TUTORING SYSTEM

NAME: KEERTHANA RAJESH
ROLL NO: 46
COURSE NAME: CSE
DATE: 17-07-2024

INTRODUCTION

The tutoring system developed in this micro project is a menu-driven application implemented in the C programming language. The primary functionalities of the system include managing users, scheduling tutoring sessions, tracking student progress, and conducting assessments. By using structures to represent different entities and binary files for data storage, the system ensures efficient data management and retrieval.

The primary objective of this microproject is to develop a comprehensive tutoring system using the C programming language. The system needs to address the following challenges:

User Management:

- Create a mechanism to add, store, and display users including Admins, Tutors, and Students.
- Ensure the system can differentiate between different roles and manage data accordingly.

Session Scheduling:

- Provide functionality for scheduling tutoring sessions, including details such as date, time, tutor, student, and subject.
- Maintain a record of all scheduled sessions.

Progress Tracking:

- Implement a method to track and update the progress of students in their respective tutoring sessions.
- Ensure accurate and updating of progress records.

Assessment Management:

- Develop functionality to record and manage assessment scores for students in various sessions.
- Display recorded assessment scores in a clear and organized manner.

SYSTEM REQUIREMENTS

- Operating System: Windows, macOS, or Linux
- Compiler: GCC (GNU Compiler Collection) or any C compiler
- Code: Text editor

Design and Development

The tutoring system is designed as a menu-driven application with several functionalities:

1. Adding users (Admin, Tutor, Student)
2. Displaying user information
3. Scheduling tutoring sessions
4. Displaying scheduled sessions
5. Updating student progress
6. Displaying student progress
7. Conducting assessments
8. Displaying assessment results

PSEUDOCODE:

Main Function:

```
BEGIN
  DO
    DISPLAY "Tutoring System Menu"
    DISPLAY "1. Add User"
    DISPLAY "2. Display Users"
    DISPLAY "3. Schedule Session"
    DISPLAY "4. Display Sessions"
    DISPLAY "5. Update Progress"
    DISPLAY "6. Display Progress"
    DISPLAY "7. Conduct Assessment"
    DISPLAY "8. Display Assessments"
    DISPLAY "0. Exit"
    PROMPT "Enter your choice: "
```

```
        GET user choice
        SWITCH (user choice)
            CASE 1: CALL addUser()
            CASE 2: CALL displayUsers()
            CASE 3: CALL scheduleSession()
            CASE 4: CALL displaySessions()
            CASE 5: CALL updateProgress()
            CASE 6: CALL displayProgress()
            CASE 7: CALL conductAssessment()
            CASE 8: CALL displayAssessments()
            CASE 0: PRINT "Exiting..."
            DEFAULT: PRINT "Invalid choice. Please try again."
    WHILE (user choice != 0)
END
```

Add User Function:

```
FUNCTION addUser()
    OPEN "users.dat" IN append mode AS file
    IF file NOT opened
        PRINT "Unable to open file"
        RETURN
DECLARE User u
    PROMPT "Enter User ID: "
    GET u.id
    PROMPT "Enter Name: "
    GET u.name
    PROMPT "Enter Role (Admin/Tutor/Student): "
    GET u.role
WRITE u TO file
    CLOSE file
    PRINT "User added successfully!"
END FUNCTION
```

Display Users Function:

```
FUNCTION displayUsers()
    OPEN "users.dat" IN read mode AS file
    IF file NOT opened
```

```
        PRINT "Unable to open file"
        RETURN

    DECLARE User u
    WHILE READ u FROM file
        PRINT "ID: " u.id ", Name: " u.name ", Role: " u.role
    END WHILE
    CLOSE file
END FUNCTION
```

Schedule Session Function:

```
FUNCTION scheduleSession()
    OPEN "sessions.dat" IN append mode AS file
    IF file NOT opened
        PRINT "Unable to open file"
        RETURN

    DECLARE Session s
    PROMPT "Enter Session ID: "
    GET s.id
    PROMPT "Enter Tutor ID: "
    GET s.tutor_id
    PROMPT "Enter Student ID: "
    GET s.student_id
    PROMPT "Enter Date (YYYY-MM-DD): "
    GET s.date
    PROMPT "Enter Time (HH: MM): "
    GET s.time
    PROMPT "Enter Subject: "
    GET s.subject

    WRITE s TO file
    CLOSE file

    PRINT "Session scheduled successfully!"
END FUNCTION
```

Display Sessions Function

```
FUNCTION displaySessions()
    OPEN "sessions.dat" IN read mode AS file
    IF file NOT opened
        PRINT "Unable to open file"
        RETURN

    DECLARE Session s
    WHILE READ s FROM file
        PRINT "ID: " s.id ", Tutor ID: " s.tutor_id ", Student ID: " s.student_id ", Date: " s.date ", Time: " s.time ", Subject: " s.subject
    END WHILE
    CLOSE file
END FUNCTION
```

Update Progress Function

```
FUNCTION updateProgress()
    OPEN "progress.dat" IN read/write mode AS file
    IF file NOT opened
        PRINT "Unable to open file"
        RETURN

    DECLARE Progress p
    DECLARE found = 0
    PROMPT "Enter Student ID: "
    GET student_id
    PROMPT "Enter Session ID: "
    GET session_id
    PROMPT "Enter new Progress (0.0 - 100.0): "
    GET new_progress

    WHILE READ p FROM file
        IF p.student_id == student_id AND p.session_id == session_id
            p.progress = new_progress
            MOVE file pointer back to the position of the last read record
            WRITE p TO file
            found = 1
```

```
            PRINT "Progress updated successfully!"
            BREAK
        END IF
    END WHILE

    IF NOT found
        PRINT "Record not found."

    CLOSE file
END FUNCTION
```

Display Progress Function

```
FUNCTION displayProgress()
    OPEN "progress.dat" IN read mode AS file
    IF file NOT opened
        PRINT "Unable to open file"
        RETURN

    DECLARE Progress p
    WHILE READ p FROM file
        PRINT "Student ID: " p.student_id ", Session ID: " p.session_id ", Progress: " p.progress
    END WHILE
    CLOSE file
END FUNCTION
```

Conduct Assessment Function

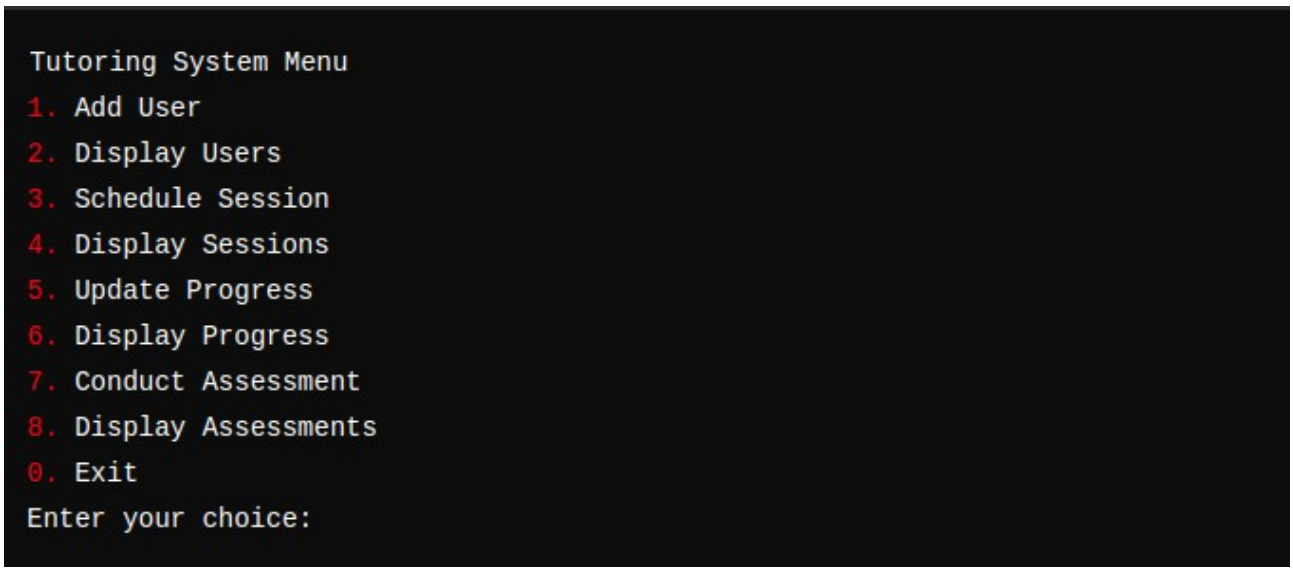
```
FUNCTION conductAssessment()
    OPEN "assessments.dat" IN append mode AS file
    IF file NOT opened
        PRINT "Unable to open file"
        RETURN

    DECLARE Assessment a
    PROMPT "Enter Student ID: "
    GET a.student_id
    PROMPT "Enter Session ID: "
    GET a.session_id
```

```
    PROMPT "Enter Assessment Score: "
    GET a.score

    WRITE a TO file
    CLOSE file

    PRINT "Assessment recorded successfully!"
END FUNCTION
```

Display Assessments Function

```
FUNCTION displayAssessments()
    OPEN "assessments.dat" IN read mode AS file
    IF file NOT opened
        PRINT "Unable to open file"
        RETURN

    DECLARE Assessment a
    WHILE READ a FROM file
        PRINT "Student ID: " a.student_id ", Session ID: " a.session_id ", Score: " a.score
    END WHILE
    CLOSE file
END FUNCTION
```

Testing and Results



1.Add User

```
Enter User ID: 1
Enter Name: Alice
Enter Role (Admin/Tutor/Student): Tutor
User added successfully!
```

2.Display Users

```
ID: 1, Name: Alice, Role: Tutor
```

3.Schedule Session

```
Enter Session ID: 1
Enter Tutor ID: 1
Enter Student ID: 2
Enter Date (YYYY-MM-DD): 2024-07-12
Enter Time (HH:MM): 10:00
Enter Subject: Math
Session scheduled successfully!
```

4.Display Sessions

```
ID: 1, Tutor ID: 1, Student ID: 2, Date: 2024-07-12, Time: 10:00, Subject: Math
```

5.Update Progress

```
Enter Student ID: 2
Enter Session ID: 1
Enter new Progress (0.0 - 100.0): 75.5
Progress updated successfully!
```

6.Display Progress

```
Student ID: 2, Session ID: 1, Progress: 75.50
```

7.Conduct Assessment

```
Enter Student ID: 2
Enter Session ID: 1
Enter Assessment Score: 88.0
Assessment recorded successfully!
```

8.Display Assessments

```
Student ID: 2, Session ID: 1, Score: 88.00
```

Conclusion

The tutoring system developed in C programming effectively addresses the core requirements for managing tutoring sessions, scheduling classes, monitoring student progress, and conducting assessments. The system includes functionalities to add users (Admins, Tutors, and Students), schedule and display sessions, update and display student progress, and conduct and display assessments. The modular design allows for easy understanding and maintenance, with each functionality encapsulated in its function.

Throughout the project, various features were implemented and tested to ensure the program runs correctly. User data is stored in files to maintain persistence across sessions, and the program offers a simple command-line interface for interaction. The overall design is robust, providing a good foundation for further development.

In future,

we can elaborate the project into,

User Authentication:

- Implement a login system with username and password to authenticate users before allowing access to the system.

Notification System:

- Implement an email or SMS notification system to remind students and tutors of upcoming sessions.

Detailed Progress Tracking:

- Include more detailed metrics for progress tracking, such as attendance records, homework completion, and participation scores.

Comprehensive Assessment Module:

- Develop a more comprehensive assessment module that supports various types of assessments, including quizzes, assignments, and exams.

User Management Enhancements:

- Add functionality for editing and deleting users, along with role-based access control to ensure that only authorized users can perform certain actions.

VIDEO REFERENCES:

1. freeCodeCamp.org