

INDIVIDUAL ASSIGNMENT

**TECHNOLOGY PARK MALAYSIA CT018-3-1-ICP**

**INTRODUCTION TO C PROGRAMMING APU1F2209CS, APU1F2209CS(IS), APD1F2209CS(IS)**

**HAND OUT DATE: 27 March 2023**

**HAND IN DATE: 12 June 2023**

**WEIGHTAGE: 50%**

**INSTRUCTIONS TO CANDIDATES:**

1. Submit your assignment online in Moodle unless advised otherwise
2. Late submission will be awarded zero(0) unless Extenuating Circumstances (EC) are upheld
3. Cases of plagiarism will be penalized
4. You must obtain at least 50% in each component to pass this module

*Table of Contents*

[**1.0 Introduction and assumptions** 3](#_Toc137285419)

[**2.0 Design of the program** 5](#_Toc137285420)

[**3.0 Additional features** 32](#_Toc137285421)

[**4.0 Sample outputs** 38](#_Toc137285422)

[**4.1 Home menu** 38](#_Toc137285423)

[**4.1.1 Available sessions menu** 40](#_Toc137285424)

[**4.1.2 Login menu** 40](#_Toc137285425)

[**4.2.0 Admin menu** 42](#_Toc137285426)

[**4.2.1 User operation** 42](#_Toc137285427)

[**4.2.1.1 Add user** 44](#_Toc137285428)

[**4.2.1.2 Delete user** 46](#_Toc137285429)

[**4.2.1.3 View all users** 48](#_Toc137285430)

[**4.2.1.4 View user profile** 49](#_Toc137285431)

[**4.2.2 Session operation** 50](#_Toc137285432)

[**4.2.2.1 Add Session** 51](#_Toc137285433)

[**4.2.2.2 Delete session** 53](#_Toc137285434)

[**4.2.2.3 View Session** 54](#_Toc137285435)

[**4.2.2.4 Enrol User** 55](#_Toc137285436)

[**4.2.2.4 Dis-enroll User** 57](#_Toc137285437)

[**4.3.0 Tutor menu** 59](#_Toc137285438)

[**4.3.1 View my session.** 60](#_Toc137285439)

[**4.3.1 View student enrolled in my session.** 60](#_Toc137285440)

[**4.4.0 Student menu** 61](#_Toc137285441)

[**4.4.1 View my session.** 61](#_Toc137285442)

[**4.4.2 Enroll into session.** 62](#_Toc137285443)

[**5.0 Conclusion** 64](#_Toc137285444)

[**6.0 References using APA Referencing** 64](#_Toc137285445)

# **1.0 Introduction and assumptions**

APU has introduced an experimental program called the "Programming Café" to enhance students' performance, problem-solving skills, and confidence in coding. The program offers additional coding sessions outside of students' regular timetable. Each session is two hours long and takes place in a designated classroom. The Programming Café sessions are conducted by select APU Graduate Students who possess in-depth knowledge of specific programming languages. These Graduate Students will be compensated with a flat rate of RM100 for each two-hour session they conduct.

To ensure a comprehensive offering, APU has selected several commonly used programming languages for the Programming Café sessions. The list of selected programming language sessions can be found in Table 1. Additional coding subjects may be added as required, but all sessions listed in Table 1 are mandatory and will be included in the system by default.

A picture containing text, screenshot, number, font

Description automatically generated

***Figure 1*** *Table 1: Programming Café Session*

A picture containing text, font, screenshot, number

Description automatically generated

***Figure 2*** *Table 2: Tutor*

To ensure data persistence even after program termination, all information will be stored in a text file. This approach saves the data from being lost and allows for easy access using specific commands in the C programming language. Additionally, when dealing with a large volume of data, manually entering each entry can be time-consuming. However, by utilizing a file that contains all the required information, accessing and manipulating the data becomes much simpler. This method also enables easy data transfer between different computers without the need for any modifications (*C Files I/O: Opening, Reading, Writing and Closing a File*, n.d.). The program will employ common file input/output operations, such as fprintf(), fscanf(), fread(), fwrite(), and fseek(), to perform CRUD (Create, Read, Update, Delete) actions on the data.

# **2.0 Design of the program**

Record Structures  
sessions  
 session\_code S  
 title  
 day  
 start\_time  
 location  
 tutor\_code  
  
enrolled\_sessions  
 session\_code  
 user\_id  
 name  
 role  
  
users  
 user\_id  
 name  
 password  
 email  
 role  
  
tutor\_profiles  
 user\_id  
 tutor\_code  
 title  
student\_profiles  
 user\_id  
 student\_code  
  
Module declarations  
/\* void main() \*/  
main  
 setup  
 home\_menu  
END  
  
/\* void home\_menu() \*/  
home\_menu  
 SET flag = 1  
 DOWHILE flag = 1  
 PRINT "Welcome to APU Programming Cafe Management System!"  
 PRINT "1. Login  
 PRINT "2. View available sessions  
 PRINT "0. Exit  
  
 SET option  
 PRINT "Please select an option below: "  
  
 IF GET INPUT NOT int THEN  
 PRINT "[SERVER WARNING] Invalid input. Please enter a numeric integer."  
 DOWHILE getchar NOT '\n'  
 ENDO  
 CONTINUE  
 ENIF  
  
 SET response = 0  
  
 CASE OF option  
 1:  
 SET response = login\_menu  
 IF response = 0 THEN  
 SET flag = 0  
 ENDIF  
 BREAK  
 2:  
 available\_sessions\_menu  
 BREAK  
 3:  
 SET flag = 0  
 PRINT "[SERVER INFO] Thank you for using APU Programming Cafe Management System!"  
 BREAK  
 DEFAULT:  
 PRINT "[SERVER WARNING] Invalid option. Please try again."  
 ENDCASE  
 ENDO  
END  
  
/\* void available\_sessions\_menu() \*/  
available\_sessions\_menu  
 title\_printer("Available sessions")  
 SET num\_lines  
  
 SET lines, num\_lines = read("sessions.txt")  
  
 SET session FROM STRUCTURE sessions  
  
 IF NOT lines NULL THEN  
 dash\_printer(98)  
 PRINT "No. | Session code | Title | Day | Start time | Location | Tutor code"  
 dash\_printer(98)  
 DO i = 0 TO num\_lines  
 SET session.session\_code = SPLIT ";" FROM lines[i]  
 SET session.title = SPLIT ";" FROM lines[i]  
 SET session.day = SPLIT ";" FROM lines[i]  
 SET session.start\_time = SPLIT ";" FROM lines[i]  
 SET session.location = SPLIT ";" FROM lines[i]  
 SET session.tutor\_code = SPLIT ";" FROM lines[i]  
  
 PRINT "{session.session\_code} {session.title} {session.day} {session.start\_time} {session.location} {session.tutor\_code}"  
 ENDDO  
 dash\_printer(98)  
 ENDIF  
  
 DO i = 0 TO num\_lines  
 FREE lines[i]  
 ENDDO  
  
 FREE lines  
END  
  
/\* int login\_menu() \*/  
login\_menu  
 title\_printer("Login page")  
  
 SET users\_file = FILE OPEN "users.txt" WITH "r"  
  
 SET authenticated = 0  
 SET login\_attempts = 0  
  
 SET max\_login\_attempts = 3  
  
 DOWHILE login\_attempts < max\_login\_attempts  
 SET user FROM STRUCTURE users  
  
 SET user\_id  
 SET password  
  
 PRINT "Please enter your User Code (TP number or Tutor code) down below: "  
 GET INPUT user\_id  
 "Please enter your password down below: "  
 GET INPUT password  
  
 SET response, user\_id = user\_code\_parser(user\_id)  
  
 IF response = 0 THEN  
 DOWHILE fscanf(users\_file, "%[^;];%[^;];%[^;];%[^;];%[^;];\n", user.user\_id, user.name, user.password,  
 user.email, user.role) NOT EOF  
 IF user.user\_id = user\_id AND user.password = password THEN  
 SET authenticated = 1  
 BREAK  
 ENDIF  
 ENDDO  
 ENDIF  
  
 IF authenticated = 1 THEN  
 FILE CLOSE users\_file  
 PRINT "[SERVER INFO] Login successful {user.name} ({user.role})!"  
 IF user.role = "admin" THEN  
 admin\_dashboard\_menu(user)  
 ELSE IF user.role = "tutor" THEN  
 tutor\_dashboard\_menu(user)  
 ELSE IF user.role = "student" THEN  
 student\_dashboard\_menu(user)  
 ENDIF  
 BREAK  
 ELSE  
 login\_attempts = login\_attempts + 1  
 PRINT "[SERVER WARNING] Login id or password is incorrect, {login\_attempts} attempts left. Please try again."  
 ENDIF  
 ENDDO  
  
 IF login\_attempts = max\_login\_attempts THEN  
 FILE CLOSE users\_file  
 PRINT "[SERVER ERROR] Max login attempts reached. Exiting program."  
 ENIF  
 RETURN authenticated  
END  
  
/\* void admin\_dashboard\_menu(user) \*/  
admin\_dashboard\_menu(user)  
 SET flag = 1  
  
 DOWHILE flag = 1  
 title\_printer("Dashboard")  
 printf "Welcome {session\_user.name} ({session\_user.role}) !"  
 printf "1. User operation."  
 printf "2. Session operation."  
 printf "0. Logout"  
  
 SET option  
 PRINT "Please select an option below: "  
 IF GET INPUT NOT int THEN  
 PRINT "[SERVER WARNING] Invalid input. Please enter a numeric integer."  
 DOWHILE getchar NOT '\n'  
 ENDO  
 CONTINUE  
 ENDIF  
  
 CASE OF option  
 1:  
 user\_operation\_menu(session\_user)  
 BREAK  
 2:  
 session\_operation\_menu(session\_user)  
 BREAK  
 3:  
 SET flag = 0  
 BREAK  
 DEFAULT:  
 PRINT "[SERVER WARNING] Invalid option. Please try again."  
 BREAK  
 ENDCASE  
 ENDDO  
END  
  
/\* void user\_operation\_menu(user) \*/  
user\_operation\_menu(user)  
 SET flag = 1  
 DOWHILE flag  
 title\_printer("User operation")  
 PRINT "1. Add user."  
 PRINT "2. Delete user."  
 PRINT "3. View all user."  
 PRINT "4. View user profile."  
 PRINT "0. Back to dashboard."  
 SET OPTION  
 PRINT "Please select an option below: "  
  
 IF GET INPUT NOT int THEN  
 PRINT "[SERVER WARNING] Invalid input. Please enter a numeric integer."  
 DOWHILE getchar NOT '\n'  
 ENDO  
 CONTINUE  
 ENDIF  
  
 CASE OF option  
 1:  
 add\_user\_menu(session\_user)  
 BREAK  
 2:  
 delete\_user\_menu(session\_user)  
 BREAK  
 3:  
 view\_all\_user\_menu(session\_user)  
 BREAK  
 4:  
 view\_user\_menu(session\_user)  
 BREAK  
 0:  
 SET flag = 0  
 BREAK  
 DEFAULT:  
 PRINT "[SERVER WARNING] Invalid option. Please try again."  
 BREAK  
 ENDCASE  
 ENDDO  
END  
  
/\* void add\_user\_menu(users session\_user) \*/  
add\_user\_menu(users session\_user)  
 SET user FROM STRUCTURE users  
 title\_printer("User operation - Add user")  
  
 DOWHILE 1  
 PRINT "Please enter the user's role (student or tutor) down below: "  
 GET INPUT user.role  
 IF user.role NOT "student" OR user.role NOT "tutor" THEN  
 PRINT "[SERVER WARNING] Invalid role. Please try again."  
 BREAK  
 ELSE  
 PRINT "[SERVER ERROR] Invalid role. Please try again."  
 ENDIF  
 ENDO  
  
 PRINT "Please enter the user's name down below: "  
 GET INPUT user.name  
  
 DOWHILE 1  
 PRINT "Please enter the user's user id down below: "  
 GET INPUT user.user\_id  
 IF user.user\_id LENGTH = 6 THEN  
 BREAK  
 ELSE  
 PRINT "[SERVER WARNING] User id must be exactly 6 characters long. Please try again."  
 ENDIF  
 ENDO  
  
 DOWHILE 1  
 PRINT "Please enter the user's password down below: "  
 GET INPUT user.password  
 IF user.user\_id LENGTH >= 8 THEN  
 BREAK  
 ELSE  
 PRINT "[SERVER WARNING] Password must be at least 8 characters long. Please try again."  
 ENDIF  
 ENDO  
  
 PRINT "Please enter the user's email down below: "  
 GET INPUT user.email  
  
 IF user.role = "student" THEN  
 SET student\_code  
 SET student FROM STRUCTURE student\_profiles  
  
 student.user\_id = user.user\_id  
 student\_code = "TP" + user.user\_id  
 student.student\_code = student\_code  
  
 SET student\_profiles\_file = FILE OPEN "student\_profiles.txt" WITH "a"  
 PRINT student INTO student\_profiles\_file  
 FILE CLOSE student\_profiles\_file  
 ENDIF  
  
 IF user.role = "tutor" THEN  
 SET tutor FROM STRUCTURE tutor\_profiles  
 SET tutor\_code  
  
 PRINT "Please enter the tutor's title down below: "  
 GET INPUT tutor.title  
  
 tutor.user\_id = user.user\_id  
 tutor\_code = "TU" + user.user\_id  
 tutor.tutor\_code = tutor\_code  
  
 SET tutor\_profiles\_file = FILE OPEN "tutor\_profiles.txt" WITH "a"  
 PRINT tutor INTO tutor\_profiles\_file  
 FILE CLOSE tutor\_profiles\_file  
 ENDIF  
  
 SET users\_file = FILE OPEN "users.txt" WITH "a"  
 PRINT user INTO users\_file  
 FILE CLOSE users\_file  
 PRINT "[SERVER INFO] User added successfully!"  
END  
  
/\* void delete\_user\_menu(users session\_user) \*/  
delete\_user\_menu(users session\_user)  
 title\_printer("User operation - Delete user")  
 SET user\_id  
  
 PRINT "Please enter the user code of the user you want to delete: "  
 GET INPUT user\_id  
  
 SET response, user\_id = user\_code\_parser(user\_id)  
  
 IF RESPONSE = 0 THEN  
 PRINT "[SERVER WARNING] Invalid user code."  
 RETURN  
 ENDIF  
  
 SET user FROM STRUCTURE users = get\_user(user\_id)  
  
 IF user.user\_id = "" THEN  
 PRINT "[SERVER ERROR] User does not exist."  
 RETURN  
 ELSE  
 SET users\_temp\_file = FILE OPEN "users\_temp.txt" WITH "w"  
 SET users\_file = FILE OPEN "users.txt" WITH "r"  
  
 SET is\_student = 0  
 SET is\_tutor = 0  
  
 DOWHILE fscanf(users\_file, "%[^;];%[^;];%[^;];%[^;];%[^;];\n", user.user\_id, user.name, user.password,  
 user.email, user.role) NOT EOF  
 IF NOT user.user\_id = user\_id THEN  
 STORE user.user\_id ";" user.name ";"user.password ";"user.email ";" user.role INTO temp\_users\_file  
 ELSE  
 IF user.role = "student" THEN  
 SET is\_student = 1  
 ELSE IF user.role = "tutor" THEN  
 SET is\_tutor = 1  
 ENDIF  
 ENDIF  
 ENDO  
  
 FILE CLOSE users\_file  
 FILE CLOSE users\_temp\_file  
  
 FILE REMOVE "users.txt"  
 FILE RENAME "users\_temp.txt" TO "users.txt"  
  
 SET enrolled\_sessions\_file = FILE OPEN "enrolled\_sessions.txt" WITH "r"  
 SET enrolled\_sessions\_temp\_file = FILE OPEN "enrolled\_sessions\_temp\_file.txt" WITH "w"  
  
 SET es FROM STRUCTURE enrolled\_sessions  
  
 DOWHILE fscanf(users\_file, "%[^;];%[^;];%[^;];%[^;];%[^;];\n", es.session\_code, es.user\_id,es.name, es.role) NOT EOF  
 IF NOT user.user\_id = user\_id THEN  
 STORE es.session\_code";"es.user\_id";"es.name";"user.email";"es.role INTO temp\_users\_file  
 ENDIF  
 ENDO  
  
 FILE CLOSE enrolled\_sessions\_file  
 FILE CLOSE enrolled\_sessions\_temp\_file  
  
 FILE REMOVE "enrolled\_sessions.txt"  
 FILE RENAME "enrolled\_sessions\_temp.txt" TO "enrolled\_sessions.txt"  
  
 IF is\_student = 1 THEN  
 SET student\_profiles\_file = FILE OPEN "student\_profiles.txt" WITH "r"  
 SET student\_profiles\_temp\_file = FILE OPEN "student\_profiles\_temp.txt" WITH "w"  
  
 SET student FROM STRUCTURE student\_profiles  
  
 DOWHILE NOT fscanf(student\_profiles\_file, "%[^;];%[^;];\n", student.user\_id, student.student\_code) = EOF  
 IF student.user\_id NOT user\_id THEN  
 STORE student.user\_id ";" student.student\_code INTO student\_profiles\_temp\_file  
 ENDIF  
 ENDDO  
  
 FILE CLOSE student\_profiles\_file  
 FILE CLOSE student\_profiles\_temp\_file  
  
 FILE REMOVE "student\_profiles.txt"  
 FILE RENAME "student\_profiles\_temp.txt" TO "student\_profiles.txt"  
  
 PRINT "[SERVER INFO] User deleted successfully!"  
 ENDIF  
  
 IF is\_tutor = 1 THEN  
 SET tutor\_profiles\_file = FILE OPEN "tutor\_profiles.txt" WITH "r"  
 SET tutor\_profiles\_temp\_file = FILE OPEN "tutor\_profiles\_temp.txt" WITH "w"  
  
 SET tutor FROM STRUCTURE  
  
 DOWHILE NOT fscanf(tutor\_profiles\_file, "%[^;];%[^;];%[^;];\n", tutor.user\_id, tutor.tutor\_code, tutor.title) = EOF  
 IF tutor.user\_id NOT user\_id THEN  
 STORE tutor.user\_id ";" tutor.tutor\_code ";" tutor.title INTO tutor\_profiles\_temp\_file  
 ENDIF  
 ENDDO  
  
 FILE CLOSE tutor\_profiles\_file  
 FILE CLOSE tutor\_profiles\_temp\_file  
  
 FILE REMOVE "tutor\_profiles.txt"  
 FILE RENAME "tutor\_profiles\_temp.txt" TO "tutor\_profiles.txt"  
  
 PRINT "[SERVER INFO] User deleted successfully!"  
 ENDIF  
 ENDIF  
END  
  
/\* void view\_all\_user\_menu(users session\_user) \*/  
view\_all\_user\_menu(users session\_user)  
 title\_printer("User operation - View all users")  
  
 SET number\_lines  
 SET lines, num\_lines = read("sessions.txt")  
  
 SET user FROM STRUCTURE users  
  
 IF NOT lines = NULL THEN  
 dash\_printer(91)  
 PRINT "No. | User id | Name | Email | Password | Role |"  
 dash\_printer(91)  
 DO i = 0 TO num\_lines  
 SET session.user\_id = SPLIT ";" FROM lines[i]  
 SET session.name = SPLIT ";" FROM lines[i]  
 SET session.password = SPLIT ";" FROM lines[i]  
 SET session.email = SPLIT ";" FROM lines[i]  
 SET session.role = SPLIT ";" FROM lines[i]  
  
 PRINT "{user.user\_id} {user.name} {user.password} {user.email} {user.role}"  
 ENDDO  
 dash\_printer  
  
 DO i = 0 TO num\_lines  
 FREE lines[i]  
 ENDDO  
  
 FREE lines  
 ENDIF  
END  
  
/\* void view\_user\_menu(users session\_user) \*/  
view\_user\_menu(users session\_user)  
 title\_printer("User operation - View user profile")  
  
 SET user\_id  
  
 PRINT "Please enter the user code of the user you want to view: "  
 GET INPUT user\_id  
  
 SET response, user\_id = user\_code\_parser(user\_id)  
  
 IF response = 0 THEN  
 PRINT "[SERVER WARNING] Invalid user code."  
 RETURN  
 ENDIF  
  
 SET user FROM STRUCTURE users = get\_user(user\_id)  
  
 IF user.user\_id = "" THEN  
 PRINT "[SERVER ERROR] User does not exist."  
 RETURN  
 ELSE  
 PRINT "## User Details ##"  
 dash\_printer(85)  
 PRINT "User ID | "Name | Password | Email | Role"  
 dash\_printer(85);  
 PRINT "{user.user\_id} {user.name} {user.password} {user.email} {user.role}"  
 dash\_printer(85)  
  
 PRINT "## User Profile ##"  
 IF user.role = "student" THEN  
 dash\_printer(16);  
 SET student FROM STRUCTURE student\_profiles = get\_student(user\_id)  
 PRINT "Student Code"  
 dash\_printer(16);  
 PRINT "{student.student\_code}"  
 dash\_printer(16)  
 ELSE IF user.role = "tutor" THEN  
 SET tutor FROM STRUCTURE tutor\_profiles = get\_tutor(user\_id)  
 dash\_printer(50);  
 PRINT "Tutor Code", "Title"  
 dash\_printer(50);  
 PRINT "{tutor.tutor\_code} {tutor.title}"  
 dash\_printer(50);  
 PRINT "{tutor.tutor\_code}"  
 ENDIF  
 PRINT "\n"  
  
 SET num\_sessions = 0  
  
 SET sessions FROM STRUCTURE enrolled\_sessions, num\_sessions = get\_enrolled\_sessions("user\_id", user\_id)  
  
 PRINT "## Enrolled Sessions ##"  
 dash\_printer(55)  
 PRINT "No. | Session | User ID | Name | Role"  
 dash\_printer(55)  
  
 IF num\_sessions > 0 THEN  
 DO i = 0 TO num\_sessions  
 PRINT "{i + 1} {session.session\_id} {user.user\_id} {user.name} {user.role}"  
 ENDDO  
 ELSE  
 PRINT "No enrolled sessions."  
 ENDIF  
  
 dash\_printer(55)  
  
 free(sessions)  
 ENDIF  
END  
  
/\* void session\_operation\_menu(users session\_user) \*/  
session\_operation\_menu(users session\_user)  
 SET flag = 1  
 DOWHILE flag  
 title\_printer("Session operation")  
 PRINT "1. Add session.  
 PRINT "2. Delete session.  
 PRINT "3. View session.  
 PRINT "4. Enroll a user.  
 PRINT "5. Disenroll a user.  
 SET OPTION  
 PRINT "Please select an option below: "  
  
 IF GET INPUT NOT int THEN  
 PRINT "[SERVER WARNING] Invalid input. Please enter a numeric integer."  
 DOWHILE getchar NOT '\n'  
 ENDO  
 CONTINUE  
 ENDIF  
  
 CASE OF option  
 1:  
 add\_session\_menu(session\_user)  
 BREAK  
 2:  
 delete\_session\_menu(session\_user)  
 BREAK  
 3:  
 view\_session\_menu(session\_user)  
 BREAK  
 4:  
 enroll\_user\_menu(session\_user)  
 BREAK  
 5:  
 disenroll\_user\_menu(session\_user)  
 BREAK  
  
 0:  
 SET flag = 0  
 BREAK  
 DEFAULT:  
 PRINT "[SERVER WARNING] Invalid option. Please try again."  
 BREAK  
 ENDCASE  
 ENDDO  
END  
  
/\* void add\_session\_menu(users session\_user) \*/  
add\_session\_menu(users session\_user)  
 title\_printer("Session operation - Add session")  
  
 SET sessions\_file = FILE OPEN "sessions.txt" WITH "a"  
 SET enrolled\_sessions\_file = FILE OPEN "enrolled\_sessions.txt" WITH "a"  
  
 SET session FROM STRUCTURE sessions  
  
 DOWHILE 1  
 PRINT "Please enter the session code: "  
 GET INPUT session.session\_code  
  
 IF LEN session.session\_code = 6 THEN  
 BREAK  
 ELSE  
 PRINT "[SERVER WARNING] Session code must be 6 characters long."  
 ENDIF  
 ENDDO  
  
 SET existing\_session FROM STRUCTURE sessions = get\_session("session\_code, "session.session\_code)  
  
 IF existing\_session.session\_code != "" THEN  
 PRINT "[SERVER WARNING] [SERVER ERROR] Session code already exists."  
 RETURN  
 ENDIF  
  
 PRINT "Please enter the session name: "  
 GET INPUT session.session\_name  
  
 PRINT "Please enter the session day: "  
 GET INPUT session.session\_day  
  
 PRINT "Please enter the session start time: "  
 GET INPUT session.session\_start\_time  
  
 PRINT "Please enter the session location: "  
 GET INPUT session.session\_location  
  
 PRINT "Enroll a tutor into the session: "  
 GET INPUT session.session\_tutor  
  
 SET response, session.tutor\_cod = user\_code\_parser(session.tutor\_code)  
  
 IF response = 0 THEN  
 PRINT "[SERVER WARNING] Invalid user code.\n"  
 RETURN  
 ENDIF  
  
 session.tutor\_code = user\_id  
  
 SET num\_sessions = 0  
 SET tutor FROM STRUCTURE users = get\_tutor(user\_id)  
 SET sessions FROM STRUCTURE enrolled\_sessions, num\_sessions = get\_enrolled\_sessions("user\_id", user\_id)  
  
 IF num\_sessions < 1 && tutor.role = tutor" THEN  
 STORE session.session\_code ";" session.title ";" session.session\_day ";" session.day ";" session.start\_time ";" session.location ";" session.tutor\_code TO sessions\_file  
 STORE session.session\_code ";" user\_id ";" tutor.name ";" tutor.role TO enrolled\_sessions\_file  
 PRINT "[SERVER INFO] Session added successfully!"  
 ELSE  
 PRINT "[SERVER ERROR] Tutor does not exist or has reached the maximum number of sessions."  
 ENDIF  
  
 FILE CLOSE sessions\_file  
 FILE CLOSE enrolled\_sessions\_file  
  
 free(sessions)  
END  
  
/\* void delete\_session\_menu(users session\_user) \*/  
delete\_session\_menu(users session\_user)  
 title\_printer("Session operation - Delete session")  
  
 PRINT "Please enter the session code: "  
 SET session\_code  
 GET INPUT session\_code  
  
 SET session FROM STRUCTURE sessions = get\_session("session\_code", session\_code)  
  
 IF session.session\_code = "" THEN  
 PRINT "[SERVER ERROR] Session does not exist."  
 RETURN  
 ENDIF  
  
 sessions\_file = FILE OPEN "sessions.txt" WITH "r"  
 sessions\_temp\_file = FILE OPEN "sessions\_temp.txt" WITH "w"  
 enrolled\_sessions\_file = FILE OPEN "enrolled\_sessions.txt" WITH "r"  
 enrolled\_sessions\_temp\_file = FILE OPEN "enrolled\_sessions\_temp.txt" WITH "w"  
  
 SET line  
 DOWHILE fgets(line, sizeof(line), sessions\_file)  
 IF strstr(line, session\_code) = NULL THEN  
 STORE line TO sessions\_temp\_file  
 ENDIF  
 ENDDO  
  
 DOWHILE fgets(line, sizeof(line), enrolled\_sessions\_file)  
 IF strstr(line, session\_code) == NULL THEN  
 STORE line TO enrolled\_sessions\_temp\_file  
 ENDIF  
 ENDO  
  
  
 FILE CLOSE sessions\_file  
 FILE CLOSE sessions\_temp\_file  
 FILE CLOSE enrolled\_sessions\_file  
 FILE CLOSE enrolled\_sessions\_temp\_file  
  
 FILE REMOVE "sessions.txt"  
 FILE RENAME "sessions\_temp.txt" TO "sessions.txt"  
  
 FILE REMOVE "enrolled\_sessions.txt"  
 FILE RENAME "enrolled\_sessions\_temp.txt" TO "enrolled\_sessions.txt"  
  
 PRINT "[SERVER INFO] Session deleted successfully!"  
END  
  
/\* void view\_session\_menu(users session\_user) \*/  
view\_session\_menu(users session\_user)  
 title\_printer("Session operation - View session")  
  
 PRINT "Please enter the session code: "  
 SET session\_code  
 GET INPUT session\_code  
  
 SET session FROM STRUCTURE sessions = get\_session("session\_code", session\_code)  
  
 IF session.session\_code = "" THEN  
 PRINT "[SERVER ERROR] Session does not exist."  
 RETURN  
 ENDIF  
  
 PRINT "## Session Details ##"  
 dash\_printer(92);  
 PRINT "Session Code | Title | Day | Start Time | Location | Tutor Code"  
 dash\_printer(92);  
 PRINT "{session.session\_code} {session.title} {session.day} {session.start\_time} {session.location} {session.tutor\_code}"  
 dash\_printer(92);  
 PRINT "\n"  
  
 SET num\_sessions = 0  
 SET enrolled\_sessions FROM STRUCTURE enrolled\_sessions, num\_sessions = get\_enrolled\_sessions("session\_code", session\_code)  
 PRINT "## Enrolled Users ##"  
 dash\_printer(55)  
 PRINT "No. | Session | User ID | Name | Role"  
 dash\_printer(55)  
  
 IF num\_sessions > 0 THEN  
 DO i = 0 TO num\_sessions  
 PRINT "{i + 1} {enrolled\_sessions[i].session\_code} {enrolled\_sessions[i].user\_id} {enrolled\_sessions[i].name} {enrolled\_sessions[i].role}"  
 ENDDO  
 ELSE  
 PRINT "No enrolled users."  
 ENDIF  
 dash\_printer(55)  
 FREE enrolled\_sessions  
END  
  
/\* void enroll\_user\_menu(users session\_user) \*/  
enroll\_user\_menu(users session\_user)  
 title\_printer("Session operation - Enroll user")  
  
 SET enrolled\_sessions\_file = FILE OPEN "enrolled\_sessions.txt" WITH "a"  
 PRINT "Please enter the session code: "  
 SET session\_code  
 GET INPUT session\_code  
  
 SET session FROM STRUCTURE sessions = get\_session("session\_code", session\_code)  
  
 IF session.session\_code = "" THEN  
 PRINT "[SERVER ERROR] Session does not exist."  
 RETURN  
 ENDIF  
  
 PRINT "Please enter the user code: "  
 SET user\_id  
 GET INPUT user\_id  
  
 SET response, user\_id = user\_code\_parser(user\_code)  
  
 IF response = 0 THEN  
 PRINT "[SERVER WARNING] Invalid user code.\n"  
 RETURN  
 ENDIF  
  
 SET user FROM STRUCTURE users = get\_user(user\_id)  
  
 IF user.user\_id = "" THEN  
 PRINT "[SERVER ERROR] User does not exist."  
 RETURN  
 ENDIF  
  
 IF user.role = "tutor" THEN  
 PRINT "[SERVER ERROR] Tutor cannot be enrolled in a session here."  
 RETURN  
 ENDIF  
  
 SET num\_sessions = 0  
 SET sessions FROM STRUCTURE enrolled\_sessions, num\_sessions = get\_enrolled\_sessions("session\_code", session\_code)  
  
 DO i = 0 TO num\_sessions  
 IF sessions[i].user\_id = user\_id THEN  
 PRINT "[SERVER ERROR] User is already enrolled in this session."  
 FREE sessions  
 RETURN  
 ENDIF  
 ENDDO  
  
 FREE sessions  
  
 STORE session.session\_code ";" user\_id ";" user.name ";" user.role TO enrolled\_sessions\_file  
  
 FILE CLOSE enrolled\_sessions\_file  
  
 PRINT "[SERVER INFO] User enrolled successfully!"  
END  
  
/\* void disenroll\_user\_menu(users session\_user) \*/  
disenroll\_user\_menu(users session\_user)  
 title\_printer("Session operation - Disenroll user")  
  
 PRINT "Please enter the session code: "  
 SET session\_code  
 GET INPUT session\_code  
  
 SET session FROM STRUCTURE sessions = get\_session("session\_code", session\_code)  
  
 IF session.session\_code = "" THEN  
 PRINT "[SERVER ERROR] Session does not exist."  
 RETURN  
 ENDIF  
  
 PRINT "Please enter the user code: "  
 SET user\_id  
 GET INPUT user\_id  
  
 SET response, user\_id = user\_code\_parser(user\_code)  
  
 IF response = 0 THEN  
 PRINT "[SERVER WARNING] Invalid user code.\n"  
 RETURN  
 ENDIF  
  
 SET user FROM STRUCTURE users = get\_user(user\_id)  
  
 IF user.user\_id = "" THEN  
 PRINT "[SERVER ERROR] User does not exist."  
 RETURN  
 ENDIF  
  
 IF user.role = "tutor" THEN  
 PRINT "[SERVER ERROR] Tutor cannot be dis-enroll."  
 RETURN  
 ENDIF  
  
 SET num\_sessions = 0  
 SET sessions FROM STRUCTURE enrolled\_sessions, num\_sessions = get\_enrolled\_sessions("session\_code", session\_code)  
  
 SET flag = 0  
  
 DO i = 0 TO num\_sessions  
 IF sessions[i].user\_id = user\_id THEN  
 SET flag = 1  
 BREAK  
 ENDIF  
 ENDDO  
  
 FREE sessions  
  
 IF flag = 0 THEN  
 PRINT "[SERVER ERROR] Enrolled session does not exist."  
 RETURN  
 ENDIF  
  
 SET enrolled\_sessions\_file = FILE OPEN "enrolled\_sessions.txt" WITH "r"  
 SET enrolled\_sessions\_temp\_file = FILE OPEN "enrolled\_sessions\_temp.txt" WITH "w"  
  
 SET e\_session FROM STRUCTURE enrolled\_sessions  
  
 DOWHILE fscanf(enrolled\_sessions\_file, "%[^;];%[^;];%[^;];%[^;];\n", e\_session.session\_code, e\_session.user\_id, e\_session.name, e\_session.role) != EOF  
 IF e\_session.session\_code = session\_code && e\_session.user\_id = user\_id THEN  
 CONTINUE  
 ENDIF  
  
 STORE e\_session.session\_code ";" e\_session.user\_id ";" e\_session.name ";" e\_session.role TO enrolled\_sessions\_temp\_file  
 ENDDO  
  
 FILE CLOSE enrolled\_sessions\_file  
 FILE CLOSE enrolled\_sessions\_temp\_file  
  
 FILE REMOVE "enrolled\_sessions.txt"  
 FILE RENAME "enrolled\_sessions\_temp.txt" TO "enrolled\_sessions.txt"  
  
 PRINT "[SERVER INFO] User dis-enrolled successfully!"  
END  
  
/\* void tutor\_dashboard\_menu(users session\_user) \*/  
void tutor\_dashboard\_menu(users session\_user)  
 SET flag = 1  
  
 DOWHILE flag = 1  
 title\_printer("Dashboard")  
 printf "Welcome {session\_user.name} ({session\_user.role}) !"  
 printf "1. View my sessions."  
 printf "2. View students enrolled in sessions."  
 printf "0. Logout"  
  
 SET option  
 PRINT "Please select an option below: "  
 IF GET INPUT NOT int THEN  
 PRINT "[SERVER WARNING] Invalid input. Please enter a numeric integer."  
 DOWHILE getchar NOT '\n'  
 ENDO  
 CONTINUE  
 ENDIF  
  
 CASE OF option  
 1:  
 view\_my\_sessions\_menu(session\_user)  
 BREAK  
 2:  
 view\_students\_enrolled\_in\_sessions\_menu(session\_user)  
 BREAK  
 3:  
 SET flag = 0  
 BREAK  
 DEFAULT:  
 PRINT "[SERVER WARNING] Invalid option. Please try again."  
 BREAK  
 ENDCASE  
 ENDDO  
END  
  
/\* void view\_my\_sessions\_menu(users session\_user) \*/  
view\_my\_sessions\_menu(users session\_user)  
 title\_printer("View students enrolled in sessions")  
  
 SET tutor\_code = "TU"  
 tutor\_code = tutor\_code + session\_user.user\_id  
  
 SET session FROM STRUCTURE sessions = get\_session("tutor\_code", tutor\_code)  
  
 SET num\_sessions = 0  
 SET enrolled\_sessions FROM STRUCTURE enrolled\_sessions, num\_sessions = get\_enrolled\_sessions("session\_code", session.session\_code)  
  
 PRINT "## Enrolled Students ##"  
 dash\_printer(55)  
  
 PRINT "No. | Session | User ID | Name | Role"  
 dash\_printer(55)  
  
 IF num\_sessions > 0 THEN  
 DO i = 0 TO num\_sessions  
 IF enrolled\_sessions[i].role = "student" THEN  
 PRINT "{i} | {enrolled\_sessions[i].session\_code} | {enrolled\_sessions[i].user\_id} | {enrolled\_sessions[i].name} | {enrolled\_sessions[i].role}"  
 ENDIF  
 ENDDO  
 ELSE  
 PRINT "No enrolled student."  
 ENDIF  
 dash\_printer(55)  
  
 FREE enrolled\_sessions  
END  
  
/\* voide student\_dashboard\_menu(users session\_user) \*/  
student\_dashboard\_menu(users session\_user)  
 SET flag = 1  
  
 DOWHILE flag = 1  
 title\_printer("Dashboard")  
 printf "Welcome {session\_user.name} ({session\_user.role}) !"  
 printf "1. View my sessions."  
 printf "2. Enroll into session."  
 printf "0. Logout"  
  
 SET option  
 PRINT "Please select an option below: "  
 IF GET INPUT NOT int THEN  
 PRINT "[SERVER WARNING] Invalid input. Please enter a numeric integer."  
 DOWHILE getchar NOT '\n'  
 ENDO  
 CONTINUE  
 ENDIF  
  
 CASE OF option  
 1:  
 view\_my\_sessions\_menu(session\_user)  
 BREAK  
 2:  
 enroll\_into\_session\_menu(session\_user)  
 BREAK  
 3:  
 SET flag = 0  
 BREAK  
 DEFAULT:  
 PRINT "[SERVER WARNING] Invalid option. Please try again."  
 BREAK  
 ENDCASE  
 ENDDO  
END  
  
/\* void view\_my\_sessions\_menu(users session\_user) \*/  
view\_my\_sessions\_menu(users session\_user)  
 title\_printer("View my sessions")  
  
 SET num\_sessions = 0  
 SET enrolled\_sessions FROM STRUCTURE enrolled\_sessions, num\_sessions = get\_enrolled\_sessions("user\_id", session\_user.user\_id)  
  
 PRINT "## Enrolled Sessions ##"  
 dash\_printer(55)  
  
 PRINT "No. | Session | User ID | Name | Role"  
 dash\_printer(55)  
  
 IF num\_sessions > 0 THEN  
 DO i = 0 TO num\_sessions  
 PRINT "{i+1} | {enrolled\_sessions[i].session\_code} | {enrolled\_sessions[i].user\_id} | {enrolled\_sessions[i].name} | {enrolled\_sessions[i].role}"  
 ENDDO  
 ELSE  
 PRINT "No enrolled session."  
 ENDIF  
 dash\_printer(55)  
  
 FREE enrolled\_sessions  
END  
  
/\* void enroll\_into\_session\_menu(users session\_user) \*/  
enroll\_into\_session\_menu(users session\_user)  
 title\_printer("Enroll into session")  
  
 SET num\_lines = 0  
 SET lines, num\_lines = read("sessions.txt")  
  
 SET available\_session FROM STRUCTURE sessions  
  
 IF NOT lines = NULL THEN  
 dash\_printer(98)  
 PRINT "No. | Session Code | Title | Day | Start time | Location| Tutor Code"  
 dash\_printer(98)  
  
 DO i = 0 TO num\_lines  
 SET session.session\_code = SPLIT ";" FROM lines[i]  
 SET session.title = SPLIT ";" FROM lines[i]  
 SET session.day = SPLIT ";" FROM lines[i]  
 SET session.start\_time = SPLIT ";" FROM lines[i]  
 SET session.location = SPLIT ";" FROM lines[i]  
 SET session.tutor\_code = SPLIT ";" FROM lines[i]  
  
 PRINT "{i+1} | {session.session\_code} | {session.title} | {session.day} | {session.start\_time} | {session.location} | {session.tutor\_code}"  
 ENDDO  
 dash\_printer(98)  
  
 DO i = 0 TO num\_lines  
 FREE lines[i]  
 ENDDO  
  
 FREE lines  
 ENDIF  
  
 PRINT "Please enter the session code: "  
 SET session\_code  
 GET INPUT session\_code  
  
 SET session FROM STRUCTURE sessions = get\_session("session\_code", session\_code)  
  
 IF session.session\_code = "" THEN  
 PRINT "[SERVER WARNING] Session does not exist."  
 RETURN  
 ENDIF  
  
 SET num\_sessions = 0  
 SET enrolled\_sessions FROM STRUCTURE enrolled\_sessions, num\_sessions = get\_enrolled\_sessions("user\_id", session\_user.user\_id)  
  
 DO i = 0 TO num\_sessions  
 IF enrolled\_sessions[i].session\_code = session.session\_code THEN  
 PRINT "[SERVER WARNING] You have already enrolled into this session."  
 FREE enrolled\_sessions  
 RETURN  
 ENDIF  
 ENDDO  
  
 FREE enrolled\_sessions  
  
 SET enrolled\_sessions\_file = FILE OPEN "enrolled\_sessions.txt" WITH "a"  
 FILE STORE "{session.session\_code} {session\_user.user\_id} {session\_user.name} {session\_user.role}" TO enrolled\_sessions\_file  
  
 FILE CLOSE enrolled\_sessions\_file  
  
 PRINT "[SERVER SUCCESS] You have successfully enrolled into {session.title}."  
END  
  
/\* void setup() \*/  
setup()  
 SET sessions\_file = FILE OPEN "sessions.txt" WITH "w"  
 SET enrolled\_sessions\_file = FILE OPEN "enrolled\_sessions.txt" WITH "w"  
 SET users\_file = FILE OPEN "users.txt" WITH "w"  
 SET tutor\_profiles\_file = FILE OPEN "tutor\_profiles.txt" WITH "w"  
 SET student\_profiles\_file = FILE OPEN "student\_profiles.txt" WITH "w"  
  
 SET default\_sessions =  
 [["PYP101", "Python Programming", "Saturday", "9.00am", "C-01-01", "TU265663"]  
 ["JAV102", "Java Programming", "Sunday", "9.00am", "C-01-02", "TU009650"]  
 ["CSC103", "C Programming", "Saturday", "2.00pm", "C-01-03", "TU544654"]  
 ["WEB104", "Web Development", "Sunday", "2.00pm", "C-01-04", "TU577001"]  
 ["CSP105", "C Sharp Programming", "Monday", "7.00pm", "C-01-05", "TU683357"]]  
  
 SET default\_enrolled\_sessions =  
 [["PYP101", "265663", "Mary", "tutor"]  
 ["JAV102", "009650", "Peter", "tutor"]  
 ["CSC103", "544654", "James", "tutor"]  
 ["WEB104", "577001", "Johnny", "tutor"]  
 ["CSP105", "683357", "David", "tutor"]  
 ["CSP105", "072187", "Jammie", "student"]]  
  
 SET default\_users =  
 [["123123", "John", "123123", "admin@apu.edu.my", "admin"]  
 ["265663", "Mary", "123123", "marry@apu.edu.my", "tutor"]  
 ["009650", "Peter", "123123", "peter@apu.edu.my", "tutor"]  
 ["544654", "James", "123123", "james@apu.edu.my", "tutor"]  
 ["577001", "Johnny", "123123", "john@apu.edu.my", "tutor"]  
 ["683357", "David", "123123", "david@apu.edu.my", "tutor"]  
 ["293333", "howard", "123123", "howard@apu.edu.my", "tutor"]  
 ["072187", "Jammie", "123123", "jamie@apu.edu.my", "student"]  
 ["073188", "Jin", "123123", "jin@apu.edu.my", "student"]]  
  
 SET default\_tutor\_profiles =  
 [["265663", "TU265663", "Python Programming Tutor"]  
 ["009650", "TU009650", "Java Programming Tutor"]  
 ["544654", "TU544654", "C Programming Tutor"]  
 ["577001", "TU577001", "Web Development Tutor"]  
 ["683357", "TU683357", "C Sharp Programming Tutor"]  
 ["073188", "TU073188", "Null"]]  
 SET default\_student\_profiles =  
 [["072187", "TP072187"]  
 ["073188", "TP073188"]]  
  
 DO i = 0 TO 5  
 DO j = 0 TO 6  
 FILE STORE "{default\_sessions[i][j]};" TO sessions\_file  
 ENDDO  
 FILE STORE "\n" TO sessions\_file  
 ENDDO  
  
 DO i = 0 TO 6  
 DO j = 0 TO 4  
 FILE STORE "{default\_enrolled\_sessions[i][j]};" TO enrolled\_sessions\_file  
 ENDDO  
 FILE STORE "\n" TO sessions\_file  
 ENDDO  
  
 DO i = 0 TO 9  
 DO j = 0 TO 5  
 FILE STORE "{default\_users[i][j]};" TO users\_file  
 ENDDO  
 FILE STORE "\n" TO sessions\_file  
 ENDDO  
  
 DO i = 0 TO 6  
 DO j = 0 TO 3  
 FILE STORE "{default\_tutor\_profiles[i][j]};" TO tutor\_profiles\_file  
 ENDDO  
 FILE STORE "\n" TO sessions\_file  
 ENDDO  
  
 DO i = 0 TO 2  
 DO j = 0 TO 2  
 FILE STORE "{default\_student\_profiles[i][j]};" TO student\_profiles\_file  
 ENDDO  
 FILE STORE "\n" TO sessions\_file  
 ENDDO  
  
 FILE CLOSE sessions\_file  
 FILE CLOSE enrolled\_sessions\_file  
 FILE CLOSE users\_file  
 FILE CLOSE tutor\_profiles\_file  
 FILE CLOSE student\_profiles\_file  
END  
  
/\* char \*\*read(char \*filename, int \*num\_lines) \*/  
\*\*read(char \*filename, int \*num\_lines)  
 SET lines  
  
 SET fp = FILE OPEN filename WITH "r"  
  
 IF fp = NULL THEN  
 PRINT "[SERVER WARNING] Failed to open file {filename}."  
 RETURN NULL  
 ENDIF  
  
 SET count = 0  
 SET c  
  
 DOWHILE NOT c = fgetc = EOF  
 IF c = '\n' THEN  
 count = count + 1  
 ENDIF  
 ENDDO  
  
 REWIND fp  
  
 SET lines = ALLOCATE count \* sizeof(char \*)  
  
 IF lines = NULL THEN  
 PRINT "[SERVER WARNING] Failed to allocate memory for lines array."  
 RETURN NULL  
 ENDIF  
  
 DO i = 0 TO count  
 SET lines[i] = ALLOCATE MAX\_STRING\_LENGTH \* sizeof(char)  
 ENDDO  
  
 SET i = 0  
 DOWHILE NOT fgets(lines[i], MAX\_STRING\_LENGTH, fp) == NULL  
 i = i + 1  
 ENDDO  
  
 FILE CLOSE fp  
 SET num\_lines = count  
  
 RETURN lines  
END  
  
/\* void title\_printer(char \*title) \*/  
title\_printer(char \*title)  
 SET str\_len = LEN title  
 SET line\_len = 4 + str\_len  
  
 dash\_printer(line\_len)  
  
 PRINT "| {title} |"  
  
 dash\_printer(line\_len)  
END  
  
/\* void dash\_printer(int num) \*/  
dash\_printer(int num)  
 DO i = 0 TO num  
 PRINT "-"  
 ENDDO  
 PRINT "\n"  
END  
  
/\* int user\_code\_parser(char \*user\_code) \*/  
user\_code\_parser(char \*user\_code)  
 SET valid\_user\_id = 0  
  
 IF user\_code[0] = 'T' && user\_code[1] = 'P' OR user\_code[0] = 't' && user\_code[1] = 'p' ||  
 user\_code[0] = 't' && user\_code[1] = 'P' OR user\_code[0] = 'T' && user\_code[1] = 'p' THEN  
 valid\_user\_id = 1  
 DO i = 0 TO LEN user\_code  
 IF user\_code[i] < '0' OR user\_code[i] > '9' THEN  
 user\_code[i] = user\_code[i + 2]  
 ENDIF  
 ENDDO  
 ELSE IF user\_code[0] = 'T' && user\_code[1] = 'U' OR user\_code[0] = 't' && user\_code[1] = 'u' ||  
 user\_code[0] = 't' && user\_code[1] = 'U' OR user\_code[0] = 'T' && user\_code[1] = 'u' THEN  
 DO i = 0 TO LEN user\_code  
 IF user\_code[i] < '0' OR user\_code[i] > '9' THEN  
 user\_code[i] = user\_code[i + 2]  
 ENDIF  
 ENDDO  
 ELSE IF user\_code[0] = 'A' && user\_code[1] = 'D' OR user\_code[0] = 'a' && user\_code[1] = 'd' ||  
 user\_code[0] = 'a' && user\_code[1] = 'D' OR user\_code[0] = 'A' && user\_code[1] = 'd' THEN  
 DO i = 0 TO LEN user\_code  
 IF user\_code[i] < '0' OR user\_code[i] > '9' THEN  
 user\_code[i] = user\_code[i + 2]  
 ENDIF  
 ENDDO  
 ENDIF  
  
 return valid\_user\_id  
END  
  
/\* sessions get\_session(char \*filter\_field, char \*filter\_value) \*/  
get\_session(char \*filter\_field, char \*filter\_value)  
 SET s FROM STRUCT sessions  
 SET buffer  
 SET fp = FILE OPEN sessions\_file WITH "r"  
  
 IF fp = NULL THEN  
 PRINT "[SERVER WARNING] Failed to open file sessions.txt."  
 RETURN s  
 ENDIF  
  
 DOWHILE NOT fgets(buffer, sizeof(buffer), fp) != NULL  
 SET session\_code\_field = SPLIT ";" FROM buffer  
 SET title\_field = SPLIT ";" FROM buffer  
 SET day\_field = SPLIT ";" FROM buffer  
 SET start\_time\_field = SPLIT ";" FROM buffer  
 SET location\_field = SPLIT ";" FROM buffer  
 SET tutor\_code\_field = SPLIT ";" FROM buffer  
 IF filter\_field = "session\_code" AND session\_code\_field = filter\_value THEN  
 SET S.session\_code = session\_code\_field  
 SET S.title = title\_field  
 SET S.day = day\_field  
 SET S.start\_time = start\_time\_field  
 SET S.location = location\_field  
 SET S.tutor\_code = tutor\_code\_field  
  
 FILE CLOSE fp  
 RETURN s  
  
 ELSE IF filter\_field = "tutor\_code" AND tutor\_code\_field = filter\_value  
 SET S.session\_code = session\_code\_field  
 SET S.title = title\_field  
 SET S.day = day\_field  
 SET S.start\_time = start\_time\_field  
 SET S.location = location\_field  
 SET S.tutor\_code = tutor\_code\_field  
  
 FILE CLOSE fp  
 RETURN s  
 ENDIF  
 ENDDO  
  
 FILE CLOSE fp  
  
 RETURN s  
END  
  
/\* enrolled\_sessions \*get\_enrolled\_sessions(char \*filter\_field, char \*filter\_value, int \*num\_sessions) \*/  
get\_enrolled\_sessions(char \*filter\_field, char \*filter\_value, int \*num\_sessions)  
 SET sessions FROM STRUCT enrolled\_sessions = NULL  
 SET BUFFER  
  
 SET fp = FILE OPEN enrolled\_sessions\_file WITH "r"  
 IF fp = NULL THEN  
 PRINT "[SERVER WARNING] Failed to open file enrolled\_sessions.txt."  
 RETURN sessions  
 ENDIF  
  
 DOWHILE NOT fgets(buffer, sizeof(buffer), fp) = NULL  
 SET session\_code\_field = SPLIT ";" FROM buffer  
 SET user\_id\_field = SPLIT ";" FROM buffer  
 SET name\_field = SPLIT ";" FROM buffer  
 SET role\_field = SPLIT ";" FROM buffer  
  
 IF filter\_field = "session\_code" AND session\_code\_field = filter\_value THEN  
 count = count + 1  
 ELSE IF filter\_field = "user\_id" AND user\_id\_field = filter\_value  
 count = count + 1  
 ENDIF  
 ENDDO  
  
 SET sessions = ALLOCATE count \* sizeof(enrolled\_sessions)  
 IF sessions = NULL THEN  
 PRINT "[SERVER WARNING] Failed to allocate memory for sessions array."  
 RETURN sessions  
 ENDIF  
  
 REWIND fp  
  
 SET i = 0  
  
 DOWHILE NOT fgets(buffer, sizeof(buffer), fp) = NULL  
 SET session\_code\_field = SPLIT ";" FROM buffer  
 SET user\_id\_field = SPLIT ";" FROM buffer  
 SET name\_field = SPLIT ";" FROM buffer  
 SET role\_field = SPLIT ";" FROM buffer  
  
 IF filter\_field = "session\_code" AND session\_code\_field = filter\_value THEN  
 SET sessions[i].session\_code = session\_code\_field  
 SET sessions[i].user\_id = user\_id\_field  
 SET sessions[i].name = name\_field  
 SET sessions[i].role = role\_field  
 i = i + 1  
 ELSE IF filter\_field = "user\_id" AND user\_id\_field = filter\_value  
 SET sessions[i].session\_code = session\_code\_field  
 SET sessions[i].user\_id = user\_id\_field  
 SET sessions[i].name = name\_field  
 SET sessions[i].role = role\_field  
 i = i + 1  
 ENDIF  
 ENDDO  
  
 NUM\_SESSIONS = count  
  
 FILE CLOSE fp  
  
 RETURN sessions  
END  
  
/\* users get\_user(char \*user\_id) \*/  
get\_user(char \*user\_id)  
 SET u FROM STRUCT users  
 SET buffer  
 SET fp = FILE OPEN users\_file WITH "r"  
  
 IF fp = NULL THEN  
 PRINT "[SERVER WARNING] Failed to open file users.txt."  
 RETURN u  
 ENDIF  
  
 DOWHILE NOT fgets(buffer, sizeof(buffer), fp) != NULL  
 SET user\_id\_field = SPLIT ";" FROM buffer  
 SET name\_field = SPLIT ";" FROM buffer  
 SET password\_field = SPLIT ";" FROM buffer  
 SET email\_field = SPLIT ";" FROM buffer  
 SET role\_field = SPLIT ";" FROM buffer  
  
  
 IF user\_id\_field = user\_id THEN  
 SET U.user\_id = user\_id\_field  
 SET U.name = name\_field  
 SET U.password = password\_field  
 SET U.email = email\_field  
 SET U.role = role\_field  
  
 FILE CLOSE fp  
 RETURN s  
 ENDIF  
 ENDDO  
  
 FILE CLOSE fp  
  
 RETURN u  
END  
  
/\* student\_profiles get\_student\_profile(char \*user\_id) \*/  
get\_student\_profile(char \*user\_id)  
 SET sp FROM STRUCT student\_profiles  
 SET buffer  
 SET fp = FILE OPEN student\_profiles\_file WITH "r"  
  
 IF fp = NULL THEN  
 PRINT "[SERVER WARNING] Failed to open file student\_profiles.txt."  
 RETURN sp  
 ENDIF  
  
 DOWHILE NOT fgets(buffer, sizeof(buffer), fp) != NULL  
 SET user\_id\_field = SPLIT ";" FROM buffer  
 SET student\_code\_field = SPLIT ";" FROM buffer  
  
  
  
 IF user\_id\_field = user\_id THEN  
 SET SP.user\_id = user\_id\_field  
 SET SP.student\_code = student\_code\_field  
  
 FILE CLOSE fp  
 RETURN sp  
 ENDIF  
 ENDDO  
  
 FILE CLOSE fp  
  
 RETURN sp  
END  
  
/\* tutor\_profiles get\_tutor\_profile(char \*user\_id) \*/  
get\_tutor\_profile(char \*user\_id)  
 SET tp FROM STRUCT tutor\_profiles  
 SET buffer  
 SET fp = FILE OPEN tutor\_profiles\_file WITH "r"  
  
 IF fp = NULL THEN  
 PRINT "[SERVER WARNING] Failed to open file tutor\_profiles.txt."  
 RETURN tp  
 ENDIF  
  
 DOWHILE NOT fgets(buffer, sizeof(buffer), fp) != NULL  
 SET user\_id\_field = SPLIT ";" FROM buffer  
 SET tutor\_code\_field = SPLIT ";" FROM buffer  
 SET title\_field = SPLIT ";" FROM buffer  
  
  
 IF user\_id\_field = user\_id THEN  
 SET TP.user\_id = user\_id\_field  
 SET TP.tutor\_code = tutor\_code\_field  
 SET TP.title = title\_field  
  
 FILE CLOSE fp  
 RETURN tp  
 ENDIF  
 ENDDO  
  
 FILE CLOSE fp  
  
 RETURN tp  
END

# **3.0 Additional features**

*(Operation) Delete User*

*A picture containing text, screenshot, software, multimedia software

Description automatically generated*

***Figure 1*** *Algorithm of deleting a user.*

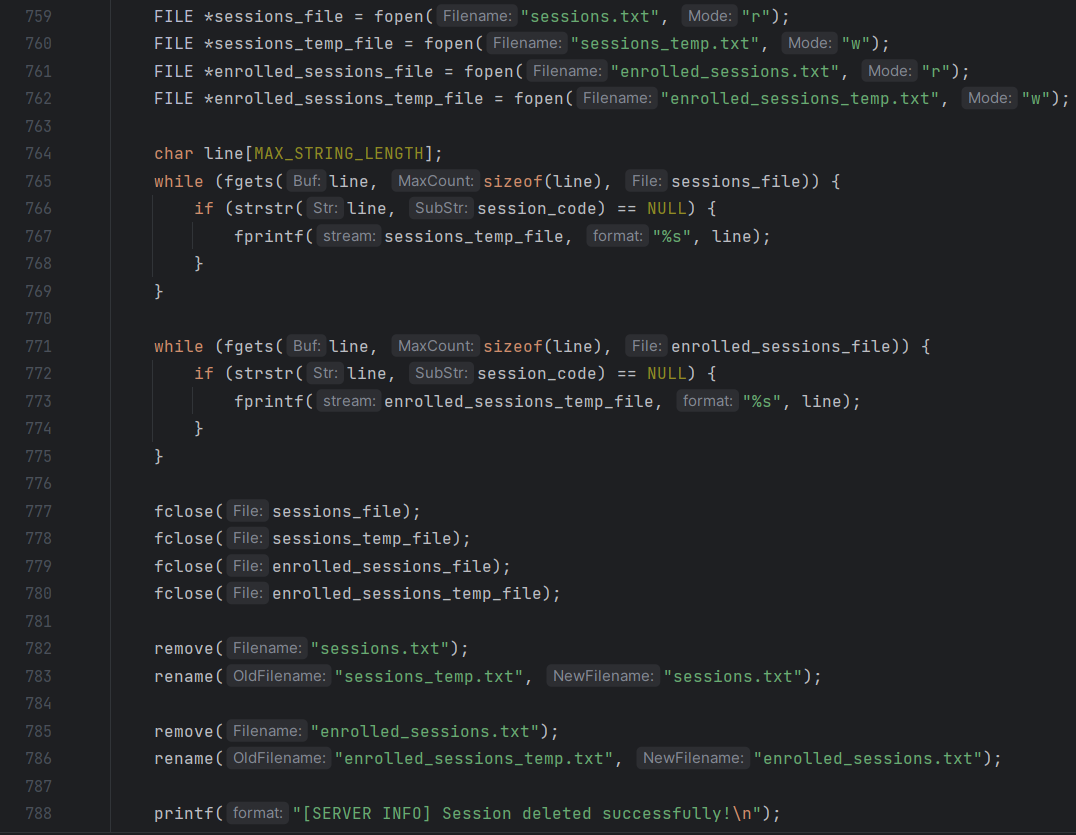
Once a user logs in as an **admin**, they are granted the privilege to delete any existing user from the system. To initiate the delete operation, the admin will be prompted to enter a user code, following which the system will proceed with the deletion process. This operation involves removing the user's **details, profile, and enrolled session records**.

The algorithm employed for the delete operation follows these steps:

1. Create a temporary file to hold non-user related content.
2. Read the existing file line by line.
3. Check each line to identify the user code entered by the admin.
4. Exclude any lines that pertain to the user being deleted.
5. Write the remaining non-user related lines to the temporary file.
6. Close both the existing file and the temporary file.
7. Delete the current file.
8. Rename the temporary file to the original file name.
9. Repeat the above process for other **txt file.**

By utilizing this algorithm, the system efficiently removes all information associated with the user being deleted while preserving the integrity of the remaining data.

*(Operation) Delete Session*

**

***Figure 2*** *Algorithm of deleting a session.*

Deleting a session applies the same algorithm with delete user, but also delete all the record in enrolled session.

*(Operation) Disenroll a user*

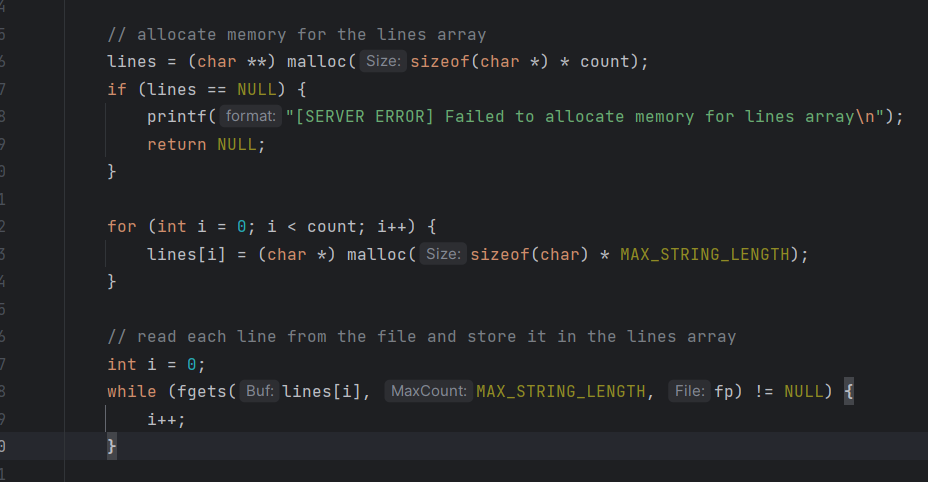
*A screenshot of a computer screen

Description automatically generated with low confidence*

***Figure 3*** *Algorithm of deleting a session.*

Dis-enrol a user applies the same algorithm with delete user.

*(API) Read*

**

The "read()" function plays a crucial role in the system by reading all the lines in a file and returning a dynamic 2D array that adjusts its size based on the number of lines and their respective lengths. This dynamic array provides a flexible structure for storing and accessing the file's contents.

Upon execution, the "read()" function scans the file and determines the number of lines present. It then creates a dynamic 2D array with the appropriate dimensions to accommodate the data. This dynamic array allows for efficient memory allocation and ensures that the array size matches the file's content, preventing any unnecessary memory wastage.

By utilizing the number count of lines, the function enables a systematic iteration over the dynamic array. This facilitates seamless data processing and allows for a wide range of actions to be performed on the retrieved information. Users can efficiently loop over the dynamic array using the line count as an indicator to access and manipulate each line of data.

*(API) Get Session*

A picture containing text, screenshot, software, multimedia software

Description automatically generated

A screenshot of a computer program

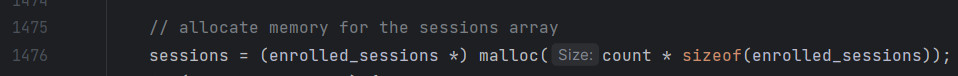
Description automatically generated with medium confidence

The "get\_session()" function serves the purpose of retrieving session information from the system. It offers the flexibility to apply various filters by accepting different arguments. By passing in either the tutor\_code or session\_code, users can specify the desired criteria for data retrieval. The function then processes the request and returns a pre-defined data structure containing the relevant session information.

*(API) Get Enrolled Session*

*A screenshot of a computer program

Description automatically generated with low confidence*

**

*A screen shot of a computer program

Description automatically generated with medium confidence*

The get\_enrolled\_session() allow user to quickly search any record in **enrolled\_sessions.txt** by passing in either the tutor\_code or session\_code, users can specify the desired criteria for data retrieval. The function will then return a data structure array containing the relevant information.

*(API) Get User*

**Refer to *(API) Get Session***

*(API) Get Student Profile & (API)*

**Refer to *(API) Get Session***

*Get Tutor Profile*

**Refer to *(API) Get Session***

# **4.0 Sample outputs**

## **4.1 Home menu**

A screenshot of a computer program

Description automatically generated with medium confidence

***Figure 1*** *Home menu (UI)*

A screen shot of a computer program

Description automatically generated with medium confidence

***Figure 2*** *Home menu (Validation) – Check if the input is integer*

A screenshot of a computer program

Description automatically generated with medium confidence

***Figure 3*** *Home menu (Validation) – Check for correct option.*

### **4.1.1 Available sessions menu**

A screenshot of a computer program

Description automatically generated with medium confidence

***Figure 4*** *Available sessions menu (UI)*

### **4.1.2 Login menu**

A screenshot of a computer program

Description automatically generated with medium confidence

***Figure 5*** *Login menu (UI)*

A screenshot of a computer screen

Description automatically generated with medium confidence

***Figure 6*** *Login menu (Validation) – User has only 3 attempts to login.*

## **4.2.0 Admin menu**

A screenshot of a computer program

Description automatically generated with medium confidence

***Figure 7*** *Admin menu (UI)*

### **4.2.1 User operation**

*A screenshot of a computer program

Description automatically generated with medium confidence*

***Figure 8*** *User operation (UI)*

A screen shot of a computer

Description automatically generated with medium confidence

***Figure 9*** *User operation (Validation) – Check if the input is an integer.*

*A screenshot of a computer program

Description automatically generated with low confidence*

***Figure 10*** *User operation (Validation) – Check if the input is an integer.*

### **4.2.1.1 Add user**

#### **A screenshot of a computer Description automatically generated with medium confidence**

***Figure 11.1*** *Add new user (UI) – Added a new student.*

*A screenshot of a computer program

Description automatically generated with medium confidence*

***Figure 11.2*** *Add new user (UI) – Added a new student.*

#### **A screenshot of a computer program Description automatically generated with medium confidence**

***Figure 12.1*** *Add new user (UI) – Added a new tutor.*

*A screen shot of a computer

Description automatically generated with medium confidence.*

***Figure 12.2*** *Add new user (UI) – Added a new tutor.*

#### **A screenshot of a computer program Description automatically generated with medium confidence**

***Figure 13*** *Add new user (Validation) – Check if the user correct role.*

*A black screen with white text

Description automatically generated with low confidence*

***Figure 14*** *Add new user (Validation) – Check if the user id is exactly 6 characters.*

*A black background with white text

Description automatically generated with low confidence*

***Figure 15*** *Add new user (Validation) – Check if the password is more than 8 characters.*

### **4.2.1.2 Delete user**

A screenshot of a computer screen

Description automatically generated with medium confidence

***Figure 16*** *Delete user (UI)*

A screenshot of a computer program

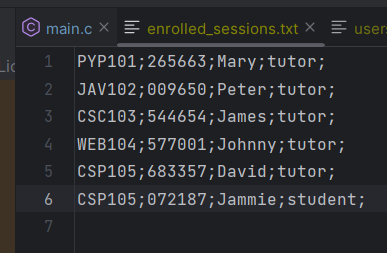
Description automatically generated with medium confidence

***Figure 17.1*** *Delete user (Data preview) – Before*

*A screenshot of a computer

Description automatically generated with medium confidence*

***Figure 17.2*** *Delete user (Data preview) – After*

**

***Figure 17.3*** *Delete user (Data preview) – Before deleting the enrolled user.*

*A screenshot of a computer

Description automatically generated with medium confidence*

***Figure 17.4*** *Delete user (Data preview) – After deleting the enrolled user.*

#### **A screenshot of a computer screen Description automatically generated with low confidence**

***Figure 18*** *Delete user (Validation) – Check if the user exists.*

#### **A screenshot of a computer Description automatically generated with medium confidence**

***Figure 19*** *Delete user (Validation) – Check if the user code is valid.*

### **4.2.1.3 View all users**

A screenshot of a computer

Description automatically generated

***Figure 20*** *View all user (UI)*

### **4.2.1.4 View user profile**

*A screenshot of a computer

Description automatically generated*

***Figure 21*** *View user profile (UI)*

#### **A screenshot of a computer screen Description automatically generated with low confidence**

***Figure 22*** *View user profile (Validation) – Check if the user exists.*

*A screenshot of a computer program

Description automatically generated with low confidence*

***Figure 23*** *View user profile (Validation) – Check if the user code is valid.*

### **4.2.2 Session operation**

#### **A screenshot of a computer program Description automatically generated with medium confidence**

***Figure 24*** *Session operation (UI)*

*A screenshot of a computer program

Description automatically generated with medium confidence*

***Figure 25*** *Session operation (Validation) – Check if the input is an integer.*

*A screenshot of a computer program

Description automatically generated with low confidence*

***Figure 26*** *Session operation (Validation) – Check if the input is valid.*

### **4.2.2.1 Add Session**

A screenshot of a computer program

Description automatically generated with medium confidence

***Figure 27*** *Add session (UI) – Before.*

A screenshot of a computer

Description automatically generated with medium confidence

***Figure 28.1*** *Add session (Data preview) – Before.*

*A screenshot of a computer program

Description automatically generated with medium confidence*

***Figure 28.2*** *Add session (Data preview) – After.*

A screenshot of a computer program

Description automatically generated with low confidence

***Figure 29*** *Add session (Validation) – Check if the session code is valid.*

*A screen shot of a computer program

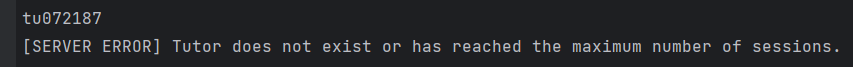
Description automatically generated with low confidence*

***Figure 30*** *Add session (Validation)– Check if the session code exists.*

*A black background with white text

Description automatically generated with low confidence*

***Figure 31*** *Add session (Validation)– Check if the tutor is enrolled in other sessions.*

**

***Figure 32*** *Add session (Validation)– Check if the tutor is enrolled in other sessions.*

### **4.2.2.2 Delete session**

A screenshot of a computer program

Description automatically generated with low confidence

***Figure 33*** *Delete session (UI)*

A screenshot of a computer program

Description automatically generated with low confidence

***Figure 34.1*** *Delete session (Data preview) – Before.*

*A screenshot of a computer program

Description automatically generated with medium confidence*

***Figure 34.2*** *Delete session (Data preview) – After.*

*A screen shot of a computer

Description automatically generated with low confidence*

***Figure 35*** *Delete session (Validation) – Check if session exists.*

### **4.2.2.3 View Session**

A screenshot of a computer program

Description automatically generated with medium confidence

***Figure 36*** *View Session (UI)*

*A screen shot of a computer

Description automatically generated with low confidence*

***Figure 37*** *View Session (Validation) – Check if the session code exists.*

### **4.2.2.4 Enrol User**

A screen shot of a computer

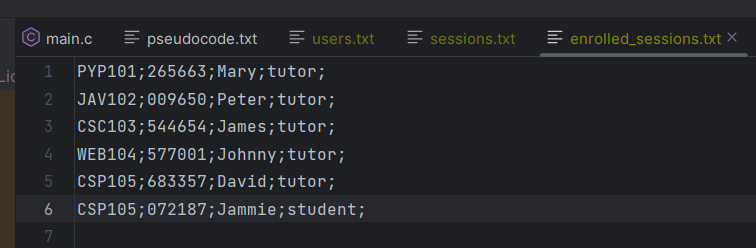
Description automatically generated with low confidence

***Figure 38*** *Enroll User (UI)*

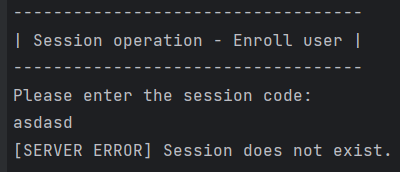
A screen shot of a computer

Description automatically generated with medium confidence

***Figure 39*** *Enroll User (Data preview) – Before.*



***Figure 39.1*** *Enroll User (Data preview) – After*

**

***Figure 40*** *View Session (Validation) – Check if the session exists.*

*A screen shot of a computer error

Description automatically generated with medium confidence*

***Figure 41*** *View Session (Validation) – Check if the user exists.*

*A screen shot of a computer program

Description automatically generated with low confidence*

***Figure 42*** *View Session (Validation) – Tutor is not allowed to enroll here.*

*A screenshot of a computer program

Description automatically generated with low confidence*

***Figure 43*** *View Session (Validation) – Check if a user enrolled into same session again.*

### **4.2.2.4 Dis-enroll User**

A screenshot of a computer screen

Description automatically generated with low confidence

***Figure 44*** *Dis-enroll User (UI)*

A screen shot of a computer

Description automatically generated with medium confidence

***Figure 44.1*** *Dis-enroll User (Data preview)*

*A screenshot of a computer program

Description automatically generated with medium confidence*

***Figure 44.2*** *Dis-enroll User (Data preview)*

*A screen shot of a computer

Description automatically generated with low confidence*

***Figure 45*** *Dis-enroll User (Validation) – Check if the session code is valid.*

*A screenshot of a computer program

Description automatically generated with low confidence*

***Figure 46*** *Dis-enroll User (Validation) – check if the user code is valid.*

*A screenshot of a computer program

Description automatically generated with low confidence*

***Figure 46*** *Dis-enroll User (Validation) – User cannot be disenrolled here, it’s in delete session menu.*

## **4.3.0 Tutor menu**

*A screenshot of a computer program

Description automatically generated with low confidence.*

***Figure 47*** *Tutor menu (UI)*

A screenshot of a computer program

Description automatically generated with medium confidence

***Figure 48*** *Tutor menu (Validation) – Check if the input is an integer.*

*A screen shot of a computer

Description automatically generated with medium confidence*

***Figure 49*** *Tutor menu (Validation) – Check if the input is valid.*

### **4.3.1 View my session.**

A screenshot of a computer

Description automatically generated with medium confidence

***Figure 50*** *View my session (UI)*

### **4.3.1 View student enrolled in my session.**

A screenshot of a computer program

Description automatically generated with low confidence

***Figure 51*** *View student enrolled in my session (UI)*

## **4.4.0 Student menu**

A screen shot of a computer program

Description automatically generated with low confidence

***Figure 52*** *Student Menu (UI)*

*A screen shot of a computer

Description automatically generated with medium confidence*

***Figure 53*** *Student Menu (Validation) – Check if the input is an integer.*

## **4.4.1 View my session.**

*A screenshot of a computer

Description automatically generated with medium confidence*

***Figure 54*** View my session *(UI)*

## **4.4.2 Enroll into session.**

A screen shot of a computer program

Description automatically generated with low confidence

***Figure 55*** *Enroll into session (UI)*

*A screenshot of a computer program

Description automatically generated with medium confidence*

***Figure 56.1*** *Enroll into session (Data preview) – Before.*

*A screenshot of a computer program

Description automatically generated with medium confidence*

***Figure 56.2*** *Enroll into session (Data preview) – After.*

*A screenshot of a computer program

Description automatically generated with medium confidence*

***Figure 56*** *Enroll into session (Validation) – Check if the user has already enrolled into the same session.*

# **5.0 Conclusion**

In conclusion, the development of the "Programming Café" system at APU has proven to be a valuable asset in enhancing students' coding skills, problem-solving abilities, and confidence. By offering additional coding sessions outside of students' regular timetable, the system provides a platform for students to further develop their programming knowledge.

**6.0 References using APA Referencing**

*C Files I/O: Opening, Reading, Writing and Closing a file*. (n.d.). https://www.programiz.com/c-programming/c-file-input-output