice :4	Direct to ExportMenu() function	Directed to ExportMenu() function	P
	3	Testing import data menu option	nChoice :5	Direct to importMenu() function	Directed to importMenu() function	P
void **userManagementMenu**(struct User userRegistrant[], int* i);	1	Test adding a new user with valid input	nChoices = 1	Direct to addNewUser() function	Directed to addNewUser() function	P
	2	Test viewing all user with valid input	nChoice = 2	Direct to viewAllUser() function	Directed to viewAllUser() function	P
	3	Test editing new user's data	nChoice = 3	Direct to editNewUser() function	Directed to editNewUser() function	P
void **deleteNewUser**(struct User userRegistrant[], int *i);	1	Test deleting a single appointment when only one appointment exists	userAppointment = [{id=1234, name="Juan", date="2022-05-01", time="10:00"}], j=1	The appointment at index 0 should be deleted and j should be decremented by 1. The new value of userAppointment should be [{id=0, name="", date="", time=""}] and j should be 0.	The appointment at index 0 was deleted and j was decremented by 1. The new value of userAppointment is [{id=0, name="", date="", time=""}] and j is 0.	P

	2	Test the function with a valid appointment number that exists in the array of appointments	userAppointment = {{"Appointment 1", "2023-04-09", "10:30 AM", "Juan Dela Cruz", "09123456789"}, {"Appointment 2", "2023-04-10", "11:00 AM", "Maria Clara", "09187654321"}}, j = 2, appNum = 1	The appointment with index 1 should be deleted, and the appointment with index 2 should be shifted to index 1, and j should be decremented by 1. A success message should be displayed. “Appointment No. %d Is Deleted Successfully!”	The appointment with index 1 should be deleted, and the appointment with index 2 should be shifted to index 1, and j should be decremented by 1. A success message should be displayed. “Appointment No. %d Is Deleted Successfully.”	P
	3	Test deleting an appointment with an invalid appointment number	userAppointment = [{id=1234, name="Juan", date="2022-05-01", time="10:00"}, {id=2, name="Maria", date="2022-05-02", time="11:00"}], j=2, appNum=3	Message (Error) The value of userAppointment and j remained unchanged.	Message (Error) The value of userAppointment and j remained unchanged.	P
int **compareUsers**(const void *a, const void *b);	1	Test function with two users with different userIDs.	struct User user1 = {"John", 123}; struct User user2 = {"Jane", 456};	If user1.userID < user2.userID, return a negative value. Otherwise, return a positive value. In this case, the expected output is a negative value.	int result = compareUsers(&user1, &user2); printf("Result: %d\n", result);	P
	2	Test function with two users with the same userID.	User A: userID = 1001 userName = "John" User B: userID = 1001 userName = "Jane"	Return 0 since user1.userID is equal to user2.userID.	Return 0	P

	3	Test the function with two users having different user IDs	User A: userID = 1001 userName = "John" User B: userID = 1002 userName = "Jane"	Return -1	Return -1	P
void **addNewUser**(struct User userRegistrant[], int *i);	1	Test the function when the user enters a userID that is less than 1000.	userID = 999	The system should display an error message (Error) that the userID must be 4 digit numbers.	The system should display an error message (Error) that the userID must be 4 digit numbers.	P
	2	Test the function when the user enters a userID that already exists.	userRegistrant[0].userID = 1234 userID = 1234	The system should display an error message that the userID already exists.	The system should display an error message that the userID already exists.	P
	3	Test the function when the user enters valid inputs.	userID = 5678 password = Test1234 fullName = Juan dela Cruz address = Manila contact = 09123456789 sex = Male	The system should add the new user to the system.	The system should add the new user to the system.	P
char* **get_first_wor**d(char* input, int width) ;	1	Testing the function with a Filipino name "Maria Dela Cruz" and a width of 5.	""Maria Dela Cruz", 5	"Maria"	"Maria"	P
	2	Testing the function with a Filipino name	"Juanito Santos", 10	"Juanito"	"Juanito"	P

		"Juanito Santos" and a width of 10.				
	3	Testing the function with a Filipino name "Andres Bonifacio" and a width of 7.	"Andres Bonifacio", 7	"Andres"	"Andres"	P
void **editNewUser**(struct User userRegistrant[]);	1	Testing if the function is able to edit user data when provided with valid user number, user ID, password, full name, address, contact number, and sex.	User Number: 1 User ID: 1234 Password: password1 Full Name: Juan Dela Cruz Address: 1234 Sampaguita Street, Makati City Contact Number: 09123456789 Sex: Male	The function should successfully edit the user data for user number 1 with the above input data.	The function successfully edited the user data for user number 1 with the input data.	P
	2	Testing if the function is able to detect invalid input when provided with an invalid contact number.	User Number: 2 User ID: 2345 Password: password2 Full Name: Maria Garcia Address: 5678 Narra Street, Quezon City Contact Number: 0912345 Sex: Female	The function should detect an invalid contact number and prompt the user to enter a valid one.	The function detected an invalid contact number and prompted the user to enter a valid one.	P
	3	Testing if the function is able to detect invalid input when provided with an invalid password.	User Number: 3 User ID: 3456 Password: qwertyuiopasdfghjklzxcvbnm Full Name: Mark Santos Address: 2468 Dahlia Street, Pasay City Contact Number: 09123456789 Sex: Male	The function should detect an invalid password and prompt the user to enter a valid one.	The function detected an invalid password and prompted the user to enter a valid one.	P
void	1	Adding a new	nChoices = 1	Direct to	Directed to	P

appointmentManagemant Menu(struct Appointment userAppointment[], int *j);		appointment		addNewAppointment();	addNewAppointment();	
	2	Viewing all appointments	nChoices = 2	Direct to viewAllAppointment();	Directed to viewAllAppointment();	P
	3	Deleting an appointment	nChoices = 3	Direct to deleteNewAppointment();	Directed to deleteNewAppointment();	P
void **addNewAppointment** (struct Appointment userAppointment[], int *j);	1	Test adding new appointment with valid inputs	appID = 1001 customerName = "Juan Dela Cruz" time = "10:00" vaccinationLocation = "Manila" vaccineType = "Moderna" appointmentDate = "2023-04-11" vaccineDoseDate = "2023-04-20" Response = "no"	Appointment added successfully with the following details: Appointment ID: 1001 Customer Name: Juan Dela Cruz Appointment Time: 10:00 Vaccination Location: Manila Vaccine Type: Moderna Appointment Date: 2023-04-11 Vaccine Dose Date: 2023-04-20	Appointment added successfully with the following details: Appointment ID: 1001 Customer Name: Juan Dela Cruz Appointment Time: 10:00 Vaccination Location: Manila Vaccine Type: Moderna Appointment Date: 2023-04-11 Vaccine Dose Date: 2023-04-20	P
	2	Test adding new appointment with invalid customer name	appID = 1002 customerName = "" time = "10:00" vaccinationLocation = "Manila" vaccineType = "Sinovac" appointmentDate = "2023-04-11"	(Error)	(Error)	P

			vaccineDoseDate = "2023-04-20" Response = "no"			
	3	Test adding new appointment with invalid vaccine dose date	appID = 1003 customerName = "Maria Cruz" time = "10:00" vaccinationLocation = "Makati" vaccineType = "Pfizer" appointmentDate = "2023-04-11" vaccineDoseDate = "2023-04-200" Response = "no"	(Error)	(Error)	P
void **editNewAppointment**(struct Appointment userAppointment[], int *j);	1	Editing an appointment that already exists in the array	Appointment Number: 2 App ID: 1000 Full Name: Juan Dela Cruz Time: 10:00 AM Vaccination Location: Philippines General Hospital First Dose Vaccine Type: Pfizer Second Dose Vaccine Type: -	Appointment successfully modified.	Appointment successfully modified.	P
	2	Testing with appointment number 1, with invalid customer name input.	Appointment number: 1 Customer name: "" Time: "02:00 PM" Vaccination Location: "Quezon City Health Center" First Dose Vaccine Type: "Sinovac" Second Dose Vaccine Type: "Sinovac"	(Error) Retry customer name input	(Error) Retry customer name input	P
	3	Testing with appointment number 2, with	Appointment number: 2 Customer name: "Maria Dela Cruz" Time: "10:00 AM"	(Error) Retry second dose	(Error) Retry second dose vaccine	P

		invalid vaccine type input.	Vaccination Location: "Mandaluyong Health Center" First Dose Vaccine Type: "AstraZeneca" Second Dose Vaccine Type: ""	vaccine type input	type input	

void **deleteNewAppointment**(struct Appointment userAppointment[], int *j);	1	Test deleting the only appointment	userAppointment = { {1, "John Doe", "10:00", "checkup"},
			{0, "", "", ""}, {0, "", "", ""},
			{0, "", "", ""}, {0, "", "", ""} };
			
			j = 1;
			appNum = 1;
				Prints ”Appointment No. %d Is Deleted Successfully!”	Prints ”Appointment No. %d Is Deleted Successfully!”	P

	2	Attempt to delete an appointment with an appointment number greater than the number of appointments in the list.	userAppointment =
			{ {1, "John Doe", "10:00", "checkup"},
			{2, "Jane Smith", "11:00", "follow-up"},
			{3, "Juan Dela Cruz", "14:00", "consultation"},
			{0, "", "", ""}, {0, "", "", ""} };
			
			j = 3;
			appNum = 5;
				Invalid appointment number. Please enter a number between 1 and 3, inclusive.	Invalid appointment number. Please enter a number between 1 and 3, inclusive.	P

	3	Delete an appointment from a list of appointments with multiple appointments.	userAppointment =
			{ {1, "John Doe", "10:00", "checkup"},
			{2, "Jane Smith", "11:00", "follow-up"},
			{3, "Juan Dela Cruz", "14:00", "consultation"},
			{0, "", "", ""},
			{0, "", "", ""} };
			
			j = 3;
			appNum = 2;
				Prints:
				“Appointment No. 2 Is Deleted Successfully!
				Do you want to delete another question and answer? (Y/N):”
				
				Then prompts the user to either choose to continue deleting another appointment or not.
					Prints:
					“Appointment No. 2 Is Deleted Successfully!
					Do you want to delete another question and answer? (Y/N):”
					
					Then prompts the user to either choose to continue deleting another appointment or not.
						P

void **chatbotManagemanetMenu**(struct Chatbot FAQ[], int *k)	1	Testing if the function directs to addNewFAQ subfunction when nChoices is 1	nChoices = 1	The function should direct to addNewFAQ() function and allow the user to add a new question and answer to the FAQ database. After completion, the user should be directed back to the main menu.	The function should direct to addNewFAQ() function and allow the user to add a new question and answer to the FAQ database. After completion, the user should be directed back to the main menu.	P
	2	Testing if the function directs to viewAllFAQ subfunction when nChoices is 2	nChoices = 2	The function should direct to viewAllFAQ() function and display all the questions and answers in the FAQ database. After completion, the user should be directed back to the main menu.	The function should direct to viewAllFAQ() function and display all the questions and answers in the FAQ database. After completion, the user should be directed back to the main menu.	P
	3	Testing if the function directs to editNewFAQ subfunction when nChoices is 3	nChoices = 3	The function should direct to editNewFAQ() function and allow the user to edit an existing question and answer in the FAQ database. After completion, the user should be directed	The function should direct to editNewFAQ() function and allow the user to edit an existing question and answer in the FAQ database. After completion, the user should be directed back to the main menu.	P

				back to the main menu.		
void **addNewFAQ**(struct Chatbot FAQ[], int *k);	1	Test adding a single FAQ to the Chatbot	Question: "What is COVID-19?" Answer: "COVID-19 is a highly contagious respiratory illness caused by the SARS-CoV-2 virus." Choice: N	FAQ[0].questions = "What is COVID-19?\n" FAQ[0].answers = "COVID-19 is a highly contagious respiratory illness caused by the SARS-CoV-2 virus.\n"	FAQ[0].questions = "What is COVID-19?\n" FAQ[0].answers = "COVID-19 is a highly contagious respiratory illness caused by the SARS-CoV-2 virus.\n"	
	2	Add two new FAQs related to the COVID-19	Question 1: "What are the possible side effects of the COVID-19 vaccine in the Philippines?" Answer 1: "The common side effects of the COVID-19 vaccine in the Philippines include pain or swelling at the injection site, fever, headache, fatigue, and muscle aches." Question 2: "Can pregnant women receive the COVID-19 vaccine in the Philippines?" Answer 2: "Yes, pregnant women in the Philippines can receive the COVID-19 vaccine. Choice: 'n'	The FAQ array should have two new entries with the provided questions and answers.	The FAQ array should have two new entries with the provided questions and answers.	P
	3	Add three new FAQs	Question 1: "What are the quarantine classifications in the Philippines?" Answer 1: "The quarantine	The FAQ array should have three new entries with the provided questions	The FAQ array should have three new entries with the provided questions and answers.	P

			classifications in the Philippines include ECQ,MECQ, GCQ, and MGCQ. Question 2: "Are gatherings allowed under pandemic restrictions?" Answer 2: "A certain number of people are allowed depending on the quarantine classification. Question 3: "What are the travel restrictions?"	and answers.		
void **editNewFAQ** (struct Chatbot FAQ[], int *k)	1	Test the function with valid index of question and answer to edit.	FAQ[0].questions = "What are the common symptoms of COVID-19?"; FAQ[0].answers = "The common symptoms of COVID-19 are fever, dry cough, and tiredness."; index = 0; choice = 'n';	The function should edit the question and answer for the given index and print the edited question and answer	The function should edit the question and answer for the given index and print the edited question and answer	P
	2	Test the function with an invalid index of question and answer to edit.	FAQ[0].questions = "What are the common symptoms of COVID-19?"; FAQ[0].answers = "The common symptoms of COVID-19 are fever, dry cough, and tiredness."; index = 1; choice = 'n';	The function should not edit any question and answer and should exit the loop when the user chooses to stop editing.	Enter the index of the question you want to edit: 1 Do you want to edit another question and answer? (Y/N): n	P
	3	Test the function with multiple edits and choices.	FAQ[0].questions = "What are the common symptoms of COVID-19?"; FAQ[0].answers = "The common symptoms of COVID-19 are fever, dry cough, and tiredness.";	The function should edit the question and answer for the given index and print the edited question and answer.	The function should edit the question and answer for the given index and print the edited question and answer.	P

			FAQ[1].questions = "How does COVID-19 spread?"; FAQ[1].answers = "COVID-19 spreads through respiratory droplets."; index = 0; choice = 'y';			
void **deleteNewFAQ** (struct Chatbot FAQ[], int *k);	1	Testing if the function can delete a question and answer from the chatbot FAQ array with only one element	FAQ[0].questions = "What are the common symptoms of COVID-19?"; FAQ[0].answers = "The common symptoms of COVID-19 are fever, dry cough, and tiredness."; *k = 1; Enter index to delete: 0	The console should display "Question and answer deleted." and return from the function.	The console should display "Question and answer deleted." and return from the function. FAQ[0].questions = "" FAQ[0].answers = “”	P
	2	Testing if the function can delete a question and answer from the chatbot FAQ array with multiple elements.	FAQ[0].questions = “What are the common symptoms of COVID-19?"; FAQ[0].answers = "The common symptoms of COVID-19 are fever, dry cough, and tiredness."; FAQ[1].questions = ""How does COVID-19 spread?" FAQ[1].answers = "COVID-19 spreads through respiratory droplets."; *k = 1; Enter index to delete: 1	The console should display "Question and answer deleted." and shift the remaining elements to the left.	The console should display "Question and answer deleted." and shift the remaining elements to the left. FAQ[0].questions = ""How does COVID-19 spread?" FAQ[0].answers = "COVID-19 spreads through respiratory droplets.";	p
	3	Testing if the	FAQ[0].questions ="What are the	The console should	User enters "3" as the input	P

		function can delete multiple questions and answers from the chatbot FAQ array.	common symptoms of COVID-19?" FAQ[0].answers = - "The common symptoms of COVID-19 are fever, dry cough, and tiredness." FAQ[1].questions ="How does COVID-19 spread?" FAQ[1].answers = "COVID-19 spreads from an infected person's mouth or nose when they cough, sneeze, or talk. The virus can also spread from surfaces." FAQ[2].questions ="How long does it take for COVID-19 symptoms to appear?" FAQ[2].answers = "It takes around 5-6 days on average, but it can take up to 14 days for COVID-19 symptoms to appear." k = 2 Index = 1;	prompt the user to enter the question number they want to delete and wait for input. The user should enter "3" as the input. The console should display a message saying "Question and answer deleted." and prompt the user if they want to delete another question and answer. The user should enter "n" as the input. The function should delete the last element in the FAQ array and decrement the size counter k to 2.	when prompted to enter the question number they want to delete. The console displays a message saying "Question and answer deleted." and prompts the user if they want to delete another question and answer. User enters "n" as the input. The function deletes the last element in the FAQ array and decrements the size counter k to 2. FAQ[0].questions ="How does COVID-19 spread?" FAQ[0].answers = "COVID-19 spreads from an infected person's mouth or nose when they cough, sneeze, or talk. The virus can also spread from surfaces." FAQ[1].questions ="How long does it take for COVID-19 symptoms to appear?" FAQ[1].answers = "It takes around 5-6 days on average, but it can take up to 14 days for COVID-19	

					symptoms to appear."	
void **ExportMenu**(struct User userRegistrant[], struct Appointment userAppointment[], struct Chatbot FAQ[], int* i, int* j,int* k);	1	Export UserRegistrant's data to "Users.txt"	filename: Users userRegistrant[].fullName = "Maria Clara" userRegistrant[].userD = 1111 userRegistrant[].contact = 09123456789 userRegistrant[].address = Manila userRegistrant[].password = asdf userRegistrant[].sex = F userRegistrant[].string[0] = 2020-10-10 userRegistrant[].string[1] = Moderna userRegistrant[].string[2] = Manila userRegistrant[].string[3] = 2021-01-10 userRegistrant[].string[4] = Moderna userRegistrant[].string[5] = Manila userRegistrant[].string[6] = 2021-04-10 userRegistrant[].string[7] = Moderna userRegistrant[].string[8] = Manila	1111 asdf Maria Manila 09123456789 F Manila 2020-10-10 Moderna Mani;a 2021-01-10 Moderna Manila 2021-04-10 Moderna	1111 asdf Maria Manila 09123456789 F Manila 2020-10-10 Moderna Mani;a 2021-01-10 Moderna Manila 2021-04-10 Moderna	P
	2		userAppoinment[].appId = 1212 userAppoinment[].password = asdf userAppoinment[].customerName = "Maria Clara" userAppoinment[].time = 12:00 userAppoinment[].string[0] = Manila userAppoinment[].string[1] = Pfizer userAppoinment[].string[2] = 2023-04-10	1222 1111 Maria Manila Pfizer 2023-04-10 12:00 2020-10-10	1222 1111 Maria Manila Pfizer 2023-04-10 12:00 2020-10-10	P

			userAppoinment[].string[3] = 2020-10-10			
	3		FAQ[].question = “What is Covid-19?” FAQ[].answer = “infectious disease caused by the SARS-CoV-2 virus”	What is Covid-19? infectious disease caused by the SARS-CoV-2 virus	What is Covid-19? infectious disease caused by the SARS-CoV-2 virus	P
void **ImportMenu**(struct User userRegistrant[], struct Appointment userAppointment[], struct Chatbot FAQ[], int* i, int* j,int* k);	1	For Users.txt	1111 asdf Maria Manila 09123456789 F Manila 2020-10-10 Moderna Mani;a 2021-01-10 Moderna Manila 2021-04-10 Moderna	userRegistrant[].fullName = “Maria Clara” userRegistrant[].userD = 1111 userRegistrant[].contact = 09123456789 userRegistrant[].address = Manila userRegistrant[].password = asdf userRegistrant[].sex = F userRegistrant[].string[0] = 2020-10-10 userRegistrant[].string[1] = Moderna userRegistrant[].string[2] = Manila userRegistrant[].string[3] = 2021-01-10 userRegistrant[].string[4] = Moderna userRegistrant[].string[5] =	userRegistrant[].fullName = “Maria Clara” userRegistrant[].userD = 1111 userRegistrant[].contact = 09123456789 userRegistrant[].address = Manila userRegistrant[].password = asdf userRegistrant[].sex = F userRegistrant[].string[0] = 2020-10-10 userRegistrant[].string[1] = Moderna userRegistrant[].string[2] = Manila userRegistrant[].string[3] = 2021-01-10 userRegistrant[].string[4] = Moderna userRegistrant[].string[5] = Manila userRegistrant[].string[6] = 2021-04-10 userRegistrant[].string[7] = Moderna	P

				Manila userRegistrant[].string[6] = 2021-04-10 userRegistrant[].string[7] = Moderna userRegistrant[].string[8] = Manila	userRegistrant[].string[8] = Manila	
	2	For Appointment.txt	1222 1111 Maria Manila Pfizer 2023-04-10 12:00 2020-10-10	userAppoinment[].appId = 1212 userAppoinment[].password = asdf userAppoinment[].customerName = “Maria Clara” userAppoinment[].time = 12:00 userAppoinment[].string[0] = Manila userAppoinment[].string[1] = Pfizer userAppoinment[].string[2] = 2023-04-10 userAppoinment[].string[3] = 2020-10-10	userAppoinment[].appId = 1212 userAppoinment[].password = asdf userAppoinment[].customerName = “Maria Clara” userAppoinment[].time = 12:00 userAppoinment[].string[0] = Manila userAppoinment[].string[1] = Pfizer userAppoinment[].string[2] = 2023-04-10 userAppoinment[].string[3] = 2020-10-10	P
	3	For Chatbot.txt	What is Covid-19? infectious disease caused by the SARS-CoV-2 virus	FAQ[].question = “What is Covid-19?” FAQ[].answer = “infectious disease caused by the SARS-CoV-2 virus”	FAQ[].question = “What is Covid-19?” FAQ[].answer = “infectious disease caused by the SARS-CoV-2 virus”	P