# Group 23

# ECG Analysis Dashboard
# Master Test Plan

## Version 1.0

Prepared By Group 23
Sajeenthiran P – 210553J
Ranaweera HK - 210523T
Rathnayaka W.T - 210536K

# Revision History

| Date | Version | Description | Author |
|---|---|---|---|
| 15/10/24 | 1.0 | Initial version | Group 23 |

# Table of Contents

# Master Test Plan

## 1. Evaluation Mission and Test Motivation

The mission of the ECG Analysis Dashboard is to deliver accurate and reliable ECG signal analysis, including preprocessing, descriptive analysis, arrhythmia detection, myocardial disease detection, and personal identification. The goal of the testing process is to ensure the system meets healthcare regulations and user expectations for reliability, accuracy, security, and performance.

This test plan focuses on evaluating the following:

- Ensuring the system functions correctly under various conditions.
- Finding as many bugs as possible and correct them before the system is being deployed to the users. A special thorough search for bugs will be carried out.
- Identify key design and implementation issues that threaten the system's quality or fail to meet user requirements. These issues will be addressed and resolved before the system's delivery.
- Meeting functional and non-functional requirements.
- The system must comply with quality standards and align with those of existing systems. To ensure this, performance testing is conducted alongside user acceptance testing to verify that the system meets the necessary requirements
- Users should be informed about the testing process as well. When user acceptance testing is conducted, stakeholders must be notified that the system is in its Beta version.
- Compliance with medical data privacy and security standards (HIPAA, GDPR)

## 2. Target Test Items

The target items for testing include:

1. **User Interfaces**:
    - The web interface for doctors and patients to upload and view ECG signals.
    - Functionality of form submissions, graphs, and results display.
2. **ECG Preprocessing and Analysis Functions**:
    - Preprocessing modules for noise and artifact removal.
    - Descriptive analysis of ECG signals, including waveform characteristics.
    - Arrhythmia and myocardial infarction detection modules.
    - Personal identification using ECG data.
3. **Data Management**:
    - Verification of correct retrieval and processing of ECG signals stored in the User web application memory.

The listing below identifies those test items—software, hardware, and supporting product elements —that have been identified as targets for testing. This list represents what items will be tested.

## 3. Test Approach

### 3.1 Testing Techniques and Types

### *3.1.1 Function Testing*

Function testing of the target system should concentrate on requirements that can be directly traced to use cases, business functions, or business rules. The primary objective of these tests is to ensure correct data input, processing, retrieval, and the accurate implementation of business rules. This form of testing follows black-box techniques, meaning the application is validated by interacting with it through the Graphical User Interface (GUI) and evaluating the outputs or results. The following table provides an overview of the recommended testing approach.

| | |
|---|---|
| Technique Objective: | Exercise target-of-test functionality, including navigation, data entry, processing, and retrieval to observe and log target behavior. |
| Technique: | Execute each use-case scenario's individual use-case flows or functions and features, using valid and invalid data, to verify that:<br>• the expected results occur when valid data is used<br>• the appropriate error or warning messages are displayed when invalid data is used<br>• each business rule is properly applied |
| Oracles: | Outline one or more strategies that can be used by the technique to accurately observe the outcomes of the test. The oracle combines elements of both the method by which the observation can be made and the characteristics of specific outcome that indicate probable success or failure. Ideally, oracles will be self-verifying, allowing automated tests to make an initial assessment of test pass or failure, however, be careful to mitigate the risks inherent in automated results determination. |
| Required Tools: | The technique requires the following tools:<br>• Test Script Automation Tool<br>• GIT Hub for backing up the code<br>• installation-monitoring tools (registry, hard disk, CPU, memory, and so forth) |
| Success Criteria: | All the tests carried out to the identified use cases are working properly with no errors.<br>All automated tests are guaranteed to give a reasonable and a validated result |
| Special Considerations: | none |

- Selenium IDE is used as the Test Script Automation Tool
  (Python Script)

| System Feature | Test Case ID | Description | Preconditions | Test Steps | Expected Results | Results |
|---|---|---|---|---|---|---|
| ECG Purifier | TC-4.1.1 | Verify that the system filters out noise from the uploaded ECG signals. | User has raw ECG data ready for upload. | 1. Navigate to the preprocessing section of the dashboard.<br>2. Upload raw ECG data.<br>3.1.2 Initiate the preprocessing operation. | The system processes the data and the output ECG signal should show reduced noise levels compared to the input signal. | Passed |
| | TC-4.1.2 | Verify that the system normalizes the amplitudes of ECG signals correctly. | User has uploaded raw ECG data. | 1. Navigate to the preprocessing section.<br>2. Upload raw ECG data<br>3. Initiate the normalization operation. | The output ECG signal should have normalized amplitude values, with all signal amplitudes within a specified range (e.g., 0 to 1). | Passed |
| | TC-4.1.3 | Ensure that the system removes baseline wander from ECG signals effectively. | User has uploaded raw ECG data. | 1. Navigate to the preprocessing section.<br>2. Upload raw ECG data.<br>3. Initiate the baseline wander removal operation. | The output ECG signal should exhibit minimal to no baseline drift, with the baseline being stable. | Passed |
| | TC-4.1.4 | Verify that the system applies a notch filter to remove powerline | User has uploaded raw ECG data with known powerline noise. | 1. Navigate to the preprocessing section.<br>2. Upload raw ECG data with powerline noise. | The output ECG signal should show a significant reduction in powerline interference, with | Passed |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | interference noise from ECG signals. | | 3. Initiate the notch filter operation. | clear signal characteristics. | |
| ECG Insights and Analysis | TC-4.2.1 | Verify that the system computes mean, median, and standard deviation of ECG signals accurately. | User has selected an ECG dataset. | 1. Navigate to the descriptive analysis section.<br>2. Select an ECG dataset for analysis.<br>3. Initiate the statistical analysis operation. | The system should return the mean, median, and standard deviation values for the selected ECG dataset. | Passed |
| | TC-4.2.2 | Ensure that the system generates time-series plots of ECG signals accurately. | User has selected an ECG dataset. | 1. Navigate to the descriptive analysis section.<br>2. Select an ECG dataset for analysis.<br>3. Initiate the plot generation operation. | The system should display time-series plots of the ECG signals, accurately reflecting the waveform characteristics. | Passed |
| Myocardial Disease Detection | TC-4.3.1 | Verify that the system accurately diagnoses potential myocardial diseases from ECG signals. | □ User has uploaded an ECG signal for analysis. | 1. Navigate to the myocardial disease detection section.<br>2. Upload the ECG signal.<br>3. Initiate the diagnosis operation. | The system should provide a diagnostic report identifying potential markers for myocardial diseases along with associated probabilities. | Passed |
| Arrhythmia Detection | TC-4.4.1 | Ensure that the system detects irregular heartbeats in the uploaded ECG signals effectively. | User has uploaded an ECG signal for analysis. | 1. Navigate to the arrhythmia detection section.<br>2. Upload the ECG signal.<br>4. Initiate the arrhythmia detection operation. | The system should identify and highlight segments of the ECG signal where arrhythmias are detected. | Passed |
| Person Identification | TC-4.5.1 | Verify that the system accurately compares two ECG signals to determine if they are from the same person. | User has uploaded two ECG signals. | 1. Navigate to the person identification section.<br>2. Upload the first ECG signal.<br>3. Upload the second ECG signal.<br>4. Initiate the comparison operation. | The system should return a similarity score with a confidence level, indicating whether the signals are from the same person. | Passed |

### *3.1.3 User Interface Testing*

The goal of UI testing is to ensure that the user interface offers proper access and smooth navigation through the application's features. Additionally, it verifies that the elements within the interface perform as intended and comply with corporate or industry standards.

| | |
|---|---|
| Technique Objective: | Perform the following tasks to assess and document compliance with standards and expected behavior:<br><br>• Render web pages efficiently, minimizing memory and data usage while maintaining high-quality output.<br>• Ensure the system provides an excellent user experience by making it intuitive and easy to navigate and control.<br>• Design the index page to be visually appealing and clearly convey the purpose of the system.<br>• Ensure platform independence, allowing web pages to run smoothly across a wide range of devices and operating systems.<br>• Make tabs and buttons intuitive, providing clear indications of their functions and content.<br>• Ensure buttons route users to the correct pages without any bugs or errors.<br>• Verify that form submissions function correctly, without producing unexpected errors. |
| Technique: | Use Selenium IDE along with the fire bug tool for Firefox to generate web requests to see whether the User Interface behaves as expected |
| Oracles: | UI components behave as expected; no broken routes or form submission errors. |
| Required Tools: | Selenium IDE to test the requests sent to the web page |
| Success Criteria: | All the pages and routes are working well. There are no miss spelling or broken routes that fails the User Interface The system is user friendly with an easily understandable user interface |
| Special Considerations: | Not all properties for custom and third-party objects can be accessed. |

| Test Case ID | Test Case Description | Objective | Steps to Execute | Expected Result | Result | Notes |
|---|---|---|---|---|---|---|
| TC-UI-001 | Verify the landing page loads successfully | Ensure that the landing page is displayed correctly. | 1. Open the application URL.<br>2. Observe the landing page. | The landing page should load without errors and display the correct title and branding. | Passed | Test on various browsers. |
| TC-UI-002 | Check responsiveness of the dashboard | Verify that the UI adjusts correctly on different devices. | 1. Access the application on different screen sizes (desktop, tablet, mobile). | The UI should adapt appropriately, maintaining usability and layout integrity. | Passed | Test across multiple devices. |
| TC-UI-003 | Layout and alignment of UI elements. | | | | Passed | |
| TC-UI-004 | Test the "Upload ECG" button functionality | Confirm that the upload button works as expected. And renders the ECG graph | 1. Click on the "Upload ECG" button.<br>2. Select a valid ECG file.<br>3. Click "Open". | The selected file should appear in the upload field, and the corresponding ECG plot should appear | Passed | Validate with different file types. |
| TC-UI-005 | Verify the "Save" button functionality | Ensure the saved ECG signal table gets updated in UI. | 1. Upload a valid ECG file.<br>2. Click the "save" button. | The system should process the file and display a loading spinner until complete. And the saved ECG table should be updated | Passed | Repeat for several ECG files |

| | | | | | | |
|---|---|---|---|---|---|---|
| TC-UI-006 | Descriptive Analysis UI | Ensure all the components are rendered | 1. Upload a valid ECG file.<br>2. Click the "save" button.<br>3. Click the ECG Insights and Analysis | All the components such as , the ECG Grid Analysis ,Heart Rate (bpm), Rhythm Regularity, Average PQ Interval, Average QRS Interval. Average ST Interval (ms) | Passed | If no ECG uploaded ,it must render 'Please upload a record first.' |
| TC-UI-007 | Arrhythmia Detection UI | Ensure all the components are rendered | 1. Upload a valid ECG file.<br>2. Click the "save" button.<br>3. Click the "Arrhythmia Detection " section | The ECG plot and arrhythmia type counts UI should be rendered. | Passes | If no ECG uploaded ,it must render 'Please upload a record first.' |
| TC-UI-008 | Myocardial Infarction Detection | Ensure all the components are rendered | 1. Upload a valid ECG file.<br>2. Click the "save" button.<br>3. Click the "Myocardial Infarction Detection " section | The ECG plot and the Myocardial detection indicator should be rendered. | Passed | If no ECG uploaded ,it must render 'Please upload a record first.' |
| TC-UI-009 | ECG signal Person Comparator | Ensure all the components are rendered | 1. Upload a valid ECG file.<br>2. Click the "save" button.<br>3. Click the "ECG signal Person Comparator " section<br>4. Select Two ECG signals and click predict to compare two signals | Two ECG signals with the comparison results must be rendered | Passed | If no ECG uploaded ,it must render 'Please upload a record first.' |

### *3.1.3 Performance Profiling*

Performance profiling is a type of performance testing that measures and evaluates response times, transaction rates, and other time-sensitive metrics. Its purpose is to ensure that performance requirements are met. This process involves profiling and tuning the performance behaviors of the system under test, considering factors such as workload or hardware configurations.

| | |
|---|---|
| Technique Objective: | Exercise behaviors for designated functional transactions or business functions under the following conditions to observe and log target behavior and application performance data:<br>• System Performance under normal anticipated workload anticipated worst-case workload<br>• System's Performance under anticipated worst-case workload. |
| Technique: | • Consider each functionality individually, executing one at a time to measure memory usage and execution time.<br>• Perform multiple transactions simultaneously, monitoring memory usage and execution time.<br>• Execute all transactions concurrently and assess memory usage and execution time.<br>• Test the entire system by simulating multiple user instances, performing all transactions, and applying the previous three steps across the system as a whole. |
| Oracles: | The results of automated tests are considered when evaluating the performance of functionalities. However, these results are platform-dependent and influenced by the user's internet connection. |
| Required Tools: | • Apache JMeter<br>• Blaze Meter<br>• Chrome Developer Tools |

| | |
|---|---|
| Success Criteria: | The technique supports testing in two scenarios:<br>• Single transaction or single user: Ensures successful execution of transaction scripts without failures caused by test implementation issues.<br>• Multiple transactions or multiple users: Ensures successful simulation of the workload without failures resulting from test implementation issues. |
| Special Considerations: | Comprehensive performance testing involves generating a background workload on the server. Several methods can be used to achieve this, including:<br>• Simulating a large number of clients, often hundreds, by creating a "virtual" user load. Remote Terminal Emulation tools are commonly used for this, and this method can also generate network "traffic."<br>• Using multiple physical clients, each running test scripts to apply load on the system. |

### *3.1.4 Load Testing*

Load testing is a type of performance test that exposes the system under test to varying workloads to assess its performance and ability to operate correctly under different load conditions. The primary goal is to ensure that the system functions properly even beyond its expected maximum workload. Additionally, load testing evaluates key performance metrics, including response times, transaction rates, and other time-sensitive aspects.

| | |
|---|---|
| Technique Objective: | Execute specific transactions or business cases under different workload conditions to monitor and record system behavior and performance data. This includes:<br>• Observing the system's behavior under normal workload conditions.<br>• Observing the system's behavior under worst-case workload conditions.<br>• Monitoring system performance with a significant number of concurrent users under both normal and worst-case workloads |
| Technique: | Should test the systems daily weekly and monthly for peek loads. User acceptance testing is useful here.<br>Workloads should include spiky Peaks and regular peaks, use Grafana K6 to create a large number of web requests to test the load.<br>The workload variations must be tested in different environments and different platforms too. |

| | |
|---|---|
| Oracles: | System performance can vary across different platforms, making manual testing impractical since it's not feasible to physically wear down a computer. Instead, the code must be modified and updated to apply significant load on the system for testing purposes |
| Required Tools: | The technique requires the following tools:<br>• Test Script Automation Tool (Grafana K6 Load test Scripting) |
| Success Criteria: | • Exhausting the system with the worst case scenario will help to determine the system performance boundaries and requirements. This will help to determine whether the system can behave as expected for an expected workload.<br>• Average transactions on single user and multiple users should work perfectly without any errors.<br>• System should be able to withstand spiky peek loads and regular peeks without failing the system. |
| Special Considerations: | Load testing should be performed on a dedicated machine or at a dedicated time. This permits full control and accurate measurement.<br>Create virtual users to load and simulate many clients and check the performance of the system for at least hundred users. Remote Terminal Emulation tools are also used load the network with workload. |

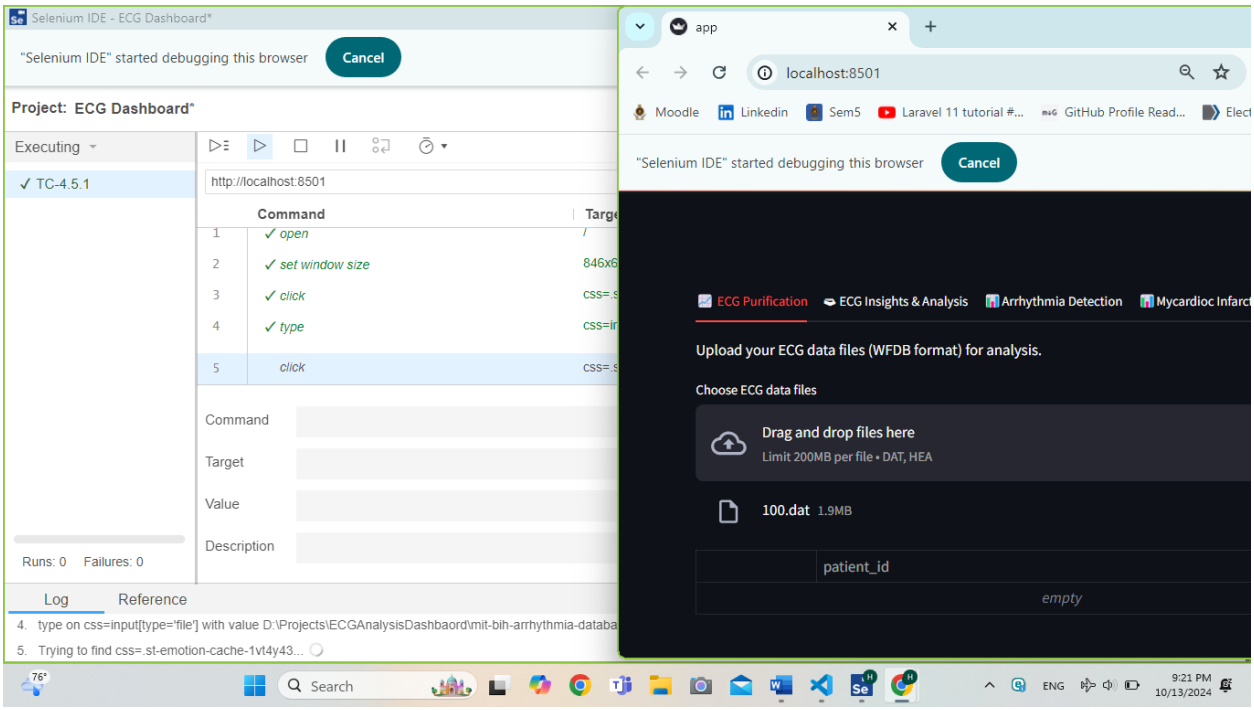
## 4. Deliverables

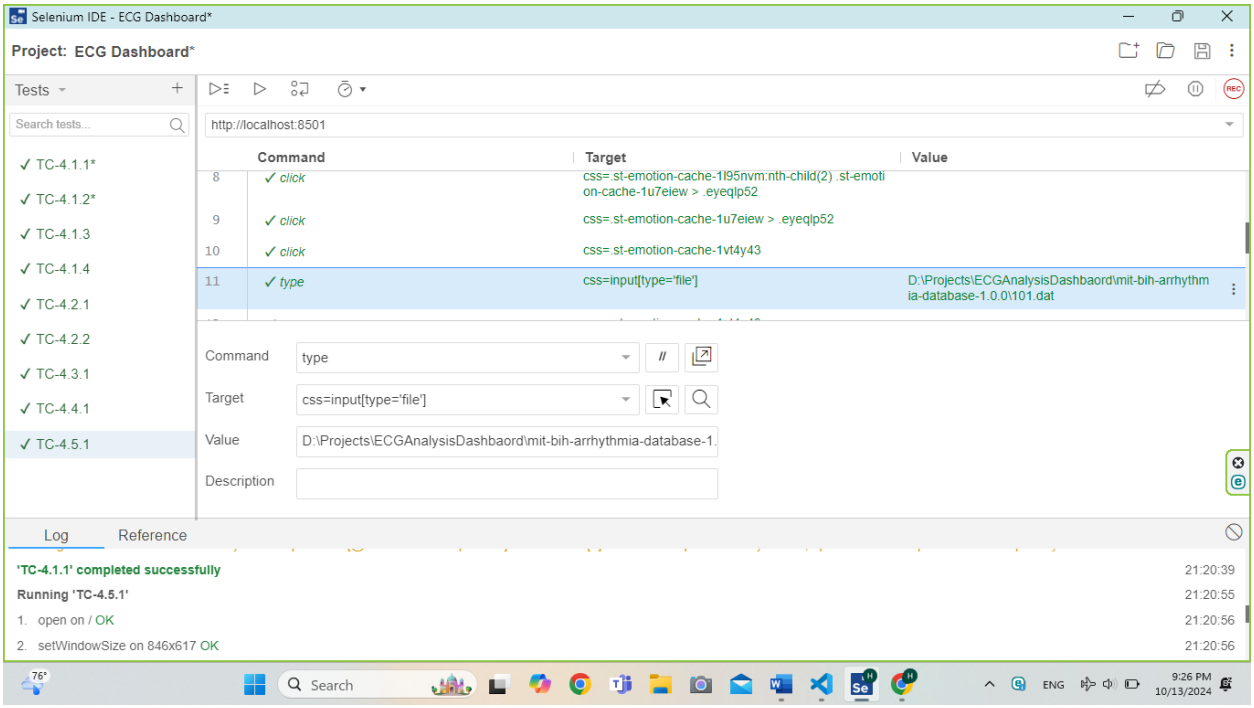
This section outlines the various artifacts that will be produced during the testing process, which are valuable deliverables for the stakeholders involved. These deliverables provide direct, tangible benefits and serve as key indicators of the test effort's success. The primary deliverables include:

- Functional Testing Results
- User Interface testing details
- Code inspection reports from Sonaqube
- Performance statistics from Blaze Meter
- Browser performance metrics from Chrome Developer Tools
- Load Testing results from Grafana K6

## Test Evaluation Summaries

- Functional Testing Results
  These are the detailed logs that give the outcome of every test that is written for the functionalities. If even a single test log is given the output as a failure, the that functionality is recorrected and all the tests are run again and the test logs are checked for faults. Test logs will be provided very often soon after a new functionality is added or a change to an existing functionality is made.

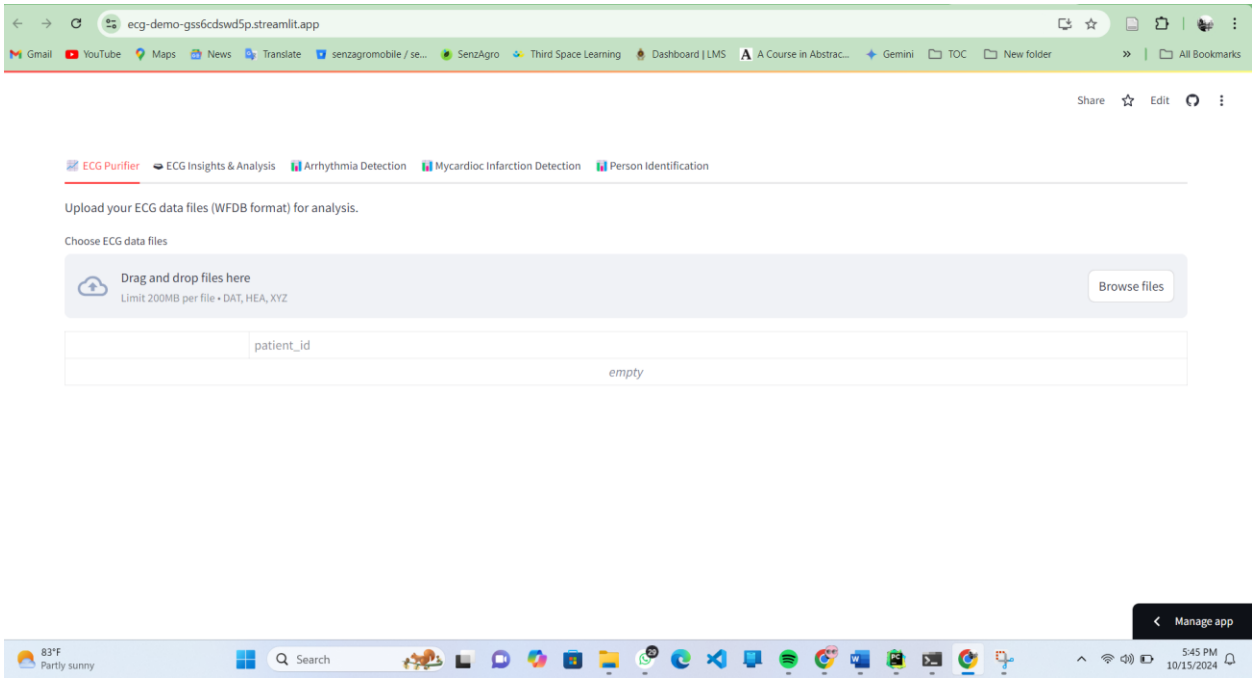




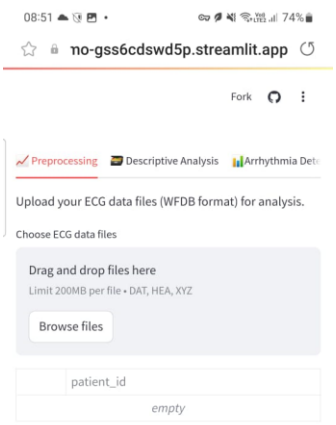

- User Interface Test Details

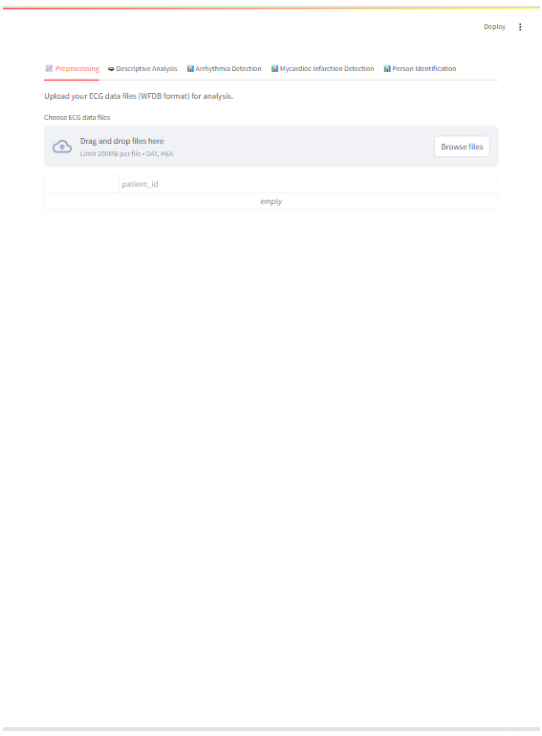
Tests will be conducted to identify faults in form submissions and functional actions. Details of the User Interface (UI) testing will be provided after the system development is complete and UI checks are performed for errors.

TC-UI-001 (Verify the landing page loads successfully)




TC-UI-002(Responsiveness)

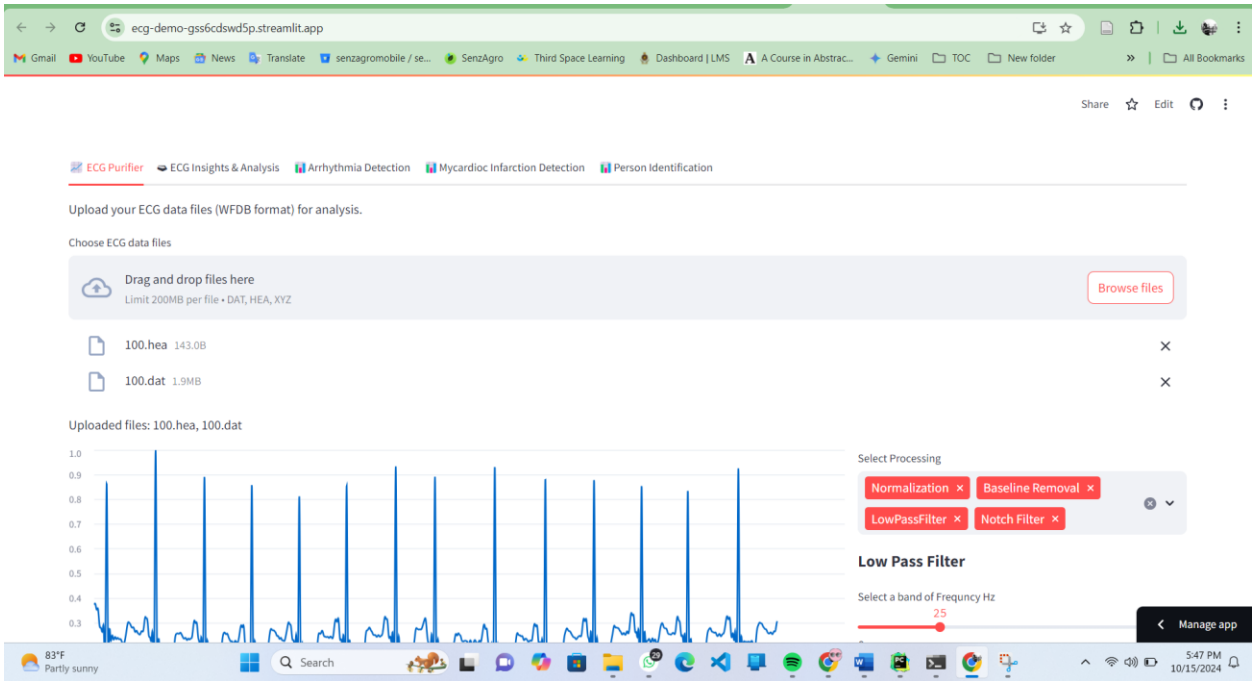


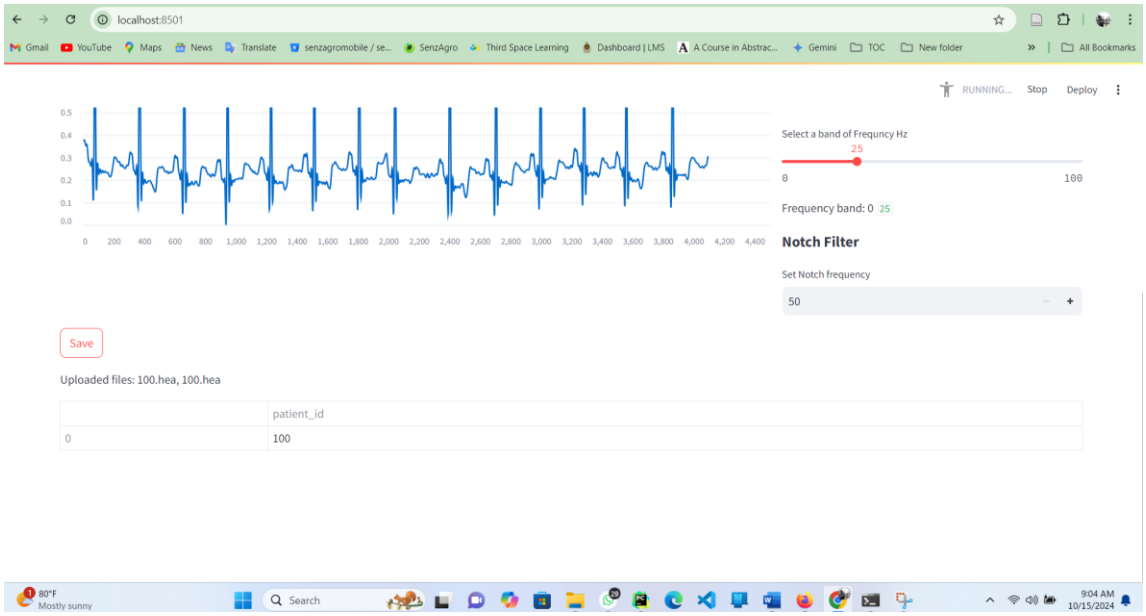

Smart Phone (Samsung A32)

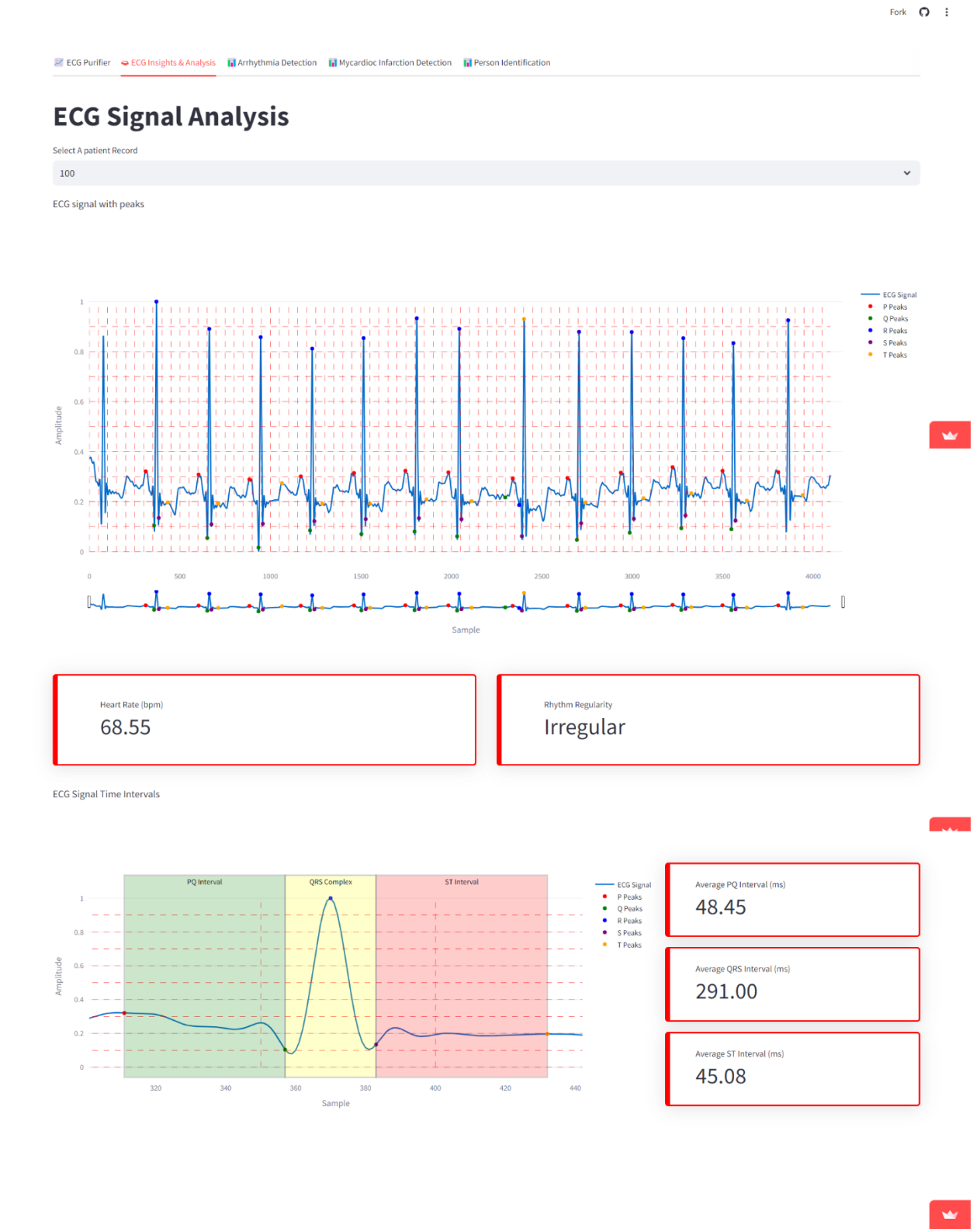
iPad Pro

TC-UI-004(Test the "Upload ECG" button functionality)

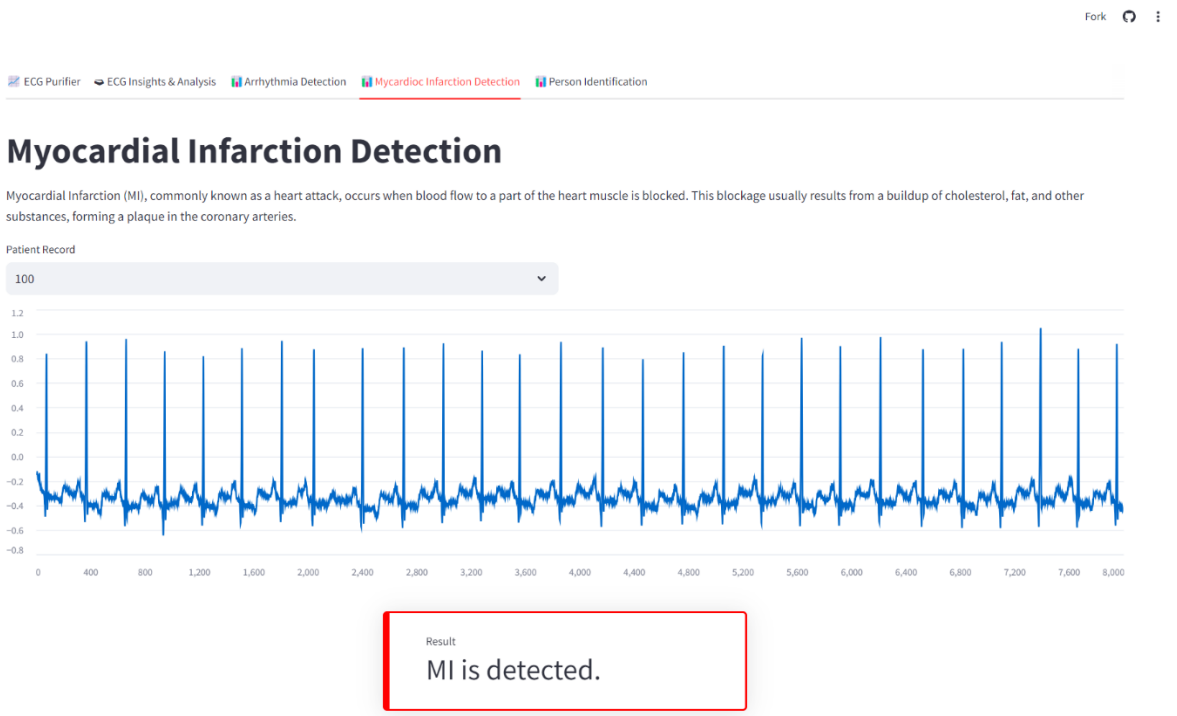


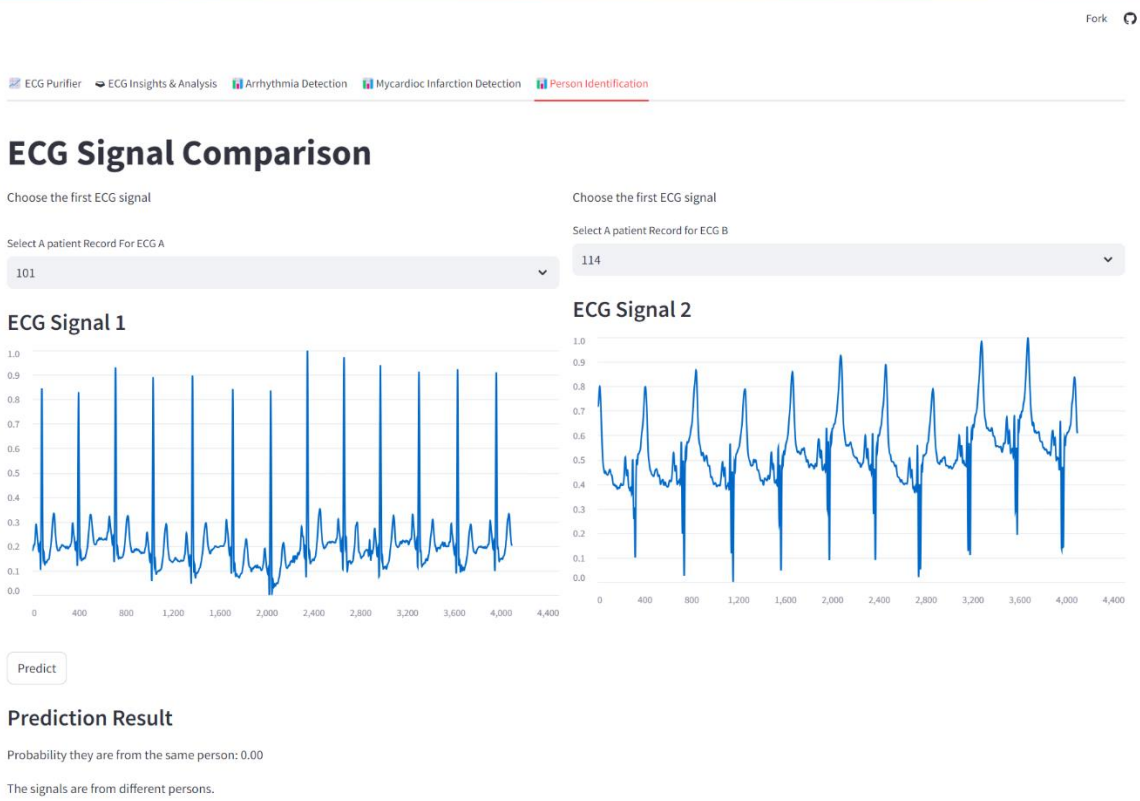

TC-UI-005 (Verify the "Save" button functionality)




Fork
ECG Purifier
ECG Insights & Analysis
Arrhythmia Detection
Mycardioc Infarction Detection
Person Identification
ECG Signal Analysis
Select A patient Record
100
ECG signal with peaks
ECG Signal
P Peaks
Q Peaks
R Peaks
S Peaks
T Peaks
Amplitude
Sample
Heart Rate (bpm)
68.55
Rhythm Regularity
Irregular
ECG Signal Time Intervals
PQ Interval
QRS Complex
ST Interval
Average PQ Interval (ms)
48.45
Average QRS Interval (ms)
291.00
Average ST Interval (ms)
45.08

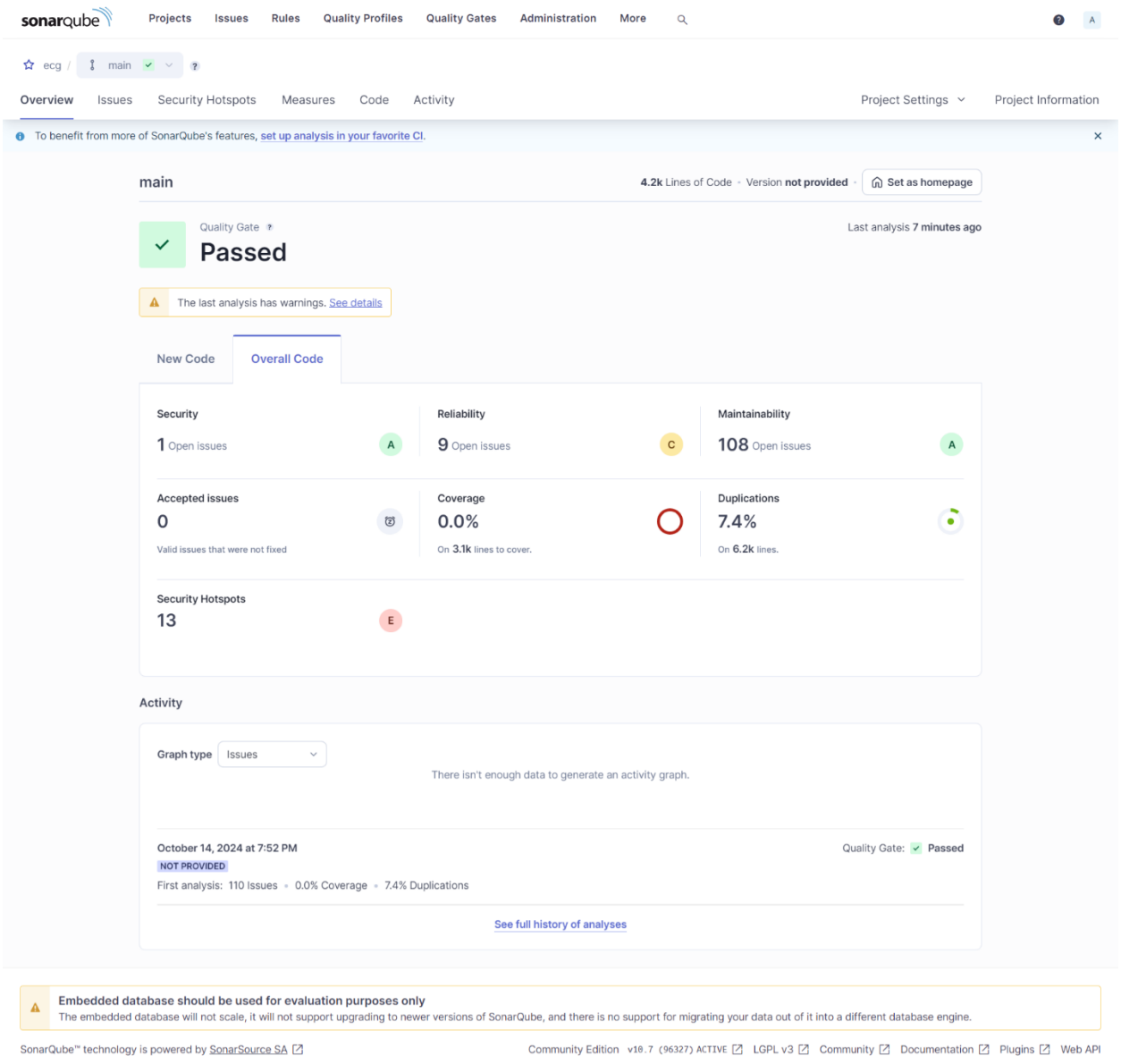

Fork
ECG Purifier
ECG Insights & Analysis
Arrhythmia Detection
Mycardioc Infarction Detection
Person Identification
ECG Signal with
Amplitude
Sample
Total number of beats
14
N beat count
9
S beat count
2
V beat count
2
F beat count
1


Fork
ECG Purifier
ECG Insights & Analysis
Arrhythmia Detection
Mycardioc Infarction Detection
Person Identification
Myocardial Infarction Detection
Myocardial Infarction (MI), commonly known as a heart attack, occurs when blood flow to a part of the heart muscle is blocked. This blockage usually results from a buildup of cholesterol, fat, and other substances, forming a plaque in the coronary arteries.
Patient Record
100
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
0
400
800
1,200
1,600
2,000
2,400
2,800
3,200
3,600
4,000
4,400
4,800
5,200
5,600
6,000
6,400
6,800
7,200
7,600
8,000
Result
MI is detected.


Fork
ECG Purifier
ECG Insights & Analysis
Arrhythmia Detection
Mycardioc Infarction Detection
Person Identification
ECG Signal Comparison
Choose the first ECG signal
Select A patient Record For ECG A
101
ECG Signal 1
Choose the first ECG signal
Select A patient Record for ECG B
114
ECG Signal 2
Predict
Prediction Result
Probability they are from the same person: 0.00
The signals are from different persons.

- Code Inspection Tool

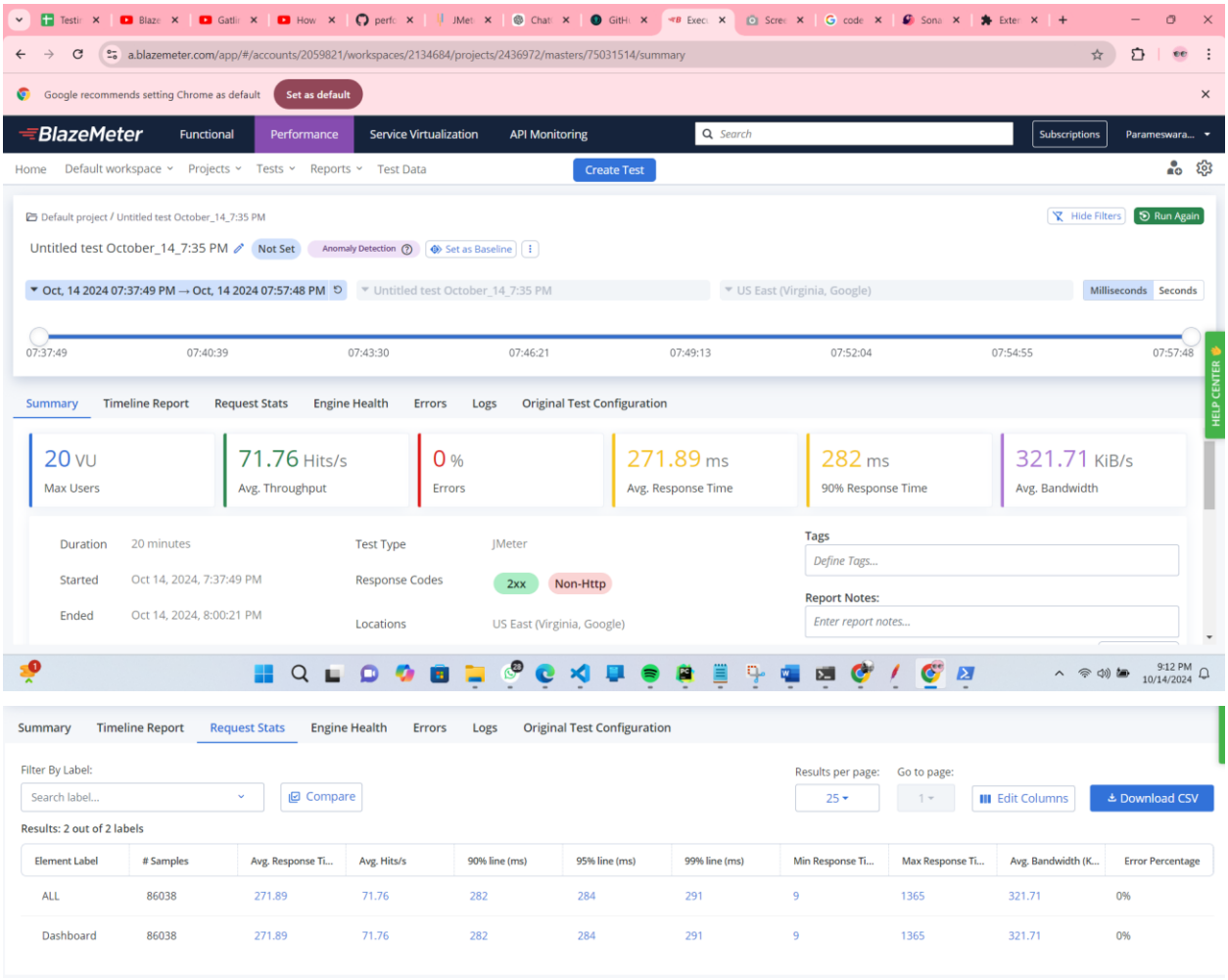
  Code inspection details can be obtained by running the code inspection test in SonaQube. This process highlights issues such as spelling errors, empty tags, and unused items, which can assist in refactoring the code for improved performance. These reports will be generated approximately once a week and at the end of the project.




- Performance statistics from Blaze Meter

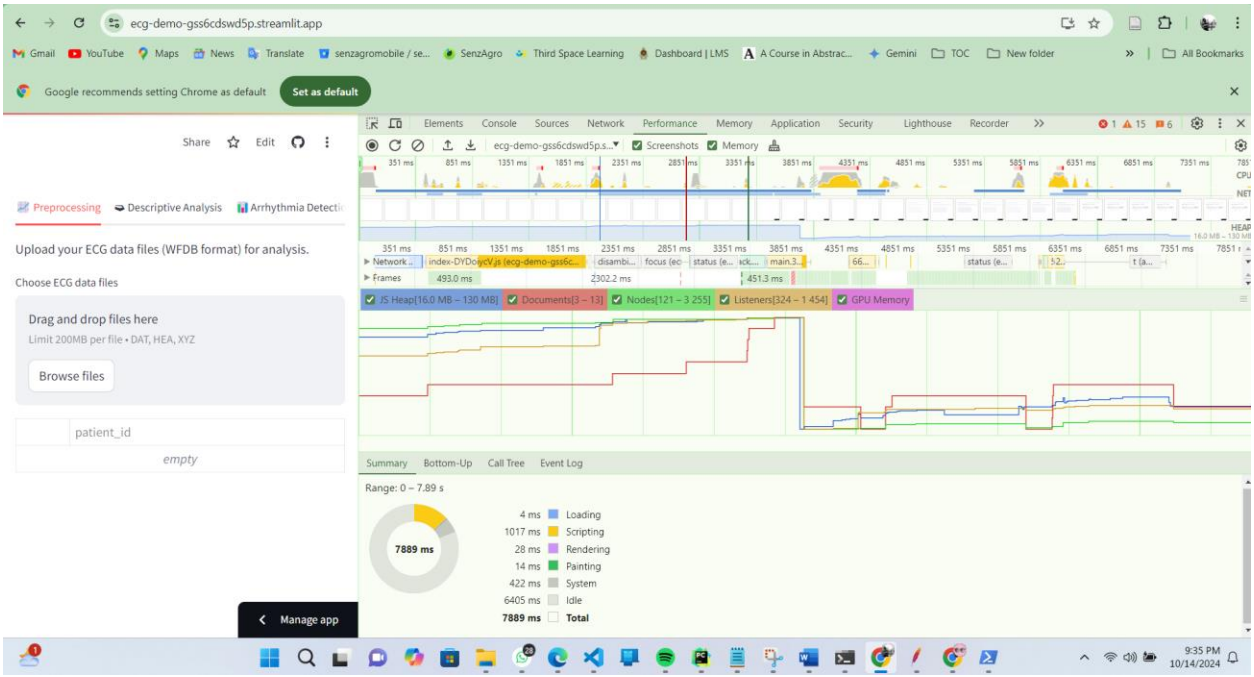
  This BlazeMeter performance test report highlights several key attributes to assess system performance under load.

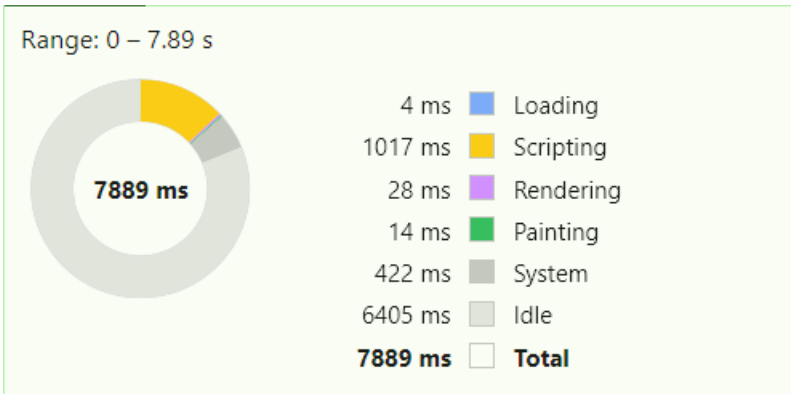
**Virtual Users (VU)** refers to the number of simulated users (20 in this case) interacting with the system during the test. **Throughput** measures how many hits (or requests) the system handled per second, averaging 71.76 hits/s. **Response Times** are crucial in evaluating speed, with an average of 271.89 milliseconds and a 90th percentile of 282 milliseconds, meaning 90% of the requests were processed within this time. **Errors** were at 0%, indicating all requests were successfully processed, while **Response Codes** show the distribution of successful (2xx) and **Non-HTTP** responses. Finally, **Bandwidth** represents the amount of data transferred, averaging 321.71 KiB/s, and the test was conducted from a **US East (Virginia, Google)** location. These attributes together give a comprehensive view of system performance, including speed, reliability, and capacity to handle traffic.




- Browser Performance Details

  Browser performance details are checked to test the memory usage for the browser while the application is running as well as the memory usage for the application. The size of the page loaded can be also checked by the browser performance details, so that the data usage for the application can be calculated.

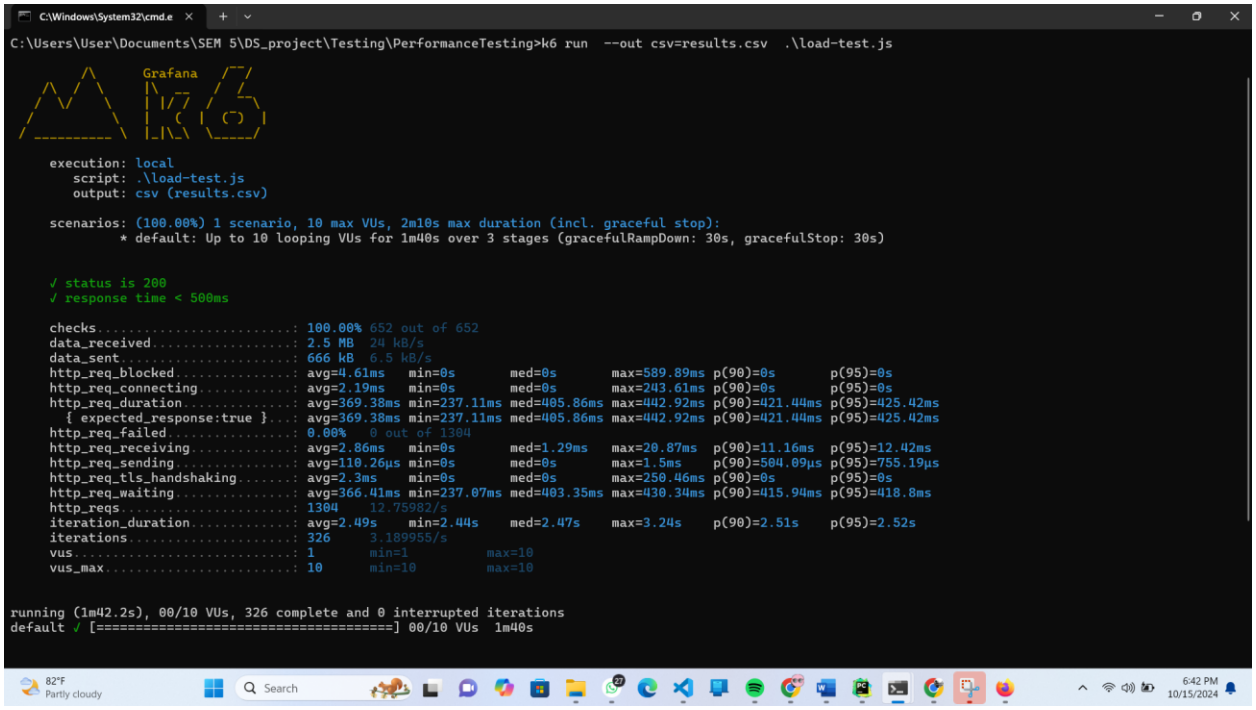



- Load Testing

**Gafana k6**, an open-source performance testing tool designed to assess the reliability and scalability of APIs, web services, and websites. It allows users to simulate a large number of virtual users (VUs) making requests simultaneously, mimicking real-world traffic,

We simulate the traffic ,by ramping up 10 users for 30sec and , hold 10 users for 1 min.




load testing script

### 4.1 Reporting on Test Coverage

Test logs will be generated daily during the development phase, as unit testing is performed regularly after each modification to a functionality. User Interface testing and code inspections will be conducted weekly throughout the development phase. During the user acceptance testing (UAT) phase, the User Interface will be tested more frequently, with feedback from users incorporated into the tests, and results will be evaluated on a daily basis.

The performance testing and load testing will be done twice a month .

A standard report for testing includes the following information:

- Date
- Logged-in user
- Test case field
- Number of test cases executed
- Number of passed tests
- Number of failed tests
- Pass percentage
- Fail percentage
- Comments

These reports should be generated for every test case, whether successful or unsuccessful.

## 5. Risks, Dependencies, Assumptions, and Constraints

| Risk | Mitigation Strategy | Contingency (Risk is realized) |
|---|---|---|
| Prerequisite entry criteria is not met. | Tester will define the prerequisites that must be met before Load Testing can start.<br><br>Customer  will endeavor to meet prerequisites indicated by Tester. | • Meet outstanding prerequisites<br>• Consider Load Test Failure |
| Test data proves to be inadequate. | Customer> will ensure a full set of suitable and protected test data is available.<br>Tester will indicate what is required and will verify the suitability of test data. | • Redefine test data<br>• Review Test Plan and modify<br>• components (that is, scripts)<br>• Consider Load Test Failure |

## 6. References


- Selenium, available at https://www.selenium.dev/selenium-ide/
- Grafana k6, available at https://k6.io/
- SonarQube, available at https://www.sonarsource.com/products/sonarqube/
- Blaze Meter, available at https://www.blazemeter.com/