Abdel Rahman El Kurdi

Toufic Nouwayhed

Flip Point Report

EECE 437

User\_system: It was supposed to deal with, in addition to signing in, signing up of new users. This wasn't implemented after we decided that all student data is found in the data storage, making it useless to do a signup feature.

Dummy data\_storage is made up of ArrayLists that store dummy information as strings. I had to do this data\_storage because of the problems we faced during the development of the database.

Dummy gui file represents the potential of the final gui. The main functionality that works includes adding courses in student's schedule.

Preregistration file only includes the constructor and main function to start the program. The program starts by instantiating the user\_system and showing the gui.

User file was supposed to be the main user file and teachers, admins and students inherit from it. It turns out that they have less in common than expected, rendering the user class empty.

Students have methods to add and remove courses, in addition to request capacity from a teacher.

Teachers have somehow similar functions to those of students. In addition, they can open capacities depending on room size.

Admins can define a certain threshold in order to accept the request for a new course. They have the method to set a course's room.

Database:

The database is found as a file uploaded along with the code.

The database is made up of 8 tables:

1. users (UserID, UserPassword)

2. teachers (UserID, TeacherName)

3. students (UserID, StudentName)

4. administrators (UserID, AdministratorName)

5. rooms (RoomID, Capacity)

6. course (CourseID, CourseTime, Capacity, UserID(teachers), RoomID(rooms))

7. courseuser (UserID(users), CourseID(course)) //used for the schedule

8. messages (UserID(users), MessageTimestamp, MessageBody)

As for the code, the implemented functions are all the required interface as defined before. The function retreiveCourse take as arguments a string the course ID, and simply checks whether this course exists or not. The function storeCourse takes in as arguments as string the course ID, the time, as integer the capacity and the teacher id, and finally as string the room. This function simply takes these arguments and stores the course with these attributes with it.

The function RetreiveSchedue takes in as arguments an integer a user ID (teacher or student) and returns an array list consisting of all the courses that this user is teaching or taking, respectively. The function StoreCourseForUser takes in as arguments both the user ID (teacher or student) and the course ID this user is teaching or taking, respectively. This function takes the arguments and store them as a pair of composite keys user ID and course ID.

The function retrieveMessages takes in as arguments an integer a user ID and returns the most recent message this user has sent. The function StoreMessage takes in as arguments a String containing the message, an integer user ID, and a string containing the time and date the message was sent. The function inserts these information in the message table with user ID and timestamp as pair of composite keys and the message as an attribute.