Simple Software Development

Overview

You will create 2 different functionality interface one is Patient and the other Admin/Receptionist Interface, they will both be separate git project. One is Patient Project and the other is Admin/Receptionist.

GUI: You will implement a standalone Java application (e.g. with a JFrame UI as shown in the Eclipse demo or something similar). The appearance of the interface is not important as long as the required functionality is provided. That is, you are not expected to implement a fancy looking interface. Also see the Oracle tutorial about JFrame: https://docs.oracle.com/javase/tutorial/uiswing/components/frame.html

Database: You need to have a working database that interacts with your program to store and retrieve relevant entities (e.g. doctors, patients, bookings, messages). MySQL database from an Eclipse project. You do not need to host the database remotely, simple MySQL tutorial: https://www.vogella.com/tutorials/MySQLJava/article.html

You should invite us to your GIT so we can see progress.

Functionality for the "Patient" interface:

- (Authentication) The system should allow a user to log in with their username and password, and log out. When the user logs in, the system should show the user's messages on the welcome screen.
- (Authorisation checks) The system should log all access from a user, i.e. who accessed what functionality and when.
- The system should allow a new user to register as a patient, choose a doctor from the list of all doctors. The system should then send confirmation messages to the patient and the doctor.
- The system should allow a patient to change their doctor using the list of all doctors. The system should then send confirmation messages to the patient and the doctor.
- The system should allow a patient to arrange a booking with their doctor by entering the date and time. If the doctor is not available for the chosen date and time, the system should warn the patient. Otherwise, the system should then send confirmation messages to the patient and the doctor.
- The system should allow a patient to view their bookings by entering a month and year.
- The system should allow a patient to reschedule a booking with their doctor by entering the date and time. If the doctor is not available for the chosen date and time, the system should warn the patient. Otherwise, the system should then send confirmation messages to the patient and the doctor.
- The system should allow a patient to view the visit details regarding a past booking, for which the doctor provided visit details and prescriptions.

Functionality for the "Admin/Receptionist" interface:

- (Authentication) The system should allow a user to log in with their username and password, and log out. When the user logs in, the system should show the user's messages on the welcome screen.
- (Authorisation checks) The system should log all access from a user, i.e. who accessed what functionality and when.
- The system should allow an admin/receptionist to enter a new doctor by providing their details, e.g. name, phone number, background. The system should then send a confirmation message to the doctor.
- The system should allow an admin/receptionist to enter a new patient by providing their details, e.g. name, phone number, and assign a doctor to the patient using the list of all doctors. The system should then send confirmation messages to the patient and the doctor.
- The system should allow an admin/receptionist to change a patient's doctor using the lists of all patients and doctors. The system should then send confirmation messages to the patient and the doctor.
- The system should allow an admin/receptionist to arrange a booking for a patient (selected from the list of all patients) with their doctor by entering the date and time. If the doctor is not available for the chosen date and time, the system should warn the admin/receptionist. Otherwise, the system should then send confirmation messages to the patient and the doctor.
- The system should allow an admin/receptionist to view bookings by selecting a doctor from the list of all doctors, by selecting a patient from the list of all patients, or by entering a month and year.

- The system should allow an admin/receptionist to remove or reschedule a booking. If the doctor is not available for the chosen date and time (in case of a rescheduling), the system should warn the admin/receptionist. Otherwise, the system should then send confirmation messages to the patient and the doctor.

Deliveries (This is for both Patient and Admin/Receptionist 2 separate git projects)

Part 1 (Due Monday, 11 March 2024 14:00 GMT Time):

- Implement authentication and most important functionality.
- Code: properly indented and commented, complying with naming conventions and quality standards, submitted in the form of a project that could be imported into a standard IDE such as Eclipse
- Test cases: JUnit test classes and results (showing tests passed), including a brief two-page test specifications document explaining what tests are performed for each class/method.

.

- Group organisation plan: brief one-page document describing who does what (e.g. leads for each feature, DB design, and tests) and plan for collaboration each week

Part 2 (Due Sunday, 17 March 2024 14:00 GMT Time):

- Implement 4 more functionalities (label/comment it as functionality – 1 name functionality)
- Code: properly indented and commented, complying with naming conventions and quality standards, submitted in the form of a project that could be imported into a standard IDE such as Eclipse
- Test cases: JUnit test classes and results (showing tests passed), including a brief two-page test specifications document explaining what tests are performed for each class/method
- Implement authorisation checks (all groups); and one other functionality for three people groups / two other functionality for four people groups / three other functionality for five people groups
- Database design document: brief two-page document describing tables, columns, keys, and justification of design choices (you can include diagrams)
- Quality assurance: brief two-page document showing examples of code reviews, refactoring, issue tracking within the group
- Video demonstration: clearly showing how each feature works and how the database is manipulated as a result (e.g. when a new patient is registered, show how the database is updated), about one minute for each feature.