Intake: APD1F2409CS(CYB) **Module Code: CT108-3-1-PYP**

Hand Out Date: Monday, 21 October 2024 Week 3
Submission Date: Sunday, 19 January 2024 Week 12

TITLE:

UNIVERSITY MANAGEMENT SYSTEM (UMS)

Module Lecturer: Mrs. Aziah Abdollah

Group ZPYP

No.	Name	TP
1	AHMED MIRAHUSAIN ALVI	TP084807
2	ANDREW WIJAYA	TP079319
3	MAHRUS SHAMSUL AHSAN	TP085562
4	ABDALLAH MOHAMED MAHMOUD MOHAMED MAHMOUD	TP085097
5	MOHAMMED YOUSEF MOHAMMED MOHAMMED	TP085042

Table of Contents

ACKNOWLEDGEMENT

We extend our sincere gratitude to everyone who played a role in the successful completion of this document. Above all, we are deeply indebted to our mentor, Ms. Aziah Abdollah, for her invaluable guidance, insightful feedback and constant encouragement, which have been instrumental in shaping the quality and direction of this work.

We also would like to express our sincere appreciation to our colleagues and peers. Their spirit of collaboration, constructive discussions and shared knowledge have greatly enriched our understanding and perspective.

A special note of thanks goes to our friends—Mohammed Yousef Mohammed Mohammed, Mahrus Shamsul Ahsan, Ahmed Mirahusain Alvi, Andrew Wijaya, and Abdallah Mohamed Mahmoud—whose steadfast support, motivation, and camaraderie have been a constant source of inspiration throughout this journey.

Finally, we acknowledge the importance of the resources that have supported this work, including books, research articles, and online materials. These resources have been invaluable in providing the necessary foundation and depth to our efforts.

To everyone who encouraged and believed in us, we owe our success to your kindness and support. Thank you for making this accomplishment a reality.

ABSTRACT

This paper explores the development and implementation university management system. The system features 5 primary roles in the system including administrator, registrar, accountant, lecturer, and student, each playing an important role in the system. The report has been structured into 4 sections, providing detailed explanation of the design, implementation and usage of the system.

Section 1introduces the project, outlining all the assumptions made in order to create this system. These assumptions laid the foundation of the system. section 2 includes the pseudocode needed to create the system. The pseudocode has been further divided to showcase code design for each role. Section 3 talks about the source code and its explanation based on its functionality. The codes are explained according to certain criteria provided. Section 5 includes all the input, and output for each role. In this section, it is shown what happens when an user uses the system. It shows the results when the user enters valid and invalid input. The report then ends with the conclusion, references and appendix.

CHAPTER 1: UNIVERSITY MANAGEMENT SYSTEM

1.1 *Introduction*

The University had an ineffective system to manage data and procedures for the five users: Students, lecturer, administrators, Accountant, Registrar. This system caused many problems. For example, it was easy to cause human errors. While operating the university menu, the users could make mistakes like choosing the wrong option. However, with an automatic system, equipped with validations, recorded data, User Friendly, the university can easily do its things with rapid, and less to make errors. Moreover, installing an automated system can be expensive due to the first investment in equipment, it could improve efficiency and reducing the staff needed to do the thing, reducing cost by time to time. Therefore, compared to the manual system, the automated one is better for the smooth running of the university, and it would also give it a modern touch.

Our team was tasked to improve system to become more efficient system. Processes like manage lecturer, register student, the fee that has been paid by student, record student grade and others would be automated. Validations and other functionalities are implemented in the system, to ensure more effective management and boost overall efficiency and accuracy.

1.2 Assumptions

- The goal for this project is to make an automatic system which is to support the five primary users which are student, Lecturer, Administrator, Registrar, Accountant Each will have their respective menu page.

- There is a login feature that was needed to access each role function, in which the Username is “UMSLOGIN,” and the password is “12589”. This had been made to ensure the security of the program .

- Some of the things that administrators can do with the system are change View All Data, add or remove students, add classes, control teachers, and make reports. What registrars do, on the other hand, is manage enrollments, add new students, keep student records up to date, send out papers, keep an eye on student information, and delete student records.

- What do lecturers do for school? They must look at the modules they have been given, write down grades, look at names of students, keep track of attendance, and go over student scores. Students can also see what classes are available, choose to take or not take, see a record of their attendance, and see their grades. Last but not least, accountants handle money problems. It is their job to keep track of payments, print fee receipts, record school fees, and look at payments that are still due.

- Error management: When managing errors, try-except blocks are used to make sure that the program can know what to do if it faces the incorrect or missing input / data without crashing. The system is strengthened and stabilized this way to avoid redundant.

- We also implemented a modular program in which all roles are in different files, different save data which are all in text file and compiled through the one folder which all can be accessed through the UMS main logic menu.

- There are eight files being used which are “Security.txt” for login, “attendance.txt” for record student present data, “fee.txt” to record how much had the student paid, “Grades.txt” for record student score in test, ”Lecturer.txt” to record the lecturer info and show the lecturer info, “Modules.txt” to show the module data, “Student.txt” to record the student info and show the student info”

CHAPTER 2: PROGRAM DESIGN & PLANNING

2.1 *UMS Main Menu*

```
BEGIN UniversityManagementSystem

    FUNCTION get_valid_input(prompt, required=True)
        WHILE TRUE DO
            DISPLAY prompt
            GET value
            SET value TO value.strip()
            IF value IS NOT EMPTY OR required IS FALSE THEN
                RETURN value
            ELSE
                DISPLAY "This field cannot be empty. Please try again."
            ENDIF
        ENDWHILE
    ENDFUNCTION

    FUNCTION login()
        DISPLAY "Please enter Login credentials"
        DISPLAY 'UserName: "UMSLOGIN", Password: "12589"'

        PROMPT "Enter Username: "
        GET username USING get_valid_input(prompt)

        PROMPT "Enter password: "
```

```
    GET password USING get_valid_input(prompt)

    TRY
      OPEN "security.txt" FOR READ AS file
      READ file LINE BY LINE INTO users

      FOR EACH user IN users DO
        SPLIT user BY "," INTO stored_name AND stored_password
        IF username EQUALS stored_name AND password EQUALS
stored_password THEN
          DISPLAY "Login successful!"
          RETURN TRUE
        ENDIF
      ENDFOR

      DISPLAY "Invalid username or password. Try again."
      RETURN FALSE

    CATCH FileNotFoundError EXCEPTION
      DISPLAY "Security file not found."
      RETURN FALSE
    ENDTRY
  ENDFUNCTION

  FUNCTION main_menu()
    WHILE TRUE DO
      DISPLAY "-------Welcome to University Management System----------"
```

```
SET login_result TO CALL login()
IF login_result IS FALSE THEN
    DISPLAY "Access Denied! Please try logging in again."
    CONTINUE
ENDIF

DISPLAY "\n---------- Main Menu ----------"
DISPLAY "1. Administrator"
DISPLAY "2. Lecturer"
DISPLAY "3. Student"
DISPLAY "4. Registrar"
DISPLAY "5. Accountant"
DISPLAY "6. Exit"

PROMPT "Enter your choice: "
GET choice

IF choice EQUALS "1" THEN
    CALL Administrator.admin_menu()
ELSE IF choice EQUALS "2" THEN
    CALL Lecturer.lecturer_menu()
ELSE IF choice EQUALS "3" THEN
    CALL Student.student_menu()
ELSE IF choice EQUALS "4" THEN
    CALL Registrar.registrar_menu()
ELSE IF choice EQUALS "5" THEN
    CALL Accountant.accountant_menu()
```

```
        ELSE IF choice EQUALS "6" THEN
            DISPLAY "Exiting now..."
            BREAK
        ELSE
            DISPLAY "Invalid choice, please try again."
        ENDIF
    ENDWHILE
ENDFUNCTION

CALL main_menu()

END UniversityManagementSystem
```

***2.2** Administrator Role*

```
BEGIN AdministratorMenu

  FUNCTION admin_menu()
    FUNCTION validate_numeric_input(prompt)
      WHILE TRUE DO
        DISPLAY prompt
        GET value
        SET value TO value.strip()
        IF value IS NUMERIC THEN
          RETURN value
        ELSE
          DISPLAY "Invalid input. Please enter a numeric value."
        ENDIF
      ENDWHILE
    ENDFUNCTION

    FUNCTION validate_date_input(prompt)
      WHILE TRUE DO
        DISPLAY prompt
        GET date_str
        TRY
          PARSE date_str AS DATE (FORMAT: YYYY-MM-DD)
          RETURN date_str
        CATCH ValueError EXCEPTION
          DISPLAY "Invalid date format. Please use YYYY-MM-DD."
        ENDTRY
```

```
    ENDWHILE
ENDFUNCTION

FUNCTION manage_module()
    DISPLAY "\nModule Management"
    PROMPT "Choose an option (Add, Update, Remove): "
    GET option
    SET option TO option.lower()

    TRY
        IF option EQUALS "add" THEN
            PROMPT "Enter module code: "
            GET module_code
            PROMPT "Enter module name: "
            GET module_name
            PROMPT "Enter module credits: "
            GET credits USING validate_numeric_input(prompt)

            APPEND (module_code, module_name, credits) TO "modules.txt"
            DISPLAY "Module added successfully."

        ELSE IF option EQUALS "update" THEN
            PROMPT "Enter module code to update: "
            GET module_code
            SET found TO FALSE

            READ ALL LINES FROM "modules.txt" INTO lines
```

```
    OPEN "modules.txt" FOR WRITE

    FOR EACH line IN lines DO
        IF line STARTS WITH (module_code + ",") THEN
            PROMPT "Enter new module name: "
            GET module_name
            PROMPT "Enter new course credits: "
            GET credits USING validate_numeric_input(prompt)
            WRITE (module_code, module_name, credits) TO FILE
            SET found TO TRUE
        ELSE
            WRITE line TO FILE
        ENDIF
    ENDFOR

    IF found THEN
        DISPLAY "Module updated successfully."
    ELSE
        DISPLAY "Module not found."
    ENDIF

ELSE IF option EQUALS "remove" THEN
    PROMPT "Enter a module code to remove: "
    GET module_code
    SET found TO FALSE

    READ ALL LINES FROM "modules.txt" INTO lines
```

```
            OPEN "modules.txt" FOR WRITE

            FOR EACH line IN lines DO
                IF NOT line STARTS WITH (module_code + ",") THEN
                    WRITE line TO FILE
                ELSE
                    SET found TO TRUE
                ENDIF
            ENDFOR

            IF found THEN
                DISPLAY "Module removed successfully."
            ELSE
                DISPLAY "Module not found."
            ENDIF

        ELSE
            DISPLAY "Invalid option. Please try again."
        ENDIF
    CATCH FileNotFoundError EXCEPTION
        DISPLAY "Error: File not found. Ensure 'modules.txt' exists."
    ENDTRY
ENDFUNCTION

FUNCTION manage_student()
    DISPLAY "\nStudent Management"
    PROMPT "Choose an option (Add, Update, Remove): "
```

```
GET option
SET option TO option.lower()

TRY
   IF option EQUALS "add" THEN
      PROMPT "Enter student ID: "
      GET student_id USING validate_numeric_input(prompt)
      PROMPT "Enter student name: "
      GET student_name
      PROMPT "Enter department: "
      GET department
      PROMPT "Enter Date of Admission [YYYY-MM-DD]: "
      GET AdmissionDate USING validate_date_input(prompt)

      APPEND (student_id, student_name, department, AdmissionDate) TO
"students.txt"
      DISPLAY "Student added successfully."

   ELSE IF option EQUALS "update" THEN
      PROMPT "Enter student ID to update: "
      GET student_id USING validate_numeric_input(prompt)
      SET found TO FALSE

      READ ALL LINES FROM "students.txt" INTO lines
      OPEN "students.txt" FOR WRITE

      FOR EACH line IN lines DO
         IF line STARTS WITH (student_id + ",") THEN
```

```
            PROMPT "Enter new student name: "
            GET student_name
            PROMPT "Enter new department: "
            GET department
            WRITE (student_id, student_name, department) TO FILE
            SET found TO TRUE
        ELSE
            WRITE line TO FILE
        ENDIF
    ENDFOR

    IF found THEN
        DISPLAY "Student updated successfully."
    ELSE
        DISPLAY "Student not found."
    ENDIF

ELSE IF option EQUALS "remove" THEN
    PROMPT "Enter student ID to remove: "
    GET student_id USING validate_numeric_input(prompt)
    SET found TO FALSE

    READ ALL LINES FROM "students.txt" INTO lines
    OPEN "students.txt" FOR WRITE

    FOR EACH line IN lines DO
        IF NOT line STARTS WITH (student_id + ",") THEN
```

```
            WRITE line TO FILE
          ELSE
            SET found TO TRUE
          ENDIF
        ENDFOR

        IF found THEN
          DISPLAY "Student removed successfully."
        ELSE
          DISPLAY "Student not found."
        ENDIF

      ELSE
        DISPLAY "Invalid option. Please try again."
      ENDIF
    CATCH FileNotFoundError EXCEPTION
      DISPLAY "Error: File not found. Ensure 'students.txt' exists."
    ENDTRY
ENDFUNCTION

FUNCTION manage_lecturer()
    DISPLAY "\nLecturer Management"
    PROMPT "Choose an option (Add, Update, Remove): "
    GET option
    SET option TO option.lower()

    TRY
```

```
        IF option EQUALS "add" THEN
            PROMPT "Enter lecturer ID: "
            GET lecturer_id USING validate_numeric_input(prompt)
            PROMPT "Enter lecturer name: "
            GET lecturer_name
            PROMPT "Enter department: "
            GET department
            PROMPT "Enter Date of Registration [YYYY-MM-DD]: "
            GET HiringDate USING validate_date_input(prompt)

            APPEND (lecturer_id, lecturer_name, department, HiringDate) TO
"lecturers.txt"
            DISPLAY "Lecturer added successfully."

        ELSE IF option EQUALS "update" THEN
            PROMPT "Enter lecturer ID to update: "
            GET lecturer_id USING validate_numeric_input(prompt)
            SET found TO FALSE

            READ ALL LINES FROM "lecturers.txt" INTO lines
            OPEN "lecturers.txt" FOR WRITE

            FOR EACH line IN lines DO
                IF line STARTS WITH (lecturer_id + ",") THEN
                    PROMPT "Enter new lecturer name: "
                    GET lecturer_name
                    PROMPT "Enter new department: "
                    GET department
```

```
            WRITE (lecturer_id, lecturer_name, department) TO FILE
            SET found TO TRUE
        ELSE
            WRITE line TO FILE
        ENDIF
    ENDFOR

    IF found THEN
        DISPLAY "Lecturer updated successfully."
    ELSE
        DISPLAY "Lecturer not found."
    ENDIF

ELSE IF option EQUALS "remove" THEN
    PROMPT "Enter lecturer ID to remove: "
    GET lecturer_id USING validate_numeric_input(prompt)
    SET found TO FALSE

    READ ALL LINES FROM "lecturers.txt" INTO lines
    OPEN "lecturers.txt" FOR WRITE

    FOR EACH line IN lines DO
        IF NOT line STARTS WITH (lecturer_id + ",") THEN
            WRITE line TO FILE
        ELSE
            SET found TO TRUE
        ENDIF
```

```
            ENDFOR

            IF found THEN
               DISPLAY "Lecturer removed successfully."
            ELSE
               DISPLAY "Lecturer not found."
            ENDIF

         ELSE
            DISPLAY "Invalid option. Please try again."
         ENDIF
      CATCH FileNotFoundError EXCEPTION
         DISPLAY "Error: File not found. Ensure 'lecturers.txt' exists."
      ENDTRY
   ENDFUNCTION

   FUNCTION view_all_data()
      TRY
         DISPLAY "\nViewing All Data:"
         SET timestamp TO CURRENT DATE AND TIME (FORMAT: YYYY-MM-
DD HH:MM:SS)
         DISPLAY "Data viewed on: " + timestamp

         DISPLAY "\nAll Students Data:"
         READ ALL CONTENTS OF "students.txt" AND DISPLAY

         DISPLAY "\nAll Modules Data:"
         READ ALL CONTENTS OF "modules.txt" AND DISPLAY
```

```
            DISPLAY "\nAll Lecturers Data:"
            READ ALL CONTENTS OF "lecturers.txt" AND DISPLAY

        CATCH FileNotFoundError EXCEPTION
            DISPLAY "Error: One or more files are missing. Ensure all required files exist."
        ENDTRY
    ENDFUNCTION

    FUNCTION generate_reports()
        TRY
            DISPLAY "\nGenerating Reports:"
            SET student_count TO COUNT LINES IN "students.txt"
            SET module_count TO COUNT LINES IN "modules.txt"
            SET lecturer_count TO COUNT LINES IN "lecturers.txt"
            SET timestamp TO CURRENT DATE AND TIME (FORMAT: YYYY-MM-DD HH:MM:SS)

            DISPLAY "Reports generated on: " + timestamp
            DISPLAY "Total Students: " + student_count
            DISPLAY "Active Modules: " + module_count
            DISPLAY "Total Lecturers: " + lecturer_count

        CATCH FileNotFoundError EXCEPTION
            DISPLAY "Error: One or more files are missing. Ensure all required files exist."
        ENDTRY
```

```
ENDFUNCTION

WHILE TRUE DO
    DISPLAY "\n---------- University Administrator ----------"
    DISPLAY "1. Manage Modules"
    DISPLAY "2. Manage Students"
    DISPLAY "3. Manage Lecturers"
    DISPLAY "4. View All Data"
    DISPLAY "5. Generate Reports"
    DISPLAY "6. Exit"
    PROMPT "Enter your choice: "
    GET choice

    IF choice EQUALS "1" THEN
        CALL manage_module()
    ELSE IF choice EQUALS "2" THEN
        CALL manage_student()
    ELSE IF choice EQUALS "3" THEN
        CALL manage_lecturer()
    ELSE IF choice EQUALS "4" THEN
        CALL view_all_data()
    ELSE IF choice EQUALS "5" THEN
        CALL generate_reports()
    ELSE IF choice EQUALS "6" THEN
        DISPLAY "Exiting Administrator menu."
        BREAK
    ELSE
```

```
            DISPLAY "Invalid choice. Please try again."
        ENDIF
    ENDWHILE
ENDFUNCTION

END AdministratorMenu
```

2.3 *Lecturer Role*

```
BEGIN LecturerMenu


   FUNCTION lecturer_menu()


      FUNCTION view_assigned_modules(lecturer_id)
         DISPLAY "Viewing Assigned Modules..."
         SET timestamp TO CURRENT DATE AND TIME (FORMAT: YYYY-MM-DD
HH:MM:SS)
         DISPLAY "time-stamp: " + timestamp
         TRY
            OPEN "modules.txt" FOR READ
            DISPLAY "Modules assigned:"
            FOR EACH line IN file DO
               SPLIT line BY "," INTO details
               IF LENGTH OF details IS 4 AND details[3] EQUALS lecturer_id THEN
                  DISPLAY details[0] + " - " + details[1] + " (" + details[2] + "
credits)"
               ENDIF
            ENDFOR
         CATCH FileNotFoundError EXCEPTION
            DISPLAY "Modules file not found."
         ENDTRY
      ENDFUNCTION


      FUNCTION view_student_list(module_code)
         DISPLAY "Student List for Module: " + module_code
```

```
    SET timestamp TO CURRENT DATE AND TIME (FORMAT: YYYY-MM-DD HH:MM:SS)
    DISPLAY "time-stamp: " + timestamp
    TRY
        OPEN "enrolments.txt" FOR READ
        DISPLAY "Enrolled Students:"
        SET found TO FALSE
        FOR EACH line IN file DO
            SET line TO line.strip()
            IF line IS EMPTY THEN
                CONTINUE
            ENDIF
            TRY
                SPLIT line BY "," INTO student_id AND mod_code
                IF mod_code EQUALS module_code THEN
                    DISPLAY "Student ID: " + student_id
                    SET found TO TRUE
                ENDIF
            CATCH ValueError EXCEPTION
                DISPLAY "Skipping invalid line: " + line
            ENDTRY
        ENDFOR
        IF NOT found THEN
            DISPLAY "No students are enrolled in module " + module_code + "."
        ENDIF
    CATCH FileNotFoundError EXCEPTION
        DISPLAY "Enrollments file not found."
    ENDTRY
```

```
ENDFUNCTION

FUNCTION record_grades()
    DISPLAY "Recording Grades..."
    PROMPT "Enter Module Code: "
    GET module_code
    PROMPT "Enter Student ID: "
    GET student_id
    PROMPT "Enter Grade: "
    GET grade

    TRY
        OPEN "grades.txt" FOR APPEND
        WRITE (module_code + "," + student_id + "," + grade) TO file
        DISPLAY "Grade recorded successfully."
    CATCH FileNotFoundError EXCEPTION
        DISPLAY "Grades file not found."
    ENDTRY
ENDFUNCTION

FUNCTION track_attendance()
    DISPLAY "Tracking Attendance..."
    PROMPT "Enter Module Code: "
    GET module_code
    PROMPT "Enter Student ID: "
    GET student_id
    PROMPT "Enter Attendance Status (Present/Absent): "
```

```
    GET attendance_status

    TRY
        OPEN "attendance.txt" FOR APPEND
        WRITE (module_code + "," + student_id + "," + attendance_status) TO file
        DISPLAY "Attendance recorded successfully."
    CATCH FileNotFoundError EXCEPTION
        DISPLAY "Attendance file not found."
    ENDTRY
ENDFUNCTION

FUNCTION view_student_grades(module_code)
    DISPLAY "Viewing Grades for Module: " + module_code
    SET timestamp TO CURRENT DATE AND TIME (FORMAT: YYYY-MM-DD
HH:MM:SS)
    DISPLAY "time-stamp: " + timestamp
    TRY
        OPEN "grades.txt" FOR READ
        DISPLAY "Student Grades:"
        FOR EACH line IN file DO
            SPLIT line BY "," INTO mod_code, student_id, grade
            IF mod_code EQUALS module_code THEN
                DISPLAY "Student ID: " + student_id + ", Grade: " + grade
            ENDIF
        ENDFOR
    CATCH FileNotFoundError EXCEPTION
        DISPLAY "Grades file not found."
    ENDTRY
```

```
ENDFUNCTION

WHILE TRUE DO
    DISPLAY "---------- Lecturer Menu ----------"
    DISPLAY "1. View Assigned Modules"
    DISPLAY "2. View Student List for a Module"
    DISPLAY "3. Record Grades"
    DISPLAY "4. Track Attendance"
    DISPLAY "5. View Student Grades"
    DISPLAY "6. Exit"
    PROMPT "Enter your choice: "
    GET choice

    IF choice EQUALS "1" THEN
        PROMPT "Enter your Lecturer ID: "
        GET lecturer_id
        CALL view_assigned_modules(lecturer_id)
    ELSE IF choice EQUALS "2" THEN
        PROMPT "Enter Module Code: "
        GET module_code
        CALL view_student_list(module_code)
    ELSE IF choice EQUALS "3" THEN
        CALL record_grades()
    ELSE IF choice EQUALS "4" THEN
        CALL track_attendance()
    ELSE IF choice EQUALS "5" THEN
        PROMPT "Enter Module Code: "
```

```
            GET module_code
            CALL view_student_grades(module_code)
        ELSE IF choice EQUALS "6" THEN
            DISPLAY "Exiting Lecturer Menu."
            BREAK
        ELSE
            DISPLAY "Invalid choice, please try again."
        ENDIF
    ENDWHILE

ENDFUNCTION

END LecturerMenu
```

2.4 *Student Role*

```
BEGIN StudentMenu

    FUNCTION student_menu()

        FUNCTION view_available_modules()
            DISPLAY "Available Modules:"
            TRY
                OPEN "modules.txt" FOR READ
                DISPLAY "Module Code - Module Name (Credits)"
                FOR EACH line IN file DO
                    SPLIT line BY "," INTO details
                    IF LENGTH OF details IS GREATER THAN OR EQUAL TO 3 THEN
                        DISPLAY details[0] + " - " + details[1] + " (" + details[2] + "
credits)"
                    ENDIF
                ENDFOR
            CATCH FileNotFoundError EXCEPTION
                DISPLAY "Modules file not found."
            ENDTRY
        ENDFUNCTION

        FUNCTION enroll_in_module(student_id)
            DISPLAY "Enroll in Module..."
            PROMPT "Enter Module Code: "
            GET module_code
```

```
        TRY
            OPEN "enrolments.txt" FOR APPEND
            WRITE (student_id + "," + module_code) TO file
            SET timestamp TO CURRENT DATE AND TIME (FORMAT: YYYY-MM-DD:HH:MM:SS)
            DISPLAY "Enrolled in module " + module_code + " successfully on " + timestamp + "."
        CATCH FileNotFoundError EXCEPTION
            DISPLAY "Enrollments file not found."
        ENDTRY
    ENDFUNCTION

    FUNCTION unroll_from_module(student_id)
        DISPLAY "Unroll from Module..."
        PROMPT "Enter Module Code: "
        GET module_code

        TRY
            SET found TO FALSE
            READ ALL LINES FROM "enrolments.txt" INTO lines
            OPEN "enrolments.txt" FOR WRITE

            FOR EACH line IN lines DO
                IF line.strip() NOT EQUALS (student_id + "," + module_code) THEN
                    WRITE line TO file
                ELSE
                    SET found TO TRUE
                ENDIF
```

```
        ENDFOR

        IF found THEN
            SET timestamp TO CURRENT DATE AND TIME (FORMAT: YYYY-MM-DD:HH:MM:SS)
            DISPLAY "Unrolled from module " + module_code + " successfully on " + timestamp + "."
        ELSE
            DISPLAY "No enrollment found for module " + module_code + "."
        ENDIF
    CATCH FileNotFoundError EXCEPTION
        DISPLAY "Enrollments file not found."
    ENDTRY
ENDFUNCTION

FUNCTION view_grades(student_id)
    DISPLAY "Your Grades:"
    TRY
        OPEN "grades.txt" FOR READ
        FOR EACH line IN file DO
            SPLIT line BY "," INTO mod_code, sid, grade
            IF sid EQUALS student_id THEN
                DISPLAY "Module Code: " + mod_code + ", Grade: " + grade
            ENDIF
        ENDFOR
    CATCH FileNotFoundError EXCEPTION
        DISPLAY "Grades file not found."
    ENDTRY
```

```
    ENDFUNCTION


    FUNCTION access_attendance_record(student_id)
        PROMPT "Enter Module Code: "
        GET module_code
        DISPLAY "Attendance Records for Student ID " + student_id + " in Module "
+ module_code + ":"
        TRY
            OPEN "attendance.txt" FOR READ
            SET total_classes TO 0
            SET attended_classes TO 0

            FOR EACH line IN file DO
                SET line TO line.strip()
                IF line IS EMPTY THEN
                    CONTINUE
                ENDIF
                TRY
                    SPLIT line BY "," INTO mod_code, sid, status
                    IF sid EQUALS student_id AND mod_code EQUALS module_code
THEN
                        INCREMENT total_classes BY 1
                        IF status.lower() EQUALS "present" THEN
                            INCREMENT attended_classes BY 1
                        ENDIF
                    ENDIF
                CATCH ValueError EXCEPTION
                    DISPLAY "Skipping invalid line: " + line
```

```
            ENDTRY
        ENDFOR

        IF total_classes IS GREATER THAN 0 THEN
            SET attendance_percentage TO (attended_classes / total_classes) * 100
            DISPLAY "Total Classes: " + total_classes
            DISPLAY "Classes Attended: " + attended_classes
            DISPLAY "Attendance Percentage: " + attendance_percentage + "%"
        ELSE
            DISPLAY "No attendance records found for Module " + module_code + "
with your Student ID."
        ENDIF
    CATCH FileNotFoundError EXCEPTION
        DISPLAY "Attendance file not found."
    ENDTRY
ENDFUNCTION

WHILE TRUE DO
    DISPLAY "---------- Student Menu ----------"
    DISPLAY "1. View Available Modules"
    DISPLAY "2. Enroll in Module"
    DISPLAY "3. Unroll from Module"
    DISPLAY "4. View Grades"
    DISPLAY "5. Access Attendance Record"
    DISPLAY "6. Exit"
    PROMPT "Enter your choice: "
    GET choice
```

```
        IF choice EQUALS "1" THEN
            CALL view_available_modules()
        ELSE IF choice EQUALS "2" THEN
            PROMPT "Enter your Student ID: "
            GET student_id
            CALL enroll_in_module(student_id)
        ELSE IF choice EQUALS "3" THEN
            PROMPT "Enter your Student ID: "
            GET student_id
            CALL unroll_from_module(student_id)
        ELSE IF choice EQUALS "4" THEN
            PROMPT "Enter your Student ID: "
            GET student_id
            CALL view_grades(student_id)
        ELSE IF choice EQUALS "5" THEN
            PROMPT "Enter your Student ID: "
            GET student_id
            CALL access_attendance_record(student_id)
        ELSE IF choice EQUALS "6" THEN
            DISPLAY "Exiting Student Menu."
            BREAK
        ELSE
            DISPLAY "Invalid choice, please try again."
        ENDIF
    ENDWHILE
  ENDFUNCTION
END StudentMenu
```

2.5 *Registrar Role*

```
BEGIN RegistrarMenu

    FUNCTION registrar_menu()

        FUNCTION get_valid_input(prompt, required=True)
            WHILE TRUE DO
                DISPLAY prompt
                GET value
                SET value TO value.strip()
                IF value IS NOT EMPTY OR required IS FALSE THEN
                    RETURN value
                ELSE
                    DISPLAY "This field cannot be empty. Please try again."
                ENDIF
            ENDWHILE
        ENDFUNCTION

        FUNCTION validate_numeric_input(prompt)
            WHILE TRUE DO
                DISPLAY prompt
                GET value
                SET value TO value.strip()
                IF value IS NUMERIC THEN
                    RETURN value
                ELSE
                    DISPLAY "Invalid input. Please enter a numeric value."
```

```
        ENDIF
    ENDWHILE
ENDFUNCTION

FUNCTION validate_date_input(prompt)
    WHILE TRUE DO
        DISPLAY prompt
        GET date_str
        TRY
            PARSE date_str AS DATE (FORMAT: YYYY-MM-DD)
            RETURN date_str
        CATCH ValueError EXCEPTION
            DISPLAY "Invalid date format. Please use YYYY-MM-DD."
        ENDTRY
    ENDWHILE
ENDFUNCTION

FUNCTION register_new_student()
    DISPLAY "Register New Student"
    PROMPT "Enter Student ID: "
    GET student_id USING validate_numeric_input(prompt)

    IF CALL check_student_id_exists(student_id) THEN
        DISPLAY "Student ID " + student_id + " has already been registered."
        RETURN
    ENDIF
```

```
    PROMPT "Enter Student Name: "
    GET student_name USING get_valid_input(prompt)
    PROMPT "Enter Program Name: "
    GET program USING get_valid_input(prompt)

    TRY
      OPEN "students.txt" FOR APPEND
      WRITE (student_id + "," + student_name + "," + program) TO file
      SET timestamp TO CURRENT DATE AND TIME (FORMAT: YYYY-MM-DD
HH:MM:SS)
      DISPLAY "Student registered on: " + timestamp
      DISPLAY "Student registered successfully."
    CATCH FileNotFoundError EXCEPTION
      DISPLAY "Students file not found."
    ENDTRY
  ENDFUNCTION

  FUNCTION check_student_id_exists(student_id)
    TRY
      OPEN "students.txt" FOR READ
      FOR EACH line IN file DO
        SPLIT line BY "," INTO existing_student_id AND IGNORE
        IF student_id EQUALS existing_student_id THEN
          RETURN TRUE
        ENDIF
      ENDFOR
      RETURN FALSE
    CATCH FileNotFoundError EXCEPTION
```

```
        DISPLAY "Students file not found."
        RETURN FALSE
    ENDTRY
ENDFUNCTION

FUNCTION update_student_record()
    DISPLAY "Update Student Record"
    PROMPT "Enter Student ID to update: "
    GET student_id USING validate_numeric_input(prompt)

    TRY
        READ ALL LINES FROM "students.txt" INTO lines
        SET updated TO FALSE
        OPEN "students.txt" FOR WRITE

        FOR EACH line IN lines DO
            IF line STARTS WITH (student_id + ",") THEN
                PROMPT "Enter Updated Student Name: "
                GET student_name USING get_valid_input(prompt)
                PROMPT "Enter Updated Program Name: "
                GET program USING get_valid_input(prompt)
                WRITE (student_id + "," + student_name + "," + program) TO file
                SET updated TO TRUE
            ELSE
                WRITE line TO file
            ENDIF
        ENDFOR
```

```
            IF updated THEN
                SET timestamp TO CURRENT DATE AND TIME (FORMAT: YYYY-MM-
DD:HH:MM:SS)
                DISPLAY "Student information updated on: " + timestamp
                DISPLAY "Student record updated successfully."
            ELSE
                DISPLAY "Student ID not found."
            ENDIF
        CATCH FileNotFoundError EXCEPTION
            DISPLAY "Students file not found."
        ENDTRY
    ENDFUNCTION

    FUNCTION manage_enrollments()
        DISPLAY "Manage Enrollments"
        PROMPT "Enter Student ID: "
        GET student_id USING validate_numeric_input(prompt)
        PROMPT "Enter Module Code: "
        GET module_code USING get_valid_input(prompt)

        SET module_exists TO FALSE
        TRY
            OPEN "modules.txt" FOR READ
            FOR EACH line IN file DO
                IF line.strip() EQUALS module_code THEN
                    SET module_exists TO TRUE
                    BREAK
```

```
        ENDIF
    ENDFOR
CATCH FileNotFoundError EXCEPTION
    DISPLAY "Modules file not found."
ENDTRY

IF NOT module_exists THEN
    DISPLAY "Module code " + module_code + " does not exist."
    RETURN
ENDIF

PROMPT "Enter Action (Enroll/Unenroll): "
GET action
SET action TO action.lower()

TRY
    IF action EQUALS "enroll" THEN
        OPEN "enrolments.txt" FOR APPEND
        WRITE (student_id + "," + module_code) TO file
        SET timestamp TO CURRENT DATE AND TIME (FORMAT: YYYY-MM-
DD:HH:MM:SS)
        DISPLAY "Student enrolled in " + module_code + " on: " + timestamp
    ELSE IF action EQUALS "unenroll" THEN
        READ ALL LINES FROM "enrolments.txt" INTO lines
        SET found TO FALSE
        OPEN "enrolments.txt" FOR WRITE

        FOR EACH line IN lines DO
```

```
            IF line.strip() NOT EQUALS (student_id + "," + module_code) THEN
                WRITE line TO file
            ELSE
                SET found TO TRUE
            ENDIF
        ENDFOR

        IF found THEN
            DISPLAY "Student " + student_id + " unenrolled from module " +
module_code + "."
        ELSE
            DISPLAY "Enrollment record not found."
        ENDIF
    ELSE
        DISPLAY "Invalid action."
    ENDIF
  CATCH FileNotFoundError EXCEPTION
    DISPLAY "Enrollments file not found."
  ENDTRY
ENDFUNCTION

FUNCTION issue_transcript()
  DISPLAY "Issue Transcript"
  PROMPT "Enter Student ID: "
  GET student_id USING validate_numeric_input(prompt)

  SET timestamp TO CURRENT DATE AND TIME (FORMAT: YYYY-MM-DD
HH:MM:SS)
```

```
    TRY
        DISPLAY "Transcript for Student ID: " + student_id
        DISPLAY "Modules and Grades:"
        OPEN "grades.txt" FOR READ
        SET found TO FALSE

        FOR EACH line IN file DO
            SPLIT line BY "," INTO mod_code, sid, grade
            IF sid EQUALS student_id THEN
                DISPLAY "Module: " + mod_code + ", Grade: " + grade
                SET found TO TRUE
            ENDIF
        ENDFOR

        IF NOT found THEN
            DISPLAY "No grades found for this student."
        ENDIF
    CATCH FileNotFoundError EXCEPTION
        DISPLAY "Grades file not found."
    ENDTRY

    DISPLAY "Transcript generated on: " + timestamp
ENDFUNCTION

FUNCTION delete_student_record()
    DISPLAY "Delete Student Record"
```

```
    PROMPT "Enter Student ID to delete: "
    GET student_id USING validate_numeric_input(prompt)

    TRY
      READ ALL LINES FROM "students.txt" INTO lines
      SET deleted TO FALSE
      OPEN "students.txt" FOR WRITE

      FOR EACH line IN lines DO
        IF NOT line STARTS WITH (student_id + ",") THEN
          WRITE line TO file
        ELSE
          SET deleted TO TRUE
        ENDIF
      ENDFOR

      IF deleted THEN
        SET timestamp TO CURRENT DATE AND TIME (FORMAT: YYYY-MM-
DD:HH:MM:SS)
        DISPLAY "Student record deleted on: " + timestamp
        DISPLAY "Student record deleted successfully."
      ELSE
        DISPLAY "Student ID not found."
      ENDIF
    CATCH FileNotFoundError EXCEPTION
      DISPLAY "Students file not found."
    ENDTRY
  ENDFUNCTION
```

```
FUNCTION view_student_information()
    DISPLAY "View Student Information"
    PROMPT "Enter Student ID to view: "
    GET student_id USING validate_numeric_input(prompt)

    TRY
        OPEN "students.txt" FOR READ
        SET found TO FALSE

        FOR EACH line IN file DO
            IF line STARTS WITH (student_id + ",") THEN
                SET timestamp TO CURRENT DATE AND TIME (FORMAT: YYYY-
MM-DD:HH:MM:SS)
                DISPLAY "Student info viewed on: " + timestamp
                DISPLAY "Student Details:"
                SPLIT line BY "," INTO id, name, program
                DISPLAY "ID: " + id + ", Name: " + name + ", Program: " + program
                SET found TO TRUE
                BREAK
            ENDIF
        ENDFOR

        IF NOT found THEN
            DISPLAY "Student ID not found."
        ENDIF
    CATCH FileNotFoundError EXCEPTION
        DISPLAY "Students file not found."
```

```
    ENDTRY
ENDFUNCTION

WHILE TRUE DO
    DISPLAY "---------- Registrar Menu ----------"
    DISPLAY "1. Register New Student"
    DISPLAY "2. Update Student Record"
    DISPLAY "3. Manage Enrollments"
    DISPLAY "4. Issue Transcript"
    DISPLAY "5. Delete Student Record"
    DISPLAY "6. View Student Information"
    DISPLAY "7. Exit"
    PROMPT "Enter your choice: "
    GET choice

    IF choice EQUALS "1" THEN
        CALL register_new_student()
    ELSE IF choice EQUALS "2" THEN
        CALL update_student_record()
    ELSE IF choice EQUALS "3" THEN
        CALL manage_enrollments()
    ELSE IF choice EQUALS "4" THEN
        CALL issue_transcript()
    ELSE IF choice EQUALS "5" THEN
        CALL delete_student_record()
    ELSE IF choice EQUALS "6" THEN
        CALL view_student_information()
```

```
        ELSE IF choice EQUALS "7" THEN
            DISPLAY "Exiting Registrar Menu."
            BREAK
        ELSE
            DISPLAY "Invalid choice, please try again."
        ENDIF
    ENDWHILE

ENDFUNCTION

END RegistrarMenu
```

***2.6** Accountant Role*

```
BEGIN AccountantMenu

  FUNCTION validate_numeric_input(prompt)
    WHILE TRUE DO
      DISPLAY prompt
      GET value
      SET value TO value.strip()
      IF value IS NUMERIC THEN
        RETURN value
      ELSE
        DISPLAY "Invalid input. Please enter a numeric value."
      ENDIF
    ENDWHILE
  ENDFUNCTION

  FUNCTION validate_date_input(prompt)
    WHILE TRUE DO
      DISPLAY prompt
      GET date_str
      TRY
        PARSE date_str AS DATE (FORMAT: YYYY-MM-DD)
        RETURN date_str
      CATCH ValueError EXCEPTION
        DISPLAY "Invalid date format. Please use YYYY-MM-DD."
      ENDTRY
    ENDWHILE
```

```
    ENDFUNCTION

    FUNCTION accountant_menu()

        FUNCTION record_tuition_fees()
            DISPLAY "Record Tuition Fees"
            PROMPT "Enter Student ID: "
            GET student_id
            PROMPT "Enter Amount Paid: "
            GET amount_paid USING validate_numeric_input(prompt)
            PROMPT "Enter Date of Payment (YYYY-MM-DD): "
            GET date_of_payment USING validate_date_input(prompt)

            TRY
                OPEN "fees.txt" FOR APPEND
                WRITE (student_id + "," + amount_paid + "," + date_of_payment) TO file
                DISPLAY "Tuition fee payment recorded successfully."
            CATCH FileNotFoundError EXCEPTION
                DISPLAY "Fees file not found."
            ENDTRY
        ENDFUNCTION

        FUNCTION view_outstanding_fees()
            DISPLAY "View Outstanding Fees"
            SET timestamp TO CURRENT DATE AND TIME (FORMAT: YYYY-MM-
DD HH:MM:SS)
            DISPLAY "Reports generated on: " + timestamp
```

```
        TRY
            DISPLAY "Outstanding Fees:"
            OPEN "fees.txt" FOR READ
            FOR EACH line IN file DO
                SPLIT line BY "," INTO details
                IF LENGTH OF details EQUALS 3 THEN
                    SET student_id, amount_paid, date_of_payment TO details
                    DISPLAY "Student ID: " + student_id + ", Outstanding Fees: " +
amount_paid
                ENDIF
            ENDFOR
        CATCH FileNotFoundError EXCEPTION
            DISPLAY "Fees file not found."
        ENDTRY
    ENDFUNCTION

    FUNCTION update_payment_records()
        DISPLAY "Update Payment Records"
        PROMPT "Enter Student ID: "
        GET student_id
        PROMPT "Enter Updated Amount Paid: "
        GET new_payment USING validate_numeric_input(prompt)
        PROMPT "Enter Updated Payment Date (YYYY-MM-DD): "
        GET payment_date USING validate_date_input(prompt)

        TRY
            READ ALL LINES FROM "fees.txt" INTO lines
            SET updated TO FALSE
```

```
            OPEN "fees.txt" FOR WRITE

            FOR EACH line IN lines DO
                IF line STARTS WITH (student_id + ",") THEN
                    WRITE (student_id + "," + new_payment + "," + payment_date) TO
file
                    SET updated TO TRUE
                ELSE
                    WRITE line TO file
                ENDIF
            ENDFOR

            IF updated THEN
                DISPLAY "Payment record updated successfully."
            ELSE
                DISPLAY "Student ID not found in payment records."
            ENDIF
        CATCH FileNotFoundError EXCEPTION
            DISPLAY "Fees file not found."
        ENDTRY
    ENDFUNCTION

    FUNCTION issue_fee_receipts()
        DISPLAY "Issue Fee Receipt"
        PROMPT "Enter Student ID: "
        GET student_id

        TRY
```

```
        OPEN "fees.txt" FOR READ
        FOR EACH line IN file DO
          IF line STARTS WITH (student_id + ",") THEN
            SPLIT line BY "," INTO details
            DISPLAY "Fee Receipt:"
            DISPLAY "Student ID: " + details[0]
            DISPLAY "Amount Paid: " + details[1]
            DISPLAY "Payment Date: " + details[2]
            RETURN
          ENDIF
        ENDFOR
        DISPLAY "No fee record found for the given Student ID."
      CATCH FileNotFoundError EXCEPTION
        DISPLAY "Fees file not found."
      ENDTRY
    ENDFUNCTION

    FUNCTION view_financial_summary()
      DISPLAY "View Financial Summary"
      SET timestamp TO CURRENT DATE AND TIME (FORMAT: YYYY-MM-
DD HH:MM:SS)
      DISPLAY "Reports generated on: " + timestamp

      TRY
        SET total_fees_collected TO 0
        OPEN "fees.txt" FOR READ
        FOR EACH line IN file DO
          SPLIT line BY "," INTO details
```

```
            IF LENGTH OF details IS GREATER THAN OR EQUAL TO 2 THEN
                ADD details[1] (AS FLOAT) TO total_fees_collected
            ENDIF
        ENDFOR
        DISPLAY "Total Fees Collected: " + total_fees_collected
    CATCH FileNotFoundError EXCEPTION
        DISPLAY "Fees file not found."
    CATCH ValueError EXCEPTION
        DISPLAY "Error reading fee amounts. Ensure all data is numeric."
    ENDTRY
ENDFUNCTION

ENDFUNCTION

END AccountantMenu
```

CHAPTER 3: SOURCE CODE & EXPLAINATION

3.1 *Application of Storage Types*

3.1.1 *Lists*

Lists are at the centre of how our system, it handles the collection of records whether that is from a student or from accountants. Data from our txt files are read into lists, enabling reliable and easier processing and operations. An example of this is the outstanding fee function that reads a txt file for example our student.txt line by line, then creates a dictionary for each record and stores all dictionaries in a list.

```
35
36      def view_outstanding_fees():
37          print("View Outstanding Fees")
38          timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
39          print(f"Reports generated on: {timestamp}")
40          try:
41              print("Outstanding Fees:")
42              with open("fees.txt", "r") as file:
43                  for line in file:
44                      details = line.strip().split(",")
45                      if len(details) == 3:
46                          student_id, amount_paid, date_of_payment = details
47                          print(f"Student ID: {student_id}, Amount Paid: {amount_paid}, Date: {date_of_payment}")
48
49          except FileNotFoundError:
50              print("Fees file not found.")
51
```

Figure 3.1.1 List

This allows us to filter, search for and update records for student. The structure of the list allows for data to be accessed sequentially or modified which is important for displaying all student records.

3.1.2 *Text Files*

Text files are used for constant storage. In our system we have used txt files to ensure that data remains connected across our different program execution.

```
def delete_student_record():
    print("Delete Student Record")
    student_id = validate_numeric_input("Enter Student ID to delete: ")

    try:
        with open("students.txt", "r") as file:
            lines = file.readlines()

        deleted = False
        with open("students.txt", "w") as file:
            for line in lines:
                if not line.startswith(student_id + ","):
                    file.write(line)
                else:
                    deleted = True
```

Figure 3.1.2 Text file

In this screenshot, we have taken a section of our delete student_record function which shows how we are able to write lists back into a text file. This process ensures changes are made during runtime, such as deleting the student record from the data.

3.2 Application of Control Structure

3.2.1 *Iteration (for loop)*

Loops (for and while) allow for repetitive actions, for our university management system we used this to iterate through student grade records to find a match. An example of this is how we use the loop to iterate through the student grade list by processing each student grade one by one.

```
def view_grades(student_id):
    print("Your Grades:")
    try:
        with open("grades.txt", "r") as file:
            for line in file:
                mod_code, sid, grade = line.strip().split(",")
                if sid == student_id:
                    print(f"Module Code: {mod_code}, Grade: {grade}")
    except FileNotFoundError:
        print("Grades file not found.")
```

Figure 3.2.1 Iteration for loop

3.2.2 Iteration: (while loop)

```
while True:
    print("---------- Student Menu ----------")
    print("1. View Available Modules")
    print("2. Enroll in Module")
    print("3. Unroll from Module")
    print("4. View Grades")
    print("5. Access Attendance Record")
    print("6. Exit")
    choice = input("Enter your choice: ").strip()

    if choice == "1":
        view_available_modules()
    elif choice == "2":
        student_id = input("Enter your Student ID: ").strip()
        enroll_in_module(student_id)
    elif choice == "3":
        student_id = input("Enter your Student ID: ").strip()
        unroll_from_module(student_id)
    elif choice == "4":
        student_id = input("Enter your Student ID: ").strip()
        view_grades(student_id)
    elif choice == "5":
        student_id = input("Enter your Student ID: ").strip()
        access_attendance_record(student_id)
    elif choice == "6":
        print("Exiting Student Menu.")
        break
    else:
        print("Invalid choice, please try again.")
```

Figure 3.2.2 While Loop

While loop has a slightly different with the For loop. For loop has a limit like how long we want this output or program to be executed while loop do not have a limit. In figure 3.2.2 the while loop is true means that the main menu will repeat every time until the user exit which mean it is break which make it become false.

3.2.3 Selective: if-else nested if

Nested if means that inside the if there is still another if. In figure 3.2.3 it shows that there are “if Sid == ...” means it is the first condition, and there is also if status. Lower () =...” mean it is the second condition. So, to output to record present in the data, the system needs to make sure that the student is in the class and module and the lecturer input that the student is present on that class

```
def access_attendance_record(student_id):
    module_code = input("Enter Module Code: ").strip()
    print(f"Attendance Records for Student ID {student_id} in Module {module_code}:")
    try:
        with open("attendance.txt", "r") as file:
            total_classes, attended_classes = 0, 0
            for line in file:
                line = line.strip()
                if not line:
                    continue
                try:
                    mod_code, sid, status = line.split(",")
                    if sid == student_id and mod_code == module_code:
                        total_classes += 1
                        if status.lower() == "present":
                            attended_classes += 1
                except ValueError:
                    print(f"Skipping invalid line: {line}")

            if total_classes > 0:
                attendance_percentage = (attended_classes / total_classes) * 100
                print(f"Total Classes: {total_classes}")
                print(f"Classes Attended: {attended_classes}")
                print(f"Attendance Percentage: {attendance_percentage:.2f}%")
            else:
                print(f"No attendance records found for Module {module_code} with your Student ID.")
    except FileNotFoundError:
        print("Attendance file not found.")
```

Figure 3.2.3 Selective if else nested if

3.3 *Application of Try…Except*

Our university management system utilizes try-except blocks to handle potential runtime errors, which helps us avoid program problems which crashes. An example of this is when extracting a header from data. The use of try-except ensures the system will see it useful when the data is missing or not properly formatted. If we did not use this to handle errors, we would run into a bigger issue which can usually cause syntax errors.

```
def view_financial_summary():
    print("View Financial Summary")
    timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    print(f"Reports generated on: {timestamp}")
    try:
        total_fees_collected = 0
        with open("fees.txt", "r") as file:
            for line in file:
                details = line.strip().split(",")
                if len(details) >= 2:
                    total_fees_collected += float(details[1])
        print(f"Total Fees Collected: {total_fees_collected:.2f}")
    except FileNotFoundError:
        print("Fees file not found.")
    except ValueError:
        print("Error reading fee amounts. Ensure all data is numeric.")
```

Figure 3.3.1 Try...except block

3.4 *Application of Validation*

```
def validate_numeric_input(prompt):
    """Prompt user for numeric input and validate it."""
    while True:
        value = input(prompt).strip()
        if value.isnumeric():
            return value
        print("Invalid input. Please enter a numeric value.")
```

Figure 3.4.1 validate numeric input

```
def validate_date_input(prompt):
    """Prompt user for a date and validate its format (YYYY-MM-DD)."""
    while True:
        date_str = input(prompt).strip()
        try:
            datetime.strptime(date_str, "%Y-%m-%d")
            return date_str
        except ValueError:
            print("Invalid date format. Please use YYYY-MM-DD.")
```

Figure 3.4.2 validate date input

Validation ensures that only correct and meaningful data is processed. In figure 3. 4.1 we found a way to make sure the user to input number if not followed it will ask the user to input again. Same as 3.4.2 and 3.4.3 which has the same function as figure 3.4.1 but has different function.

This prevents invalid updates to number and date as if a user attempts to modify a number and date that does not exist it will validate and then push them to the else block. We also ensured that if a user is to make such a mistake, we would inform them through the print block which allows them to be informed of their mistake and then correct their input.

3.5 *The Flow of Add Function*

```
try:
    if option == "add":
        lecturer_id = validate_numeric_input("Enter lecturer ID: ")
        lecturer_name = input("Enter lecturer name: ").strip()
        department = input("Enter department: ").strip()
        HiringDate = validate_date_input("Enter Date of Registration [YYYY-MM-DD]:")

        with open("lecturers.txt", "a") as file:
            file.write(f"{lecturer_id},{lecturer_name},{department},{HiringDate}\n")
        print("Lecturer added successfully.")
```

Figure 3.5.1 add function

In figure 3.5.1 For the flow of add functions, the user input the lecturer ID, name, department, and hiring date. After the user input everything, the code will open the lecturer.txt file and add the lecturer data inside the txt and print “lecturer added successfully.”

3.6 *The Flow of Update Function*

```
elif option == "update":
    module_code = input("Enter module code to update: ").strip()
    found = False

    with open("modules.txt", "r") as file:
        lines = file.readlines()

    with open("modules.txt", "w") as file:
        for line in lines:
            if line.startswith(module_code + ","):
                module_name = input("Enter new module name: ").strip()
                credits = validate_numeric_input("Enter new course credits: ")
                file.write(f"{module_code},{module_name},{credits}\n")
                found = True
            else:
                file.write(line)

    if found:
        print("Module updated successfully.")
    else:
        print("Module not found.")
```

Figure 3.6.1 Update function

In figure 3.6.1, the flow of update is it will ask the user to enter which module to update and the code will open the modules. txt and search for it. If it was found inside the data, it will ask the user to input the new module name and the course credit and it will save the data inside and print module updated successfully. If it does not find any data related to the module name, it will print module not found.

3.7 The Flow of Delete/Remove Function

```
        elif option == "remove":
            student_id = validate_numeric_input("Enter student ID to remove: ")
            found = False

            with open("students.txt", "r") as file:
                lines = file.readlines()

            with open("students.txt", "w") as file:
                for line in lines:
                    if not line.startswith(student_id + ","):
                        file.write(line)
                    else:
                        found = True

            if found:
                print("Student removed successfully.")
            else:
                print("Student not found.")
        else:
            print("Invalid option. Please try again.")
    except FileNotFoundError:
        print("Error: File not found. Ensure 'students.txt' exists.")
```

Figure 3.7.1 delete function

In figure 3.7.1 for the flow for delete function is it will ask the user to put student ID and it will open the student file and check for the student ID. If they find the student ID in the record, the data related to that student Id will be deleted. While if they do not find it will say print student not found.

3.8 The Flow of Generating Reports

```
def generate_reports():
    """Generate reports for the system with a timestamp."""
    print("\nGenerating Reports")
    try:
        student_count = sum(1 for _ in open("students.txt", "r"))
        module_count = sum(1 for _ in open("modules.txt", "r"))
        lecturer_count = sum(1 for _ in open("lecturers.txt", "r"))
        timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

        print(f"Reports generated on: {timestamp}")
        print(f"Total Students: {student_count}")
        print(f"Active Modules: {module_count}")
        print(f"Total Lecturers: {lecturer_count}")
    except FileNotFoundError:
        print("Error: One or more files are missing. Ensure all required files exist.")
```

3.8.1 Figure 3.8.0 Generating Reports

In figure 3.8.1 the code will start by opening the txt that was needed to generate the reports like student name, module id, lecturer name. After that, the code will start to print all the information that was found in the record.

3.9 Additional Features

```
def update_payment_records():
    print("Update Payment Records")
    student_id = input("Enter Student ID: ").strip()
    new_payment = validate_numeric_input("Enter Updated Amount Paid: ")
    payment_date = validate_date_input("Enter Updated Payment Date (YYYY-MM-DD): ")

    try:
        with open("fees.txt", "r") as file:
            lines = file.readlines()

        updated = False
        with open("fees.txt", "w") as file:
            for line in lines:
                if line.startswith(student_id + ","):
                    file.write(f"{student_id},{new_payment},{payment_date}\n")
                    updated = True
                else:
                    file.write(line)

        if updated:
            print("Payment record updated successfully.")
        else:
            print("Student ID not found in payment records.")
    except FileNotFoundError:
        print("Fees file not found.")
```

Figure 3.9.1 flag feature

Flag feature is like a switch. Its function is to avoid buggy release. The flow of this can be seen in figure 3.9.1 the flag feature is located at updated = false. So, it is a switch, in this case, false mean on and True means off. So, this can tell the program when to stop writing the data and when to start writing the data. The flow can be seen in figure 3.9.1 first it will ask for input which are the student ID, new payment, and the payment date. And after the opening the fee.txt first we will tell the program that the data is error or still need to be updated which is “updated == false” and after we re-enter all the correct one, we will tell the data is already correct by “updated ==” correct.”

```python
def login():
    print('Please enter Login credentials')
    print('UserName: "UMSLOGIN", Password: "12589" ')
    username = get_valid_input("Enter Username: ")
    password = get_valid_input("Enter password: ")

    # Assuming you have a file 'security.txt' with username,password pairs
    try:
        with open('security.txt', "r") as file:
            users = file.readlines()
            for user in users:
                stored_name, stored_password = user.strip().split(",")
                if username == stored_name and password == stored_password:
                    print("Login successful!")
                    return True
            print("Invalid username or password. Try again.")
            return False
    except FileNotFoundError:
        print("Security file not found.")
        return False
```

Figure 3.9.2 Security feature

We have a login feature or the security feature. How this work if the user put the right username and password it will enter. However, if not it will decline. The flow is the program will ask the user for username and password. After getting it from user, the program will check the security.txt which has a username and password inside. If it same with the username or password inside it will let the user continue however if the program found that the username or password is wrong, it will tell the user to re-enter the username and password till correct.

```python
def view_student_list(module_code):
    print(f"Student List for Module: {module_code}")
    timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    print(f"time-stamp: {timestamp}")
    try:
        with open("enrolments.txt", "r") as file:
            print("Enrolled Students:")
            found = False
            for line in file:
                line = line.strip()
                if not line:
                    continue  # Skip empty lines
                try:
                    student_id, mod_code = line.split(",")
                    if mod_code == module_code:
                        print(f"Student ID: {student_id}")
                        found = True
                except ValueError:
                    print(f"Skipping invalid line: {line}")
            if not found:
                print(f"No students are enrolled in module {module_code}.")
    except FileNotFoundError:
        print("Enrollments file not found.")
```

Figure 3.9.3 Timestamp

Time stamp is used to give the user information about what time and date use the function. The flow of time stamp is first we need to import date and time and then need to use a variable to represent the datetime and just call the variable to be used. As in figure 3.9.3 after the output come out it will also show the time and date that the output come at.

CHAPTER 4: INPUT/OUTPUT SAMPLE

4.1 UMS Main Menu

```
-------Welcome to University Management System-----------
Please enter Login credentials
UserName: "UMSLOGIN", Password: "12589"
Enter Username:
```

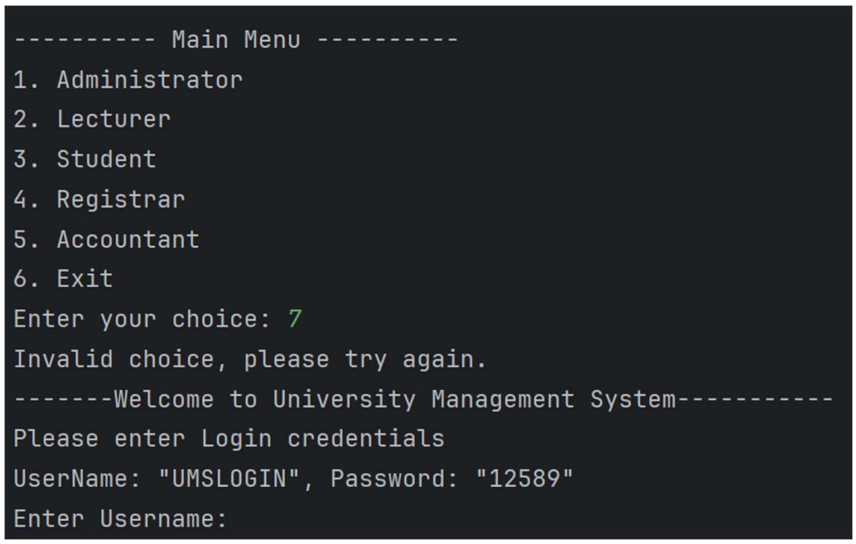
Upon entering the university management system, you are welcomed by this main menu. The user is required to enter the username and password to access the features in the system. The username and password have been provided for simplicity purposes.

```
Please enter Login credentials
UserName: "UMSLOGIN", Password: "12589"
Enter Username: UMSLOGIN
Enter password: 12589
Login successful!

---------- Main Menu ----------
1. Administrator
2. Lecturer
3. Student
4. Registrar
5. Accountant
6. Exit
Enter your choice:
```

Upon entering the username and password correctly, the user is presented with the main menu. Here the user can choose what feature to access based on their role.

```
-------Welcome to University Management System-----------
Please enter Login credentials
UserName: "UMSLOGIN", Password: "12589"
Enter Username: UMSLOGIN
Enter password: 12345
Invalid username or password. Try again.
Access Denied! Please try logging in again.
-------Welcome to University Management System-----------
Please enter Login credentials
UserName: "UMSLOGIN", Password: "12589"
Enter Username:
```

If the user enters an invalid username or password, an error message will be shown stating that the username or password entered was invalid and access will be denied. The user will then be asked to enter the username and password again.



If the user enters an invalid choice in the main menu, an error message will be shown stating the choice was invalid and the user will be asked to enter the username and password once again

4.2 Administrator Role

```
---------- Main Menu ----------
1. Administrator
2. Lecturer
3. Student
4. Registrar
5. Accountant
6. Exit
Enter your choice: 1

---------- University Administrator ------------
1. Manage Modules
2. Manage Students
3. Manage Lecturers
4. View All Data
5. Generate Reports
6. Exit
Enter your choice:
```

When the user selects Administrator (1) from the main menu it brings up the University Administrator Menu, from where the user will be able to add and remove modules, students, and lecturers, as well as view data and reports.

```
---------- University Administrator ------------
1. Manage Modules
2. Manage Students
3. Manage Lecturers
4. View All Data
5. Generate Reports
6. Exit
Enter your choice: 1

Module Management
Choose an option (Add, Update, Remove): Add
Enter module code: CS103
Enter module name: Computer Science
Enter module credits: 3
Module added successfully.
```

```
---------- University Administrator ------------
1. Manage Modules
2. Manage Students
3. Manage Lecturers
4. View All Data
5. Generate Reports
6. Exit
Enter your choice: 1

Module Management
Choose an option (Add, Update, Remove): Update
Enter module code to update: CS103
Enter new module name: PHY104
Enter new course credits: 4
Module updated successfully.
```

```
---------- University Administrator ------------
1. Manage Modules
2. Manage Students
3. Manage Lecturers
4. View All Data
5. Generate Reports
6. Exit
Enter your choice: 1

Module Management
Choose an option (Add, Update, Remove): remove
Enter a module code to remove: CS104
Module removed successfully.
```

A user can manage the modules by selecting an option either to Add, Update, or Remove a module.

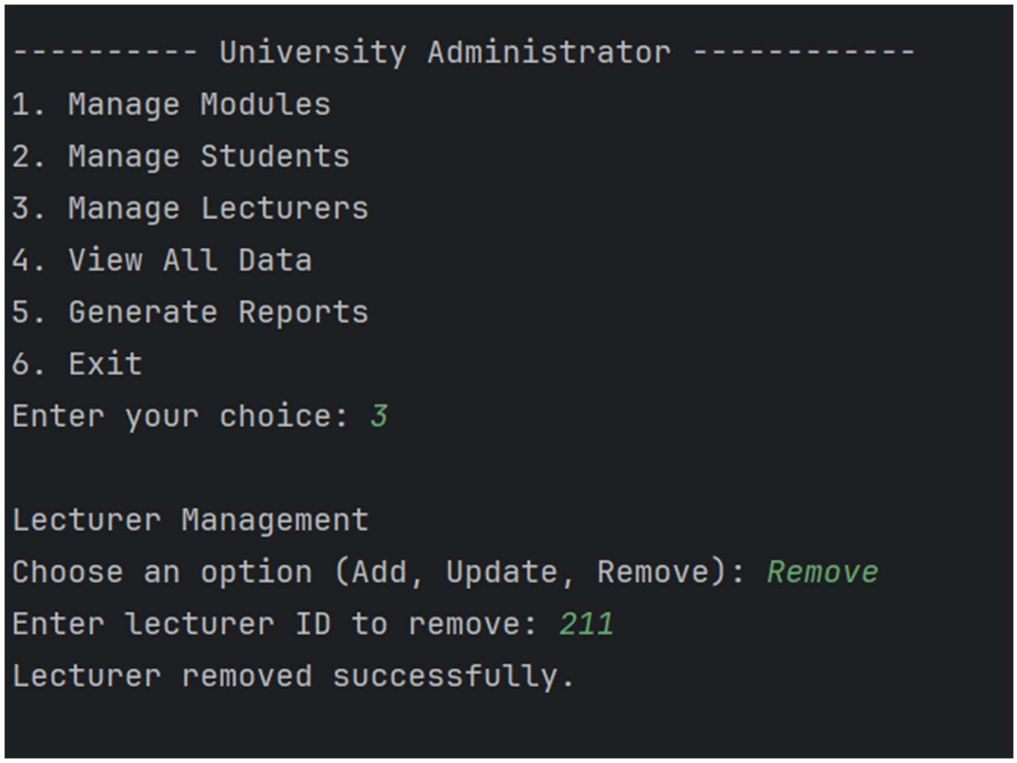
```
---------- University Administrator ------------
1. Manage Modules
2. Manage Students
3. Manage Lecturers
4. View All Data
5. Generate Reports
6. Exit
Enter your choice: 2

Student Management
Choose an option (Add, Update, Remove): Add
Enter student ID: 104
Enter student name: Bob
Enter department: Mathematics
Enter Date of Admission [YYYY-MM-DD]:2 2025-1-19
Student added successfully.
```

```
---------- University Administrator ------------
1. Manage Modules
2. Manage Students
3. Manage Lecturers
4. View All Data
5. Generate Reports
6. Exit
Enter your choice: 2

Student Management
Choose an option (Add, Update, Remove): Remove
Enter student ID to remove: 104
Student removed successfully.
```

The user can manage students by selecting to Add, Update, or Remove student records.

```
---------- University Administrator ------------
1. Manage Modules
2. Manage Students
3. Manage Lecturers
4. View All Data
5. Generate Reports
6. Exit
Enter your choice: 3

Lecturer Management
Choose an option (Add, Update, Remove): Add
Enter lecturer ID: 211
Enter lecturer name: Jonathan
Enter department: Physics
Enter Date of Registration [YYYY-MM-DD]:2025-1-19
Lecturer added successfully.
```

```
---------- University Administrator ------------
1. Manage Modules
2. Manage Students
3. Manage Lecturers
4. View All Data
5. Generate Reports
6. Exit
Enter your choice: 3

Lecturer Management
Choose an option (Add, Update, Remove): Update
Enter lecturer ID to update: 211
Enter new lecturer name: Jonathan
Enter new department: Mathematics
Lecturer updated successfully.
```



The user can manage lecturers by choosing an option to Add, Update, or Remove lecturer data. In case the user chooses Remove, he will be asked for the ID of the lecturer, and if that ID does not exist, an error message will appear stating that the lecturer was not found.

```
---------- University Administrator ------------
1. Manage Modules
2. Manage Students
3. Manage Lecturers
4. View All Data
5. Generate Reports
6. Exit
Enter your choice: 4

Viewing All Data:
Data viewed on: 2025-01-19 20:11:39

All Students Data:
101,John Doe,Computer Science,2022-03-14
102,Jane Smith,Mathematics,2022-11-27
103,Alice Johnson,Physics,2023-05-06
105,Charlie Davis,Computer Science,2022-12-02
106,Daisy Evans,Mathematics,2024-04-22
107,Edward Green,Mathematics,2023-07-30
108,Fiona Hall,Engineering,2024-10-15
109,George White,Computer Science,2024-02-08
110,Hannah Black,Mathematics,2023-11-11


All Modules Data:
CS101,Programming Basics,3,201
CS102,Data Structures,4,205
CS103,PHY104,4
PHY101,Classical Mechanics,3,202
PHY102,Quantum Physics,4,206
MAT101,Calculus,3,203
MAT102,Linear Algebra,4,207
ENG101,Literature,3,204
ENG102,Grammar,4,208
```

```
All Lecturers Data:
201,Dr. Alan Turing,Computer Science,2022-04-17
202,Dr. Marie Curie,Physics,2023-06-05
203,Dr. Katherine Johnson,Mathematics,2023-07-22
204,Dr. Nikola Tesla,Engineering,2023-11-07
205,Dr. Ada Lovelace,Computer Science,2022-08-06
206,Dr. Rosalind Franklin,Physics,2024-11-13
207,Dr. John Nash,Mathematics,2023-08-07
208,Dr. Thomas Edison,Engineering, 2024-12-14
209,Dr. Grace Hopper,Computer Science, 2022-12-28
210,Dr. Stephen Hawking,Physics, 2022-08-31
```

The user can View all Data which has records of students details, module details, as well as the lecturers details. Aside from that, this system provides the users with access logs that include the dates and times when the said information was accessed. bomb

```
---------- University Administrator ------------
1. Manage Modules
2. Manage Students
3. Manage Lecturers
4. View All Data
5. Generate Reports
6. Exit
Enter your choice: 5

Generating Reports
Reports generated on: 2025-01-19 20:12:48
Total Students: 9
Active Modules: 9
Total Lecturers: 13
```

The user can also Generate Reports that will tell the user the Total Students, Active Modules, and the Total Lecturers.

```
---------- University Administrator ------------
1. Manage Modules
2. Manage Students
3. Manage Lecturers
4. View All Data
5. Generate Reports
6. Exit
Enter your choice: 6
Exiting Administrator menu.
-------Welcome to University Management System-----------
Please enter Login credentials
UserName: "UMSLOGIN", Password: "12589"
Enter Username:
```

IF the user decides to leave the University Administrator Menu, he can choose Option 6 which will then Exit the user from the Administrator Menu and back to the Login.

4.3 Lecturer Role

```
---------- Main Menu ----------
1. Administrator
2. Lecturer
3. Student
4. Registrar
5. Accountant
6. Exit
Enter your choice: 2
---------- Lecturer Menu ----------
1. View Assigned Modules
2. View Student List for a Module
3. Record Grades
4. Track Attendance
5. View Student Grades
6. Exit
Enter your choice:
```

When the user chooses the Lecturer option (2) from the Main Menu, he will then be able to see the Lecturer Menu.

```
---------- Main Menu ----------
1. Administrator
2. Lecturer
3. Student
4. Registrar
5. Accountant
6. Exit
Enter your choice: 8
Invalid choice, please try again.
-------Welcome to University Management System----------
Please enter Login credentials
UserName: "UMSLOGIN", Password: "12589"
Enter Username:
```

If the user enters an invalid choice, an error message will be shown stating that the choice is invalid, and the user will be asked to enter their choice again.

```
---------- Lecturer Menu ----------
1. View Assigned Modules
2. View Student List for a Module
3. Record Grades
4. Track Attendance
5. View Student Grades
6. Exit
Enter your choice: 1
Enter your Lecturer ID: 201
Viewing Assigned Modules...
time-stamp: 2025-01-19 12:45:46
Modules assigned:
CS101 - Programming Basics (3 credits)
CS103 - Algorithms (4 credits)
```

If the user chooses option 1, they will be able to view the assigned modules by entering on of the Lecturer ID that is provided in the Lecturer text file.

```
---------- Lecturer Menu ----------
1. View Assigned Modules
2. View Student List for a Module
3. Record Grades
4. Track Attendance
5. View Student Grades
6. Exit
Enter your choice: 2
Enter Module Code: CS101
Student List for Module: CS101
time-stamp: 2025-01-19 12:53:33
Enrolled Students:
Student ID: 101
Student ID: 102
```

If the user selects option 2, he or she will be able to see list of students who have registered for a particular module by entering Module Code. The system will then show the information of the students who have registered for that Module.

```
---------- Lecturer Menu ----------
1. View Assigned Modules
2. View Student List for a Module
3. Record Grades
4. Track Attendance
5. View Student Grades
6. Exit
Enter your choice: 2
Enter Module Code: CT102
Student List for Module: CT102
time-stamp: 2025-01-19 13:29:05
Enrolled Students:
No students are enrolled in module CT102.
```

If the user enters a wrong Module Code, he will then be shown an error saying stating that no students are enrolled in this module

```
---------- Lecturer Menu ----------
1. View Assigned Modules
2. View Student List for a Module
3. Record Grades
4. Track Attendance
5. View Student Grades
6. Exit
Enter your choice: 3
Recording Grades...
Enter Module Code: CS101
Enter Student ID: 103
Enter Grade: B
Grade recorded successfully.
```

If the User Selects option 3, he will then be able to Record Grades for a student by typing the module Code as well as the Student ID, then the user will be able to choose which grade the student will receive

```
---------- Lecturer Menu ----------
1. View Assigned Modules
2. View Student List for a Module
3. Record Grades
4. Track Attendance
5. View Student Grades
6. Exit
Enter your choice: 4
Tracking Attendance...
Enter Module Code: PHY102
Enter Student ID: 105
Enter Attendance Status (Present/Absent): Present
Attendance recorded successfully.
```

If the user selects option 4, the user can track attendance by entering the Module Code, Student ID, and choosing if the student is "Present" or "Absent." The attendance record is then saved in the Attendance file.

```
---------- Lecturer Menu ----------
1. View Assigned Modules
2. View Student List for a Module
3. Record Grades
4. Track Attendance
5. View Student Grades
6. Exit
Enter your choice: 5
Enter Module Code: CS101
Viewing Grades for Module: CS101
time-stamp: 2025-01-19 13:15:36
Student Grades:
Student ID: 101, Grade: A
Student ID: 102, Grade: B
Student ID: 101, Grade: A
Student ID: 103, Grade: B
```

If the user selects option 5, the user can view Students grades for a course by simply typing the Module Code.

```
---------- Lecturer Menu ----------
1. View Assigned Modules
2. View Student List for a Module
3. Record Grades
4. Track Attendance
5. View Student Grades
6. Exit
Enter your choice: 6
Exiting Lecturer Menu.
```

If the user decides to leave the Lecturer Menu, he can choose option 6 where it will then Exit the Lecturer Menu.

4.4 Student Role

```
---------- Main Menu ----------
1. Administrator
2. Lecturer
3. Student
4. Registrar
5. Accountant
6. Exit
Enter your choice: 3
---------- Student Menu ----------
1. View Available Modules
2. Enroll in Module
3. Unroll from Module
4. View Grades
5. Access Attendance Record
6. Exit
```

When the user selects the student option (3) from the Main Menu, they will be directed to the Student Menu.

```
---------- Student Menu ----------
1. View Available Modules
2. Enroll in Module
3. Unroll from Module
4. View Grades
5. Access Attendance Record
6. Exit
Enter your choice: 1
Available Modules:
Module Code - Module Name (Credits)
CS101 - Programming Basics (3 credits)
CS102 - Data Structures (4 credits)
CS103 - Algorithms (4 credits)
PHY101 - Classical Mechanics (3 credits)
PHY102 - Quantum Physics (4 credits)
MAT101 - Calculus (3 credits)
MAT102 - Linear Algebra (4 credits)
ENG101 - Literature (3 credits)
ENG102 - Grammar (4 credits)
CS104 - Artificial Intelligence (5 credits)
```

If the user selects option 1, the user gets a list of available modules as the output. The system shows the Module Code, Module Name and the number of credit hours given to each module. This enables the student to view all courses available for them to take before having to sign up for them.

```
---------- Student Menu ----------
1. View Available Modules
2. Enroll in Module
3. Unroll from Module
4. View Grades
5. Access Attendance Record
6. Exit
Enter your choice: 2
Enter your Student ID: 110
Enroll in Module...
Enter Module Code: PHY101
Enrolled in module PHY101 successfully on 2025-01-19:44:50.
```

If the user selects option 2, they can register for a module by entering the student ID and the Module code. When the enrolment is affected, a message confirming this will be generated accompanied with the time stamp.

```
---------- Student Menu ----------
1. View Available Modules
2. Enroll in Module
3. Unroll from Module
4. View Grades
5. Access Attendance Record
6. Exit
Enter your choice: 3
Enter your Student ID: 101
Unroll from Module...
Enter Module Code: CS101
Unrolled in module CS101 successfully on 2025-01-19:47:12.
```

When the user chooses option 3, they can unroll from a module by typing in their Student ID and the Module Code.

```
---------- Student Menu ----------
1. View Available Modules
2. Enroll in Module
3. Unroll from Module
4. View Grades
5. Access Attendance Record
6. Exit
Enter your choice: 3
Enter your Student ID: 107
Unroll from Module...
Enter Module Code: CS101
No enrollment found for module CS101.
```

In the case that the student is not accorded to the specified module, the system gives a message that no enrolment was found. If the unrolling is complete, a message confirming this, along with the time stamp is displayed.

```
---------- Student Menu ----------
1. View Available Modules
2. Enroll in Module
3. Unroll from Module
4. View Grades
5. Access Attendance Record
6. Exit
Enter your choice: 4
Enter your Student ID: 109
Your Grades:
Module Code: ENG102, Grade: A
```

When the use chooses option 4, they can view their grades after entering their Student ID. This means the system will show module codes and the scores which has been obtained for each module.

```
---------- Student Menu ----------
1. View Available Modules
2. Enroll in Module
3. Unroll from Module
4. View Grades
5. Access Attendance Record
6. Exit
Enter your choice: 5
Enter your Student ID: 101
Enter Module Code: CS101
Attendance Records for Student ID 101 in Module CS101:
Total Classes: 1
Classes Attended: 1
Attendance Percentage: 100.00%
```

If the user chooses option 5, he will be able to check his attendance record of a certain module. When the student types of his Student ID and the Module Code the system will show the Total classes, number of classes attended and the attendance percentage for the stated module.

```
---------- Student Menu ----------
1. View Available Modules
2. Enroll in Module
3. Unroll from Module
4. View Grades
5. Access Attendance Record
6. Exit
Enter your choice: 6
Exiting Student Menu.
-------Welcome to University Management System-----------
Please enter Login credentials
UserName: "UMSLOGIN", Password: "12589"
Enter Username:
```

If the user selects option 6, they exit the Student Menu and return to the Main Menu. The system logs out the student from the current session and displays the login screen.

4.6 Registrar Role

```
---------- Main Menu ----------
1. Administrator
2. Lecturer
3. Student
4. Registrar
5. Accountant
6. Exit
Enter your choice: 4
---------- Registrar Menu ----------
1. Register New Student
2. Update Student Record
3. Manage Enrollments
4. Issue Transcript
5. Delete Student Record
6. View Student Information
7. Exit
Enter your choice:
```

Upon choosing number 5, the user will be shown the registrar menu. Here the user will have 7 options to choose from.

```
---------- Registrar Menu ----------
1. Register New Student
2. Update Student Record
3. Manage Enrollments
4. Issue Transcript
5. Delete Student Record
6. View Student Information
7. Exit
Enter your choice: 1
Register New Student
Enter Student ID: 148
Enter Student Name: MAHRUS SHAMSUL AHSAN
Enter Program Name: CALCULUS
student registered on : 2025-01-19:42:07
Student registered successfully.
```

Choosing option 1 will allow the user to register a new student. The user will have to enter the new student ID, name of the new student and the program name. If

added successfully, a message will be shown stating that the student has been registered successfully.

```
---------- Registrar Menu ----------
1. Register New Student
2. Update Student Record
3. Manage Enrollments
4. Issue Transcript
5. Delete Student Record
6. View Student Information
7. Exit
Enter your choice: 2
Update Student Record
Enter Student ID to update: 148
Enter Updated Student Name: mahrus ahsan
Enter Updated Program Name: grammer
student information updated on: 2025-01-19:56:53
Student record updated successfully.
```

Choosing option 2 will allow the user to update student record. The user will have to enter the student ID, updated student name and updated program name. Once successfully added a time stamp will be shown and a message will also be shown stating the student record was updated successfully.

```
---------- Registrar Menu ----------
1. Register New Student
2. Update Student Record
3. Manage Enrollments
4. Issue Transcript
5. Delete Student Record
6. View Student Information
7. Exit
Enter your choice: 3
Manage Enrollments
Enter Student ID: 148
Enter Module Code: cs101
Enter Action (Enroll/Unenroll): enroll
student enroll in cs101  on : 2025-01-19:36:45
Student  148  enrolled in module  cs101
```

Choosing option 3 will allow the user to manage the enrollment of students. The user will have to enter the student ID, module code, and whether to enroll or unenroll.

A time stamp and a message will be shown if the student was successfully enrolled or unenrolled.

```
---------- Registrar Menu ----------
1. Register New Student
2. Update Student Record
3. Manage Enrollments
4. Issue Transcript
5. Delete Student Record
6. View Student Information
7. Exit
Enter your choice: 4
Issue Transcript
Enter Student ID: 101
Transcript for Student ID: 101
Modules and Grades:
Transcript generated on : 2025-01-19:53:10
Module: {'CS101'}  Grade: {'A'}
```

Choosing option 4 will allow the user to issue transcript for a student. The user will have to enter the student ID. Then a transcript will be generated stating the student ID, and the grade of the student. A time stamp will also be generated.

```
---------- Registrar Menu ----------
1. Register New Student
2. Update Student Record
3. Manage Enrollments
4. Issue Transcript
5. Delete Student Record
6. View Student Information
7. Exit
Enter your choice: 5
Delete Student Record
Enter Student ID to delete: 148
student record deleted on : 2025-01-19:40:42
Student record deleted successfully.
```

Choosing option 5 will allow the user to delete student record. The user will have to enter the student ID. A time stamp will and generated and a message will be shown if the deletion was successful.

```
---------- Registrar Menu ----------
1. Register New Student
2. Update Student Record
3. Manage Enrollments
4. Issue Transcript
5. Delete Student Record
6. View Student Information
7. Exit
Enter your choice: 6
View Student Information
Enter Student ID to view: 101
student info viewed on : 2025-01-19:41:06
Student Details:
ID: 101, Name: mahrus, Program: grammer
```

Choosing option 6 will allow the user to view any student's information. The user will have to enter the student ID. The user will then be shown the student ID, name, and program the student is enrolled in. A time stamp will be generated.

```
---------- Registrar Menu ----------
1. Register New Student
2. Update Student Record
3. Manage Enrollments
4. Issue Transcript
5. Delete Student Record
6. View Student Information
7. Exit
Enter your choice: 7
Exiting Registrar Menu.
-------Welcome to University Management System-----------
Please enter Login credentials
UserName: "UMSLOGIN", Password: "12589"
Enter Username:
```

Choosing option 7 will allow the user to exit the registrar menu and return to the UMS login menu.

4.6 Accountant Role

```
---------- Main Menu ----------
1. Administrator
2. Lecturer
3. Student
4. Registrar
5. Accountant
6. Exit
Enter your choice: 5
---------- Accountant Menu ----------
1. Record Tuition Fees
2. View Outstanding Fees
3. Update Payment Records
4. Issue Fee Receipt
5. View Financial Summary
6. Exit
Enter your choice:
```

Upon entering number 5 to access the accountant feature, the user is presented with the accountant menu. The user can choose six options from this menu.

```
---------- Accountant Menu ----------
1. Record Tuition Fees
2. View Outstanding Fees
3. Update Payment Records
4. Issue Fee Receipt
5. View Financial Summary
6. Exit
Enter your choice: 7
Invalid choice, please try again.
---------- Accountant Menu ----------
1. Record Tuition Fees
2. View Outstanding Fees
3. Update Payment Records
4. Issue Fee Receipt
5. View Financial Summary
6. Exit
Enter your choice:
```

If the user enters an invalid choice, an error message will be shown stating that the choice is invalid, and the user will be asked to enter their choice again.

```
---------- Accountant Menu ----------
1. Record Tuition Fees
2. View Outstanding Fees
3. Update Payment Records
4. Issue Fee Receipt
5. View Financial Summary
6. Exit
Enter your choice: 1
Record Tuition Fees
Enter Student ID: 101
Enter Amount Paid: 5000
Enter Date of Payment (YYYY-MM-DD): 2025-01-19
Tuition fee payment recorded successfully.
```

Choosing option 1 will allow the user to record tuition fees. The user will be asked to enter the student ID, the amount paid, and the date of payment. Once the user has entered the details, they will get a message if the record has been added successfully.

```
---------- Accountant Menu ----------
1. Record Tuition Fees
2. View Outstanding Fees
3. Update Payment Records
4. Issue Fee Receipt
5. View Financial Summary
6. Exit
Enter your choice: 1
Record Tuition Fees
Enter Student ID: 101
Enter Amount Paid: 2000
Enter Date of Payment (YYYY-MM-DD): 19-2025-01
Invalid date format. Please use YYYY-MM-DD.
Enter Date of Payment (YYYY-MM-DD):
```

If the user enters the date in a wrong format, an error message will be shown, and the user will be asked to enter the date again.

```
Enter your choice: 2
View Outstanding Fees
Reports generated on: 2025-01-19 13:04:02
Outstanding Fees:
Student ID: 101, Outstanding Fees: 5000
Student ID: 102, Outstanding Fees: 3500
Student ID: 103, Outstanding Fees: 4000
Student ID: 104, Outstanding Fees: 4500
Student ID: 105, Outstanding Fees: 3200
Student ID: 106, Outstanding Fees: 3000
Student ID: 107, Outstanding Fees: 3500
Student ID: 108, Outstanding Fees: 5000
Student ID: 109, Outstanding Fees: 6000
Student ID: 110, Outstanding Fees: 4500
Student ID: 999, Outstanding Fees: 9999
Student ID: 998, Outstanding Fees: 9998
Student ID: 101, Outstanding Fees: 5000
```

Choosing option 2 will allow the user to view all outstanding fees. They will be able to see the student ID and the amount of money outstanding. The time stamp will be shown as well.

```
---------- Accountant Menu ----------
1. Record Tuition Fees
2. View Outstanding Fees
3. Update Payment Records
4. Issue Fee Receipt
5. View Financial Summary
6. Exit
Enter your choice: 3
Update Payment Records
Enter Student ID: 101
Enter Updated Amount Paid: 200
Enter Updated Payment Date (YYYY-MM-DD): 2025-01-19
Payment record updated successfully.
```

Choosing option 3 will allow the user to update any payment record. The user will be asked to enter the student ID, updated amount paid, and the payment date. If

updated successfully, the user will be shown a message stating the record has been updated successfully.

```
---------- Accountant Menu ----------
1. Record Tuition Fees
2. View Outstanding Fees
3. Update Payment Records
4. Issue Fee Receipt
5. View Financial Summary
6. Exit
Enter your choice: 3
Update Payment Records
Enter Student ID: 101
Enter Updated Amount Paid: -899
Invalid input. Please enter a numeric value.
Enter Updated Amount Paid:
```

If the user enters an invalid input for the amount paid an error message will be shown and the user will be asked to enter the amount again.

```
---------- Accountant Menu ----------
1. Record Tuition Fees
2. View Outstanding Fees
3. Update Payment Records
4. Issue Fee Receipt
5. View Financial Summary
6. Exit
Enter your choice: 4
Issue Fee Receipt
Enter Student ID: 101
Fee Receipt:
Student ID: 101
Amount Paid: 200
Payment Date: 2025-01-19
```

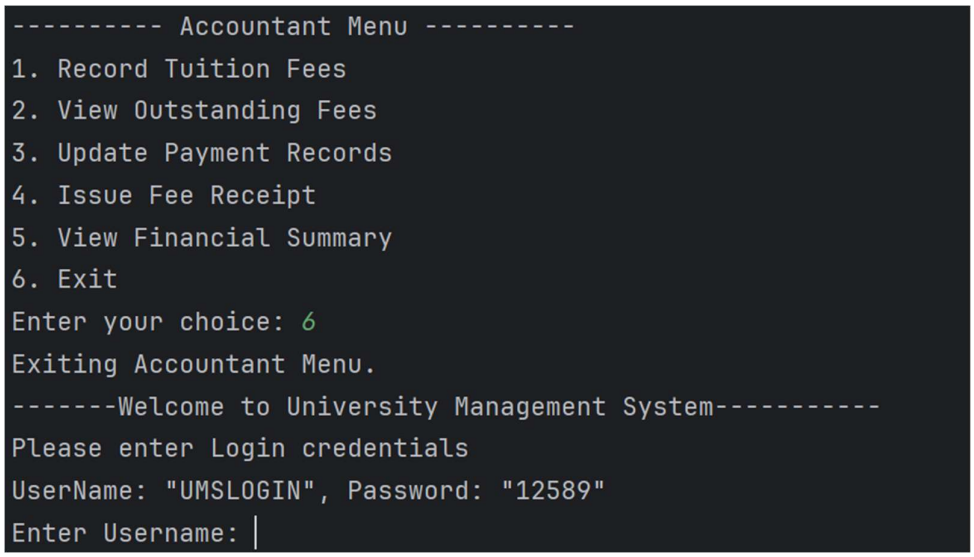
Choosing option 4 will allow the user to issue a fee receipt. The user will be asked to enter the student ID. The receipt will then be issued stating the student ID, the amount paid and the date of payment.

```
---------- Accountant Menu ----------
1. Record Tuition Fees
2. View Outstanding Fees
3. Update Payment Records
4. Issue Fee Receipt
5. View Financial Summary
6. Exit
Enter your choice: 4
Issue Fee Receipt
Enter Student ID: 123
No fee record found for the given Student ID.
```

If the user enters the wrong student ID, an error message will be shown stating that no fee record found for the given student ID.

```
---------- Accountant Menu ----------
1. Record Tuition Fees
2. View Outstanding Fees
3. Update Payment Records
4. Issue Fee Receipt
5. View Financial Summary
6. Exit
Enter your choice: 5
View Financial Summary
Reports generated on: 2025-01-19 13:15:40
Total Fees Collected: 57597.00
```

Choosing option 5 will allow the user to view the financial summary. The user will be shown the time stamp of the report and the total fees collected.



Choosing option 6 will allow the user to exit the accountant menu and go back to the login menu. The user will be required to login again in order to choose any of the features.

CONCLUSION

This University Management System (UMS) is a helpful solution for managing nearly all the operations of the university and has more specific operations for administrators, lecturers, students, registrars, and accountants to perform in their various roles. Developed with Python, the system uses such basic programming activities as functions, files, and input/output validation wherein no additional libraries are used for data processing. Combined with the isolated deployments, this means that each feature is separated into its own functions, which provides the beginning for a clean, structured, readable, and maintainable codebase. This way, roles reduce security and usability risks and allow users to see only the functionality that matches their positions. Moreover, error checking and input formatting reduce the likelihood of errors and make data processing more accurate. However, as the system advances and works with high volumes of data, the use of simple databases as text files for data storage (CSV) will be an issue.

Changing to the database-based solution would dramatically improve both scalability and performance. But as basic for current implementation, making more improvement could enhance the usability of the system, more function such as logging, detailed report and GUI need to be added on the system. Thus, UMS system has confirmed a reasonable approach to developing basic programming concepts and offers a stable option for managing university activities.

This is especially ideal for small and medium educational facilities; and there exist quite pronounced possibilities for the subsequent optimization with respect to the growth in data storing capacity, response speed, as well as the degree of easiness for users.

REFERENCES

GeeksforGeeks. (2024, December 8). *Python Try Except*. Retrieved from GeeksforGeeks: https://www.geeksforgeeks.org/python-try-except/

GeeksforGeeks. (2024, August 9). *Python3 – if , if..else, Nested if, if-elif statements*. Retrieved from GeeksforGeeks: https://www.geeksforgeeks.org/python3-if-if-else-nested-if-if-elif-statements/?ref=outind

Programiz. (n.d.). *Python File Operation*. Retrieved from Programiz: https://www.programiz.com/python-programming/file-operation

W3Schools. (n.d.). *Python Lists*. Retrieved from W3Schools: https://www.w3schools.com/python/python_lists.asp

APPENDIX A: TEXT FILES

Lecturer Text file

lecturers.txt

```
201,Dr. Alan Turing,Computer Science,2022-04-17
202,Dr. Marie Curie,Physics,2023-06-05
203,Dr. Katherine Johnson,Mathematics,2023-07-22
204,Dr. Nikola Tesla,Engineering,2023-11-07
205,Dr. Ada Lovelace,Computer Science,2022-08-06
206,Dr. Rosalind Franklin,Physics,2024-11-13
207,Dr. John Nash,Mathematics,2023-08-07
208,Dr. Thomas Edison,Engineering, 2024-12-14
209,Dr. Grace Hopper,Computer Science, 2022-12-28
210,Dr. Stephen Hawking,Physics, 2022-08-31
```

Modules Text file

modules.txt

```
CS101,Programming Basics,3,201
CS102,Data Structures,4,205
CS103,Algorithms,4,201
PHY101,Classical Mechanics,3,202
PHY102,Quantum Physics,4,206
MAT101,Calculus,3,203
MAT102,Linear Algebra,4,207
ENG101,Literature,3,204
ENG102,Grammar,4,208
CS104,Artificial Intelligence,5,209
```

Students Text file

students.txt

```
101,John Doe,Computer Science,2022-03-14
102,Jane Smith,Mathematics,2022-11-27
103,Alice Johnson,Physics,2023-05-06
104,Bob Brown,Engineering,2023-09-19
105,Charlie Davis,Computer Science,2022-12-02
106,Daisy Evans,Mathematics,2024-04-22
107,Edward Green,Mathematics,2023-07-30
108,Fiona Hall,Engineering,2024-10-15
109,George White,Computer Science,2024-02-08
110,Hannah Black,Mathematics,2023-11-11
```

Fees Text file

fees.txt

```
101,5000,2024-01-01
102,3500,2024-02-09
103,4000,2024-01-03
104,4500,2024-01-04
105,3200,2024-01-05
106,3000,2024-01-06
107,3500,2024-01-07
108,5000,2024-01-08
109,6000,2024-01-09
110,4500,2024-01-10
999,9999,2025-01-16
998,9998,2025-01-16
```

Grades Text file

grades.txt

```
CS101,101,A
CS101,102,B
CS102,103,A
CS103,104,C
PHY101,105,B
PHY102,106,A
MAT101,107,C
ENG101,108,B
ENG102,109,A
CS104,110,A
CS101,101,A
CS101,103,B
```

Security Text file

Security.txt

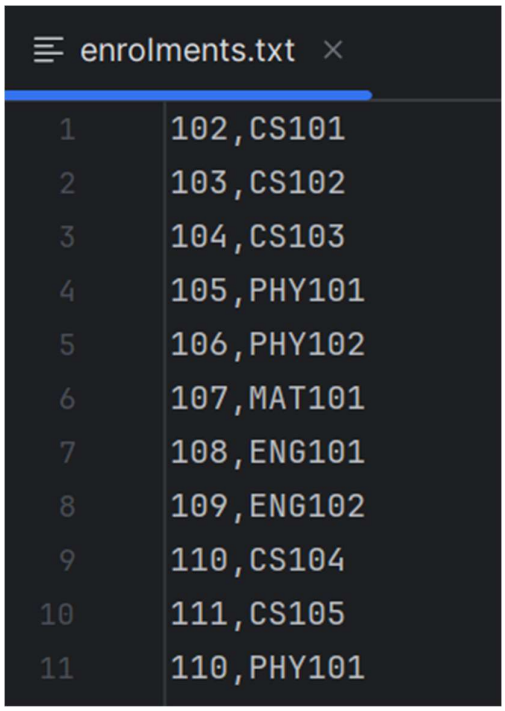
```
UMSLOGIN,12589

```

Attendance Text file

attendance.txt

```
1   CS101,101,Present
2   CS101,102,Absent
3   CS102,103,Present
4   CS103,104,Present
5   PHY101,105,Absent
6   PHY102,106,Present
7   MAT101,107,Present
8   ENG101,108,Absent
9   ENG102,109,Present
10  CS104,110,Present
11  MAT101,106,Present
12  PHY102,105,Present
13  CT103,!02,Absent
```

Enrolments Text file



APPENDIX B: WORKLOAD MATRIX

Task/Activity	Description	Assigned Member	Percentage
Requirement Analysis	Define project objectives, scope, and requirements.	Ahmed Mirahusain Alvi TP084807	20%
		ANDREW WIJAYA TP079319	20%
		Mahrus Shamsul AhsanTP085562	20%
		Abdallah Mohamed Mahmoud Mohamed Mahmoud TP085097	20%
		Mohammed Yousef Mohammed Mohammed TP085042	20%
System Design	Flowcharts / Pseudocode	Ahmed Mirahusain Alvi TP084807	20%
		ANDREW WIJAYA TP079319	20%
		Mahrus Shamsul AhsanTP085562	20%
		Abdallah Mohamed Mahmoud Mohamed Mahmoud TP085097	20%
		Mohammed Yousef Mohammed Mohammed TP085042	20%
Backend Development	Implement core functional	Ahmed Mirahusain Alvi TP084807 (Administrator)	20%
		ANDREW WIJAYA TP079319 (Register)	20%

		Mahrus Shamsul AhsanTP085562 (Accountant)	20%
		Abdallah Mohamed Mahmoud Mohamed Mahmoud TP085097 (Lecturer)	20%
		Mohammed Yousef Mohammed Mohammed TP085042 (Student)	20%
Documentation	Table of contents	Ahmed Mirahusain Alvi TP084807	20%
	Acknowledgement	ANDREW WIJAYA TP079319	20%
	Abstract	Mahrus Shamsul AhsanTP085562	20%
	Introduction and assumptions of your system developed	Abdallah Mohamed Mahmoud Mohamed Mahmoud TP085097	20%
	Explanation of programming concepts applied with sample of source code based-on the system developed.	Mohammed Yousef Mohammed Mohammed TP085042	20%
	Screenshots of sample input/output with explanation		
	Conclusion		