List of user stories that are fully-implemented:
User story 1: hire physicians Harrish
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 13: email family doctor with patient information after discharge Mehregan
User story 15: registration, login and validation module: Gaurav

Main developers of:
User story 1: Harrish Elango
User story 2: Harrish Elango
User story 3: Harrish Elango
User story 4: Harrish Elango
User story 5: Amira Mohamed
User story 6: Amira Mohamed, Mehregan Mesgari
User story 7: Parmoun Khalkhali Sharifi
User story 8: Parmoun Khalkhali Sharifi
User story 13:Mehregan Mesgari
User story 15: Gaurav Charan Moturi

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 story 9: Mehregan Mesgari
User story 10: Gaurav Charan Moturi
User story 11: Mehregan Mesgari
User story 12: Amira Mohamed
User story 14:Mehregan Mesgari
Involved
Testing

Part 1: End-to-end testing

User story 5: Store and retrieve vital signs of patient:
End-to-end testing -Record vital sign data for patient:

1. Choose "Login as Doctor"
2. Sign in as physician with appropriate credentials
3. Click "display all patients" button to find patients under care
4. Press "record vital signs" button

5. Fill in the patients ID number that you want to add vitals for
6. Press “record vital signs” button
7. Vital signs have been recorded under the patients.

Expected Result:
Vital signs added for patient under physician care.
All information matches the records in the database.

Edge test case finds:
Temperature has to be between 30-45
Systolic pressure needs to be between 0-200
Diastolic pressure needs to be between 0-200
Heart rate needs to be between 0-200
Respiratory rate needs to between 10-60
Will still try and save vitals if patient id does not exist - one failure
Otherwise works as intended and assigns vital signs to a patient under the care of the Physician

End-to-end testing - Retrieve vital sign data from patient:
1. Choose “Login as Doctor”
2. Sign in as physician with appropriate credentials
3. Click "display all patients" button to find patients under care
4. Press "retrieve vital signs" button
5. Fill in the patients ID number that you want to see vitals of
6. Press “Submit” button
7. vital signs of patient are visible in frame.

Expected Result:
Vital signs visible for patient chosen by Physician.
All information matches the records in the database.

Edge test case finds:
Wrong patient id gives same response as patient without any vital signs saved yet - one failure
Otherwise works as intended and the Physician can see the vital signs of patients in his care

User story 6: Find a patient and display their information
1. Choose “Login as Doctor”.
2. Sign in as physician with appropriate credentials
3. Click on the “Display All Patients” button to view all patient information

3b. Click “Get patient by their name” to find a patient by their name to view their information.
3c. Click “Get patient by their ID” to find a patient by their patient ID number to view their information.

4. 4. Click “View Labs and Meds” button
5. 5. Search for patient by ID or by name
6. 6. Press “View Prescriptions” or “View Medications” button to retrieve this data on patients under physician care.

Expected Results
The patient's information is displayed, including personal details and medical history.
All displayed information accurately reflects the records in the database.

Edge test case finds:
Will display no information if either name or ID does not exist in database
If no labs or medications assigned to patient, it will let the Physician know that there is none.

Part 2: Code smells found in the classes used in user stories 5 and 6

The severity scale is from 1 (least severe) to 5 (most severe), and provide each one has a suggestion for addressing it.

VitalSigns.java
- Large Class: Severity 4
  - Suggestion: Refactor the class to adhere to the Single Responsibility Principle (SRP). Divide the class into smaller classes, where one handles vital signs data manipulation and another handles database interactions.
- Speculative Generality: Severity 3
  - Suggestion: Identify and remove or simplify unused methods or functionalities. Keep the implementation as simple as possible with a focus on current requirements rather than speculative future use cases.

VitalSignsGUI.java
- Duplicate Code: Severity 4
  - Suggestion: Abstract repeated code patterns into shared methods or utilize a factory pattern for GUI components that are initialized in a similar manner. This will reduce code duplication and improve maintainability.
- Long Parameter List: Severity 3
  - Suggestion: Introduce parameter objects for methods that require many parameters. Alternatively, group related parameters into objects that make logical sense together. This simplifies method signatures and improves code readability.

Patient.java
- Missing Encapsulation: Severity 4
  - Suggestion: Ensure all fields are private and accessible only through getters and setters. This improves encapsulation and protects the integrity of the data.
- Feature Envy: Severity 3
  - Suggestion: If a method is more interested in a class other than its own, consider moving this method to the class it is most interested in. This may also involve rethinking the design to better encapsulate behavior with relevant data.

PatientGUI.java
- Switch Statements: Severity 3
  - Suggestion: Replace switch statements or if-else chains with polymorphism, where different behaviors are encapsulated in their respective classes. This approach leverages inheritance and interfaces to dynamically determine behavior.
- Long Method: Severity 4
  - Suggestion: Break down long methods into smaller, more focused methods. This improves readability, makes the code easier to maintain, and facilitates unit testing.

PatientInformationGUI

- Large Class / God Object Smell: Severity 5
  - Suggestion: Apply the MVC pattern to separate concerns. Create separate classes for handling the view (GUI), the model (data), and the controller (logic) aspects. This reduces class complexity and enhances maintainability.

- Magic Strings and Numbers: Severity 3
  - Suggestion: Replace magic strings and numbers with constants or configuration files. This centralizes the management of these values, making the code easier to maintain and modify.