CS 460W – Software Development
Project Deliverable #5

Team Members: Alisa Vongsarasinh, Benjamin Bahadory, Daniel Baqaeen

Unit Testing:

Test-case identifier	Login
Test location	final_project\Login\src\main\java\com\example\login\LoginController.java
Feature to be tested	Users can log in and be routed to their specific UI
Feature Pass/Fail Criteria	Users can log in and be routed to their specific UI with little delay.
Means of Control	1. The start() method is called via a test driver LoginMain (contained in the same directory as the Login test).
Data	2. User will choose which type of user they are and input their credentials. 3. If credentials are in database, user will be routed to their specific UI where they will be able to perform their tasks.
Test Procedure	User will select which type of user they are (Registrar, Nurse, Doctor, Billing, Patient), enter their credentials and then click the login button.
Special requirements	The test stub HospitalUI is needed for the test execution.

Test-case identifier	Nurse
Test location	final_project\Login\src\main\java\com\example\login\NurseController.java
Feature to be tested	Nurse can perform their tasks (e.g., view and edit patient records).
Feature Pass/Fail Criteria	The test passes if the nurse can search for a specific patient and receive information about them from the database that fills in the text fields. Furthermore, the nurse should be able to edit specific information (e.g.,

	height, weight, assigned physician, body temperature, blood pressure, nights stayed, nurse notes) which updates in the database.
Means of Control	1. The start() method is called via a test driver NurseMain (contained in the same directory as the Nurse test).
Data	2. Nurse will input patientID into search bar. 3. If patientID is found in database, text fields will be filled with information from the database that was stored in regards to the patient. 4. Nurse will be prompted to edit certain text fields (as mentioned in the 'Feature Pass/Fail Criteria' section).
Test Procedure	The test is started by first having the nurse log in with their credentials. Once they are logged in, their UI should appear which they will then click on the View/Edit Patient Record tab. Then, once that tab loads, they will enter the patient's ID in the search bar which will fill in the text fields with information from the hospital database. The nurse can then edit certain information that they have access to and then click on the update button which saves and updates the patient's information in the database with little to no delay.
Special requirements	The test stub HospitalUI is needed for the test execution as well as a nurse being logged in.

Test-case identifier	Doctor
Test location	final_project\Login\src\main\java\com\example\login\DoctorController.java
Feature to be tested	Doctor can perform their tasks (e.g., view and edit patient record, write discharge information).
Feature Pass/Fail Criteria	The test passes if the doctor can search for a specific patient and receive information about them from the database that fills in the text fields. Furthermore, the doctor should be able to edit specific information (e.g., tests, doctor notes, diagnosis, and medication) which updates in the database. The doctor should also be able to submit discharge information which the patient can view.

Means of Control	1. The start() method is called via a test driver DoctorMain (contained in the same directory as the Doctor test).
Data	2. Doctor inputs patientID in search bar. 3. If patientID is stored in database, information saved along with it is used to fill in record. 4. Doctor will be prompted to choose tests that they have administered, add notes, and choose a diagnosis. 5. Doctor will repeat the process when the Write Discharge Info tab is clicked.
Test Procedure	The test is started by first having the doctor log in with their credentials. Once they are logged in, their UI should appear which they will then click on the View/Edit Patient Record tab. Then, once the tab loads, they will enter the patient's ID in the search bar which will fill in the text fields with information from the hospital database. The doctor can then edit certain information that they have access to and then click on the update button which saves and updates the patient's information in the database with little to no delay. The doctor will then click on the Write Discharge Info tab and once it loads, will input a patient ID into the search bar to search for specific patient. If patient found, information regarding tests, doctor notes, diagnosis, and medication will appear, prompting doctor to make any updates. Once/if updates are made, submit button will be clicked which updates the information in the database.
Special requirements	The test stub HospitalUI is needed for the test execution as well as a doctor being logged in.

Test-case identifier	Patient
Test location	final_project\Login\src\main\java\com\example\login\patientController.java
Feature to be tested	Patients can view their specified personal information and nothing else.

Feature Pass/Fail Criteria	Fail criteria: Patients cannot access all of their information, or can access more than they are supposed to. An unhandled error occurs. The program crashes. Pass criteria: All data is displayed as it is stored in the database.
Means of Control	The start() method is called via the driver class HospitalUI, and the user logs in as a valid patient from the database
Data	Patient info is all found in patient_record in the database, and it must be displayed at the appropriate times.
Test Procedure	From the welcome page, the Information tab is selected. All data in the use cases for the patient methods are visible and correct as it has been edited in the database. No extra information is displayed. The patient may not edit any of the information. From the information tab we navigated to the View Discharge Info tab and clicked the View Discharge Info button. The diagnosis, medication, tests provided, and doctor notes all display as they are in the database. No extra information is displayed. Clicking the button again has no effect, as intended. From the View Discharge Info tab, we navigated to the View Bill tab. Clicking on View Bill filled the empty boxes with the services and their respective costs, correct as stored in the database. The multiplication and addition is always correct as there is nothing new to input, only data to read from the database. All tabs can be visited at any time, in any order. The logout button on the Home tab functions correctly and reroutes to the main Login page.
Special requirements	The test stub HospitalUI is needed for the test execution.

Test-case identifier	Registrar
Test location	final_project\Login\src\main\java\com\example\login\RegistrarController.java
Feature to be tested	Registrars can input all necessary data into database using the provided GUI features.

Feature Pass/Fail Criteria	Fail criteria: Data input does not enter database, enters incorrectly, or overwrites the wrong data. Pass criteria: All data is passed into the database on submission. Error methods explain every improper input.
Means of Control	The start() method is called via the driver class HospitalUI, and the user logs in as a valid registrar from the database
Data	Text entered into text boxes will become data in tables in the mySQL database.
Test Procedure	The home page is accessed by logging in with valid registrar credentials. From the home tab, we navigated to the Create Patient Record page. We input data into ever box and clicked Create. All data appeared inside the correct table in the database, and in the correct columns. We navigated to the View/Edit Patient Record tab, entered a patient ID and clicked the Search button. All the data from the columns in the patient_record table appeared in their corresponding text boxes. All error messages worked properly: Invalid inputs in the Create Patient Record tab and invalid IDs in the search tab. All pages can be navigated to and from in any order. The logout button returns the user to the login page.
Special requirements	The test stub HospitalUI is needed for the test execution.

Test-case identifier	Patient
Test location	final_project\Login\src\main\java\com\example\login\patientController.java
Feature to be tested	Patients can view their specified personal information and nothing else.
Feature Pass/Fail Criteria	Fail criteria: Patients cannot access all of their information, or can access more than they are supposed to. An unhandled error occurs. The program crashes.

	Pass criteria: All data is displayed as it is stored in the database.
Means of Control	The start() method is called via the driver class HospitalUI, and the user logs in as a valid patient from the database
Data	Patient info is all found in patient_record in the database, and it must be displayed at the appropriate times.
Test Procedure	From the welcome page, the Information tab is selected. All data in the use cases for the patient methods are visible and correct as it has been edited in the database. No extra information is displayed. The patient may not edit any of the information. From the information tab we navigated to the View Discharge Info tab and clicked the View Discharge Info button. The diagnosis, medication, tests provided, and doctor notes all display as they are in the database. No extra information is displayed. Clicking the button again has no effect, as intended. From the View Discharge Info tab, we navigated to the View Bill tab. Clicking on View Bill filled the empty boxes with the services and their respective costs, correct as stored in the database. The multiplication and addition is always correct as there is nothing new to input, only data to read from the database. All tabs can be visited at any time, in any order. The logout button on the Home tab functions correctly and reroutes to the main Login page.
Special requirements	The test stub HospitalUI is needed for the test execution.

Test-case identifier	Billing
Test location	final_project\Login\src\main\java\com\example\login\billingController.java
Feature to be tested	Billing agent can access all services provided to each patient in addition to the number of nights they stayed in hospice. The total bill is automatically calculated. Clicking submit stores the value in the patient_record in the database.

Feature Pass/Fail Criteria	Fail criteria: Calculations are incorrect, or the bill is incorrectly stored in the database, or it is not stored at all. Pass criteria: Calculations are correct and enter the database as they are shown on the page.
Means of Control	The start() method is called via the driver class HospitalUI, and the user logs in as a valid billing agent from the user_accounts table in the database.
Data	Patient info is all found in patient_record in the database, and it must be displayed at the appropriate times. The total bill must be submitter to patient_record upon clicking Submit Bill.
Test Procedure	From the home tab, we navigated to the Create Bill tab. A valid patient ID was entered, and the relevant billing information was displayed. All calculations were correct for every patient in the database. Clicking on Submit Bill stores the total bill value in the patient_record table. Inputting an invalid patient ID results in an appropriate error message. The pages can be navigated to and from repeatedly, and the logout button on the homepage works properly, returning the user to the homepage.
Special requirements	The test stub HospitalUI is needed for the test execution.

Integration Testing:

Test-case identifier	HospitalUI
Test location	final_project\Login\src\main\java\com\example\login\HospitalUI.java
Feature to be tested	Integrated hospital UI
Feature Pass/Fail Criteria	The test passes if the entire hospital UI works properly in conjunction with the database.
Means of Control	1. The start() method is called via a test stub HospitalUI (contained in the same directory as the HospitalUI test).
Data	2. User logs in by choosing their user type and by

	inputting their credentials. 3. If credentials are in the database, user will be routed to their specific UI where they will be able to perform their tasks.
Test Procedure	The test is started by running the test stub. When the login page appears, user will be prompted to choose which type of user they are and to enter their credentials. If the user is part of Registrar, Registrar UI will appear which allows them to perform the tasks of Create Patient Record or View/Edit Patient Record. If Create Patient Record is selected, registrar will prompt the patient to fill in the record with requested information. Once information is filled, the create button will be clicked which stores the record in the database. If a record needs to be edited, registrar will need to enter a patient's ID in the search bar and if the ID is located in the database, the record will fill with information that is stored with the ID. The registrar will then be prompted to make edits to all text fields and once done, can click the update button which updates the information in the database. When registrar has finished their tasks and has no more patients, they can logout by going back to the Home tab and by clicking the logout button. If the user is part of Nurse, Nurse UI will appear which allows them to perform the task of View/Edit Patient Record. First to find a patient, they will need to enter a patient's ID in the search bar and if the ID is located in the database, the record will fill with information that is stored with the ID. The nurse will then be prompted to make edits to text fields such as height, weight, assigned physician, body temperature, blood pressure, nights stayed, and nurse notes. Once edits are made, the update button will be clicked to update the information in the database. When nurse has finished their tasks and has no more patients, they can logout by going back to the Home tab and by clicking the logout button. If the user is part of Doctor, Doctor UI will appear which allows them to perform the tasks of View/Edit Patient Record and Write Discharge Info. If View/Edit Patient Record is selected, the doctor will need to enter the patient's ID into the search bar and click the search

	button. If the ID is located in the database, the record will fill with information that is stored with the ID. The doctor will then be prompted to make edits to text fields such as tests, doctor notes, diagnosis, and medication. Once edits are made, the update button will be clicked to update the information in the database. When a patient is ready to be discharged, Write Discharge Info tab can be clicked which prompts the doctor to enter the patient's ID into the search bar and then click the search button. If ID is found in the database, information regarding their tests, doctor notes, diagnosis, and medication will be displayed. Doctor will check over information and if correct, will click the submit button to create a form that the billing department can view to create their bill and for the patient to view. When doctor has finished their tasks and has no more patients, they can logout by going back to the Home tab and by clicking the logout button. If the user is part of Billing, Billing UI will appear which allows them to perform the task of Create Bill. First to find a patient, they will need to enter a patient's ID in the search bar and if the ID is located in the database, a bill will be filled with information that is stored with the ID. The billing staff will then be able to click the submit button which creates a bill that the patient will then be able to view. When billing user has not more bills to create, they can logout by going back to the Home tab and by clicking the logout button. If the user is Patient, Patient UI will appear which allows them to view their own record, discharge information, and bill. When the patient is finished viewing the information, they can logout by going back to the Home tab and by clicking the logout button.
Special requirements	The test stub HospitalUI is needed for the test execution.