List of user stories that are fully-implemented:

User story 1: hire physicians **Harrish/ Gaurav**
User story 2: hire nurses **Harrish**
User story 3: resign physicians **Harrish**
User story 4: resign nurses **Harrish**
User story 5: store and retrieve vital signs of the patient **Amira**
User story 6: find a patient and display their information **Amira/Mehregan**
User story 7: Family doctor to get the patient's information and be searched **Parmoun**
User story 8: Setting the patient's consent form status **Parmoun**
User story 9: Requesting the lab on the physicians page **Gaurav**
User story 10: Add lab results on the physicians page **Mehregan**
User story 11: Retrieve lab results on the physicians page **Mehregan**
User Story 12: patient discharge **Amira**
User story 13: email family doctor with patient information after discharge **Mehregan**
User story 14: laboratory dashboard (retrieve lab request and fulfill the lab) **Mehregan**
User story 15: registration, login and validation module **Gaurav**
User story 16: Prescribe medication to the patient **Parmoun**

Harrish- Testing user stories 5,6
Amira- Testing user stories 1, 3
Parmoun- Testing user stories 2,4
Mehregan- Testing user stories 7,15
Gaurav- Testing user stories 8, 13

User stories:

User Story 1:

Physicians can be added with incomplete information, can be added without first or last name, can be added without a username or password, Physicians can enter the wrong data type for first and last name. There was an issue with the text fields for creating a physician object. There were no rules except that only an Integer value was accepted for age.

Refactoring Approach: Added checks for each text field to make sure that the required fields were covered and the proper characters were used for each. There is also a check that makes sure that the age is an Integer value and above 0. Most of this refactoring was assigned to and completed by **Gaurav Moturi**.

User Story 2:

Nurses can be added with incomplete information, can be added without first or last name, can be added without a username or password, Nurse can enter the wrong data type for first and last name. There was an issue with the text fields for creating a nurse object. There were no rules except that only an Integer value was accepted for age.

Refactoring Approach: Added checks for each text field to make sure that the required fields were covered and the proper characters were used for each. There is also a check that makes sure that the age is an Integer value and above 0.

User Story 3:

No removal of the resignation option when there are no physicians in the system. It was possible to reach the Resign Physician page even when no physicians are currently employed by the hospital.

Refactoring Approach: Refactoring was not done on this issue because it did not negatively impact the user experience of the user.

User Story 4:

No removal of the resignation option when there are no nurses in the system. It was possible to reach the Resign Nurse page even when no physicians are currently employed by the hospital.

Refactoring Approach: Refactoring was not done on this issue because it did not negatively impact the user experience of the user.

User Story 5:

No Bugs were reported for this user story. All the test cases worked as expected.
The code smells were discussed with the team members and we decided that they were not severe. However, I did some refactoring to improve the code and that was added to a new document under the Refactoring folder on GitHub.

VitalSigns.java
Large Class: Severity 4
VitalSignsDisplayGUI
Separation of Input and Display Panels:
I separated the input field and the display area into two separate panels, named “inputPanel” and “displayPanel”. This separation improves the readability of the interface when new input is entered. With this separation, when we clear out the display panel’s components using “displayPanel.removeAll()”, the input panel remains intact, providing a clearer user experience and allowing for further input without disruption.

Removal of Main Method:
I removed the main method from this class, making it accessible and runnable only from another class, specifically the “PatientGUI” class. This change adheres to the Single Responsibility Principle (SRP) by delegating the responsibility of application startup to a dedicated class.

User Story 6:

No Bugs were reported for this user story. All the test cases worked as expected.
The code smells were discussed with the team members and we decided that they were not severe. However, I did some refactoring to improve the code and that was added to a new document under the Refactoring folder on GitHub.

PatientGUI has undergone refactoring to enhance the "Display All Patients" feature, now specifically displaying patients assigned to the physician rather than all patients in the hospital. Additionally, the "Get Patient by ID" functionality has been tailored to retrieve information solely for patients assigned to the physician.

User Story 7:

Improper Family Doctor Assignment in GUI:

The family doctor field in the nurse's GUI does not connect to a specific patient, leading to potential misassignments.

Refactoring Approach:
The family doctor assignment feature was refactored in the GUI. Now, the nurse must first select a patient from the "View Patients" list, after which they can assign a family doctor through a dedicated interface. This ensures the correct linkage between a patient and their family doctor, enhancing data integrity and reducing user errors.

User story 8:

- **Hardcoded Database Credentials:**

Hardcoded Database Credentials: Database connection details are hardcoded, posing a security risk and reducing flexibility.

Refactoring Approach:
Credentials were externalized to a configuration file or environmental variables, which are read dynamically at runtime. This not only secures sensitive information but also allows easy changes to the database configuration without the need to recompile the code.

- **Improper Exception Handling:**

The use of 'e.printStackTrace()' for exception handling can leak sensitive information.

Refactoring Approach:
Exception handling was restructured to log errors to a logging framework with appropriate logging levels. This reduces information leakage while still preserving the ability to diagnose issues from the logs.

- **Single Responsibility Principle Violation:**

The class handles multiple database operations, making it complex and difficult to maintain.

Refactoring Approach:
The class was refactored to stick to the Single Responsibility Principle. Distinct responsibilities were decoupled into separate classes, each handling a specific aspect of database interaction, which simplified maintenance and enhanced modularity.

- **Primitive Obsession:**

Uses primitive data types ('String', 'int') extensively instead of more descriptive types or objects.

Refactoring Approach:
Custom classes or structures were introduced to encapsulate complex data, replacing primitives with objects that provide more context and functionality. For example, a PatientID object could encapsulate validation logic that ensures ID integrity throughout the system.

User story 13:

Reported Issue: 'FamilyDoctor.java' & 'Nurse java'

Duplicated Code: Potential duplicated code in handling patient information, suggesting a need for abstraction.

Feature Envy: Excessive access to data or methods of another class, indicating that some methods might be better placed in the class they interact with the most.

'NurseGUI.java' & 'NursePatientAddFrame.java
Large Class / God Object: Manages multiple responsibilities from UI components to event handling, violating the Single Responsibility Principle.
Long Method: Methods handling events could become overly complex, indicating a need for decomposition .

'PatientinformationGUI.java'
Long Parameter List: Methods requiring many parameters suggest the need for grouping parameters into a single object.
Speculative Generality: Presence of unused methods designed for future flexibility, unnecessarily complicating the code.

Approach: The main reported bugs for this user story were relevant to other user stories and after the other user stories fixed their issues, the remove patient method in the DB was updated and the bugs were closed.