Elderly Fall Detection System

Table of contents

Problem statement

Develop an application to help in the growing field of Human Activity Recognition (HAR).

App giving vital information about the status of human activities:
– Gait analysis
– Fall detection

The system must be developed in Matlab and must include a GUI with graphical simulation of the displays and buttons.

1. Introduction

(a) Scope

The "Elderly Fall Detection System" is a safety-critical software application designed to monitor the daily activity and falls of elderly individuals. This system is built using Matlab and utilizes publicly available datasets to simulate real-life scenarios. The system caters to two roles, patients and doctors, each with specific functionalities and access levels.

(b) Plan for Requirements Management

- **Information Gathering**: To gather information about the application field, we will focus on conducting group discussions. We will also review existing literature and case studies.
- **Use Public Datasets:**
    - MobiAct v2.0 Dataset which can be used for fall detection and activity recognition in elderly individuals.
- **Use Tools and Resources in MATLAB:**
    - *Signal Processing Toolbox*: For preprocessing and analyzing sensor data.
    - *Machine Learning Toolbox*: For training and evaluating classifiers for fall detection.
    - *Data Acquisition Toolbox*: For real-time data processing.
    - *App Designer*: For developing a user-friendly interface.
- **Requirements Documentation**: Documenting functional and non-functional requirements in a Software Requirements Specification (SRS) document.
- **Validation and Verification**: Regular reviews and validation sessions to ensure requirements are accurately captured and understood.

(c) Use cases

View Activity Data (Patient)

- The patient uses the Patient Web App to view:
    - Activities data
    - A number of falls were detected and fall.
    - Notifications for any detected falls.

- Fall Alert and Response:
  - Option to confirm if the fall was a false alarm or if assistance is needed.

Monitor Patient Data (Doctor)

- The doctor uses the Doctor Web App to monitor,
  - activity data for multiple patients.
  - Detailed fall incident reports including time and activity during the fall.
  - Alerts for patients with multiple falls

Receive Emergency Alerts (Doctor)

- The doctor receives an emergency alert if a patient falls.

(d) Process model

Our team will adopt a hybrid Agile-Scrum and V model for this project.

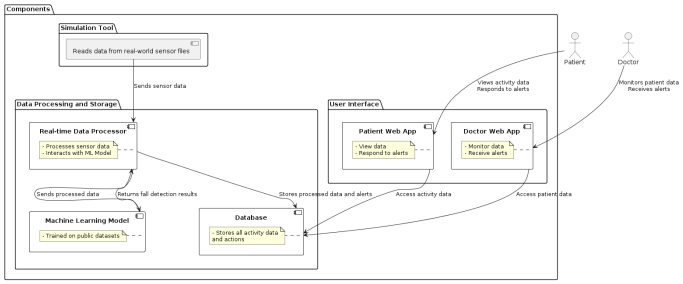
The Agile-Scrum model allows us to divide the project into smaller tasks and work on them in iterative sprints, promoting flexibility and adaptability. This model encourages regular communication among team members, allowing for immediate feedback and continuous improvement.

The V model, on the other hand, is useful for its emphasis on verification and validation. This model ensures that every stage of the software development process is thoroughly tested, which is crucial for a safety-critical system like an Elderly Care Monitoring System.

The combination of these two models will provide us with the flexibility and rigor necessary for successful project execution.

2. System Architecture

(a) Overall System Architecture

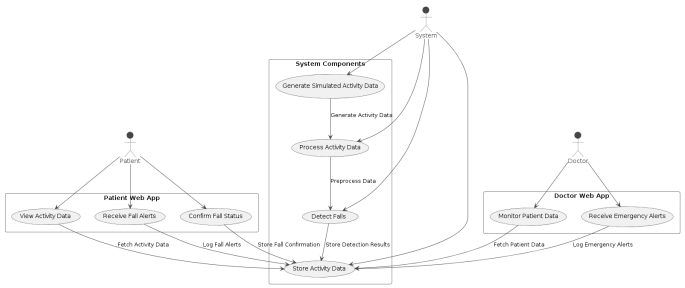


- **Actors:**
  - Patient: Uses the Patient Web App to view their activity data and receive alerts.
  - Doctor: Uses the Doctor Web App to monitor patient data and receive emergency alerts.
- **Components:**
  - Stream Data:
    - Read data from real-world sensor files.
  - Data Processing and Storage:
    - Real-time Data Processor: Processes incoming activity data and interacts with the Machine Learning Model for fall detection.
    - Machine Learning Model: Trained to detect falls using public datasets.
    - Database: Stores all activity data and system actions.
  - User Interface:
    - Patient Web App: Allows patients to view their activity data and respond to alerts.
    - Doctor Web App: Allows doctors to monitor multiple patients' data and receive alerts.

(b) UML Diagrams

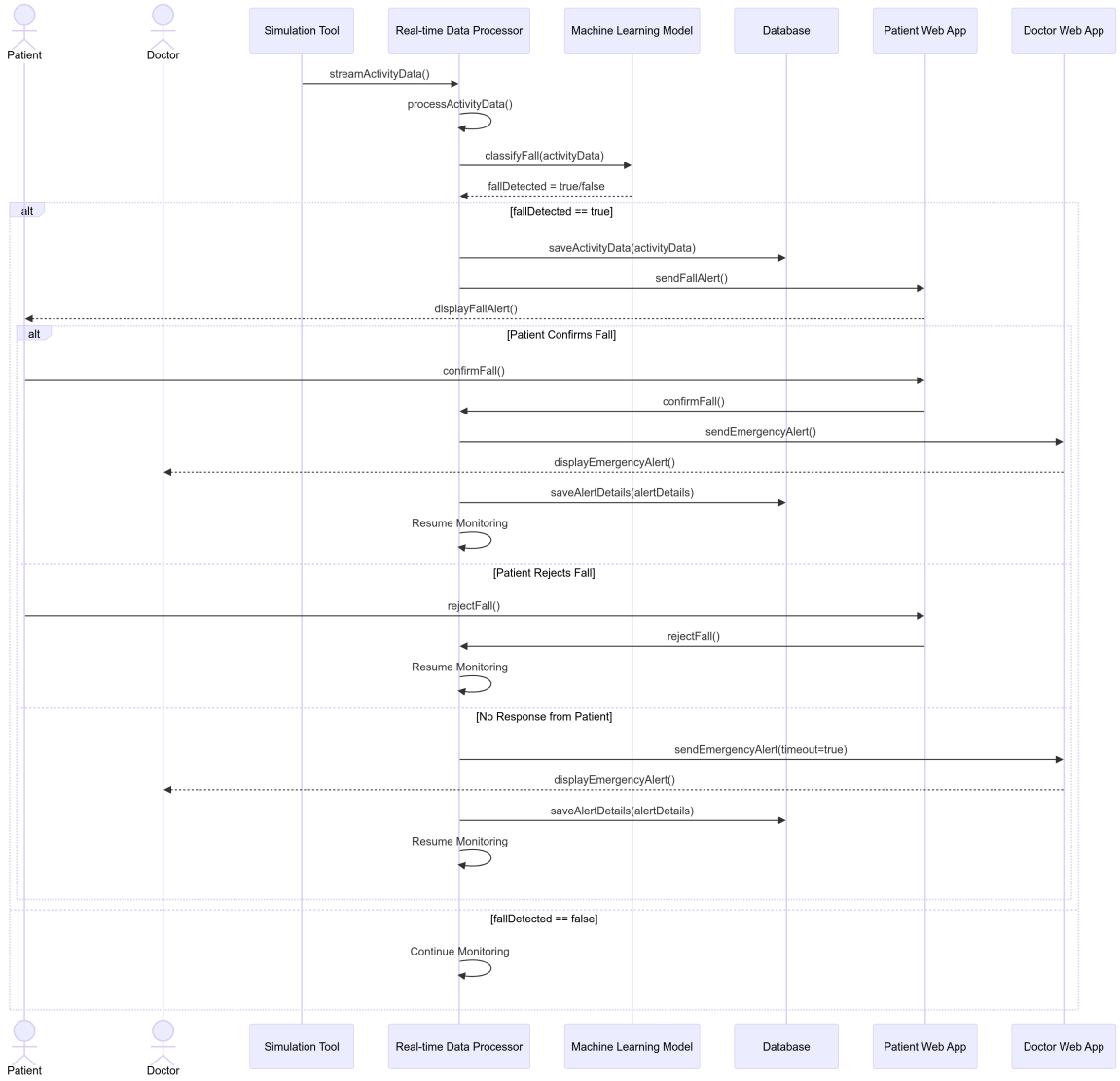
Use Case Diagram

The use case diagram shows the interaction between users and the system.



- **Actors:**
  - Patient: Interacts with the system to view activity data, receive alerts, and confirm/reject alerts.
  - Doctor: Monitors patient data and receives emergency alerts.
- **Use Cases:**
  - View Activity Data: Patients can view their activity data.
  - Receive Alerts: Patients receive alerts for detected falls.
  - Confirm/Reject Alerts: Patients confirm or reject fall alerts.
  - Monitor Patients: Doctors monitor activity data for multiple patients.
  - Receive Emergency Alerts: Doctors receive alerts for confirmed falls or if the patient does not respond to the alert in a certain timeframe.

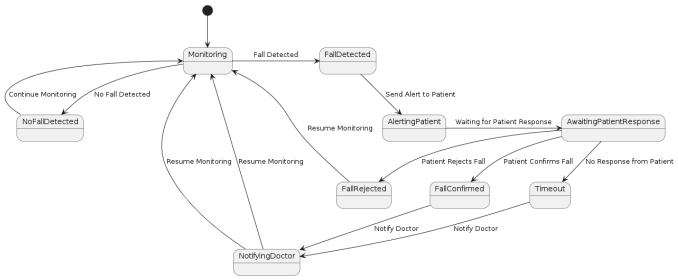
Sequence Diagram



- **Flow:**
  1. Stream Data:
     - The Data Steamer streams activity data to the Real-time Data Processor .
  2. Processing Data:
     - The Processor processes the data and sends it to the Machine Learning Model for fall detection.
     - The Machine Learning Model returns whether a fall is detected.
  3. Handling Detected Fall:
     - If a fall is detected,
     - The Processor sends a fall alert to the Patient Web App.
     - The Patient Web App displays the fall alert to the patient.
     - Depending on the patient's response:
       - If the patient confirms the fall, the processor sends an emergency alert to the doctor web app, the doctor web app displays the alert, and Monitoring then resumes.
       - If the patient rejects the fall, monitoring resumes without notifying the doctor.
       - Suppose there is no response from the patient within a timeout period. In that case, the Processor sends an emergency alert to the Doctor Web App , the Doctor Web App displays the crtitcal alert , and Monitoring then resumes.
  4. No Fall Detected:
     - If no fall is detected, the Processor continues monitoring.

State Transition Diagram

This diagram represents the states an object goes through in response to events, particularly for the fall detection and emergency alert feature.



- **States:**
  - Monitoring: The system is monitoring patient activity data.
  - FallDetected: A fall is detected by the system.
  - AlertingPatient: The system sends an alert to the patient.
  - AwaitingPatientResponse: The system is waiting for the patient's response to the alert.
  - FallConfirmed: The patient confirms the fall.
  - NotifyingDoctor: The system notifies the doctor about the confirmed fall.
  - FallRejected: The patient rejects the fall alert.
  - Timeout: No response from the patient within the specified time.
- **Transitions:**
  - From Monitoring to FallDetected: When a fall is detected.
  - From FallDetected to AlertingPatient: The system sends an alert to the patient.
  - From AlertingPatient to AwaitingPatientResponse: The system waits for the patient's response.
  - From AwaitingPatientResponse to FallConfirmed: The patient confirms the fall.
  - From FallConfirmed to NotifyingDoctor: The system notifies the doctor about the fall.
  - From NotifyingDoctor to Monitoring: The system resumes monitoring after notifying the doctor.
  - From AwaitingPatientResponse to FallRejected: The patient rejects the fall alert.
  - From FallRejected to Monitoring: The system resumes monitoring after the fall alert is rejected.
  - From AwaitingPatientResponse to Timeout: The patient does not respond within the specified time.
  - From Timeout to NotifyingDoctor: The system notifies the doctor due to the timeout.
  - From Monitoring to NoFallDetected: No fall is detected.
  - From NoFallDetected to Monitoring: The system continues monitoring.

3. Data Analysis Part of the Project

It involves processing of data to detect falls accurately and provide meaningful insights to patients and doctors.

Steps Involved:

1. **Data Loading:**
   - Load and preprocess data from directories.
2. **Label mapping:**
   - Mapped labels with Numeric.
3. **Data Normalization:**
   - Cleanse the data to remove any anomalies or inconsistencies.
   - Normalize the data to ensure uniformity.
4. **Data Segmentation:**
   - Segment the data into fixed-size windows with 50% overlap.
5. **Features Extraction:**
   - Extract relevant features from the preprocessed data, such as Time-domain features, Orientation features, and Frequency-domain features (using FFT).
   - Combine all features into a single row vector.
   - Saved combined data into a CSV file for final verification.
6. **Model Traning:**
   - Split data into training (70%) and testing (30%) sets
   - Compute class weights
   - Hyperparameter tuning with grid search

- Train a Random Forest model with class weights
- Save the best model

5. Project Estimates

(a) Function Point Analysis (FPA)

We will use the Function Point (FP) method to estimate the size and complexity of the software.

Function Point Calculation

1. **External Inputs (EI)**: User inputs that the system processes.
   - Patient confirmation of fall
   - Patient rejection of fall
   - Streaming activity data from the Simulation Tool
2. **External Outputs (EO)**: User outputs generated by the system.
   - Display fall alert to patient
   - Notify the doctor of the fall
   - Display emergency alert to doctor
3. **External Inquiries (EQ)**: User-initiated inquiries that do not change the system state.
   - View activity data (patient)
   - View patient data (doctor)
4. **Internal Logical Files (ILF)**: Logical groups of data maintained within the system.
   - Activity data
   - Alert details
5. **External Interface Files (EIF)**: Logical groups of data used for reference that are maintained by other systems.
   - Public datasets for training the Machine Learning Model

Assigning Complexity Weights

For simplicity, we'll assume all functions have average complexity.

Function Type	Count	Complexity Weight (Average)	Total FP
External Inputs (EI)	3	4	12
External Outputs (EO)	3	5	15
External Inquiries (EQ)	2	4	8
Internal Logical Files (ILF)	2	7	14
External Interface Files (EIF)	1	5	5

Total Function Points Calculation

Total Unadjusted Function Points (UFP) = 54

Value Adjustment Factor (VAF)

VAF = 0.65 + (sum of degree of influence) / 100

Assuming a typical degree of influence summing to 35,
VAF = 0.65 + 0.35 = 1.00

Adjusted Function Points (AFP)

AFP = UFP * VAF = 54 * 1.00 = 54

(b) COCOMO II Estimation

Using the COCOMO II model, we can estimate the effort required to complete the project based on the function point count.

Source Lines of Code (SLOC) Conversion

The conversion factor from function points to SLOC.

For MATLAB, a typical conversion factor is around 100 SLOC per function point.

Adjusted SLOC Calculation:

- SLOC = AFP * MATLAB Conversion Factor
- SLOC = 54 * 100 = 5.4 KSLOC

Effort Calculation

Using the COCOMO II model, Let's calculate the effort required.

COCOMO II formula:
{Effort (Person-Months)} = A * [{SLOC}}^B * E]

For MATLAB and nominal parameters:

- A=2.94: This is the base constant for the COCOMO II model in nominal mode.
- B = 1.05B=1.05: This exponent reflects the scale factor.

- E = 1.0E=1.0: This indicates a nominal effort multiplier, assuming typical project attributes.

Effort = 2.94 * (5.4 KSLOC)^1.05 * 1.0 ≈ **18.79 person-months**.

Team and Duration

Given 8 team members:

Duration (Months) = Effort / Number of Team Members = 18.79 / 8 ≈ 10 Weeks

(c) Final Summary

- **Function Points:** 54
- **Estimated Effort:** 18.79 Person-Months
- **Project Duration:** 10 weeks with 8 team members

6. Risk Management

Risk ID	Risk Description	Impact	Likelihood	Priority	Mitigation Strategy	Monitoring	Management
R1	Resource allocation issues	High	Medium	High	Clear task assignments, monitor workload	Weekly resource reviews	Adjust resource allocation
R2	Delays in task completion	High	High	High	Detailed timeline, regular progress reviews	weekly progress reviews	Adjust tasks and resources as needed
R3	Scope creep	High	Medium	High	Define clear project scope, change management process	Review scope regularly	Approve changes through a formal process
R4	Communication breakdowns	Medium	Medium	Medium	Regular team meetings, clear communication channels	Weekly meetings	Use project management tools
R5	Data security breaches	High	Low	Low	follow best practices	Regular security audits	Ensure compliance with data protection
R6	Usability challenges for the elderly	Medium	High	Low	simplify UI	Usability testing	Iterate based on feedback

7. Safety Plan

(a) Failure Modes and Effects Analysis (FMEA)

Component/Function	Potential Failure Mode	Potential Effect(s) of Failure	Severity (S)	Potential Cause(s)/Mechanism(s) of Failure	Occurrence (O)	Current Design Controls	Detection (D)	RPN (S x O x D)	Recommended Actions
StreamData	Corrupt sensor dataset in files	Inaccurate data	7	Software bugs, incorrect algorithms	5	Code reviews	4	140	Use of real-world dataset and stream those data
Real-time Data Processor	Data Processing Delay	Delayed fall detection	8	High data load, inefficient algorithms	4	Performance optimization	3	96	Optimize code
Machine Learning Model	False Positive Fall Detection	Unnecessary emergency alerts	6	Model sensitivity, insufficient training data	6	Model tuning, extensive training	3	108	Improve training dataset, refine model
	False Negative Fall Detection	Missed falls, delayed medical response	9	Model limitations, unrepresentative training data	4	Model validation, use of diverse datasets	4	144	Enhance model, increase training diversity
Database	Data Storage Failure	Loss of activity data	8	software bugs	3	Data redundancy, regular backups	2	48	Implement a robust backup and recovery plan
	Data Retrieval Delay	Slow response times for users	7	High data load, inefficient queries	4	Query optimization	3	84	Optimize database queries
Patient Web App	Poor Usability	User frustration, reduced usage	5	Complex design, inadequate user testing	5	Usability testing, user feedback	3	75	Conduct usability testing
	Fall Alert Not Received	Missed alerts, delayed response	9	Network issues, app bugs	3	Network monitoring, error logging	3	81	Improve network reliability
Doctor Web App	Inaccurate Data Display	Misleading information, incorrect decisions	8	Data synchronization issues, display bugs	4	Data validation, error logging	3	96	Improve data synchronization
	Alert Not Received	Missed emergency alerts, delayed response	10	Network issues, app bugs	3	Network monitoring, error logging	2	60	Implement redundant alerting mechanisms

(b) STAMP/STPA analysis

ID	Control Action	Hazardous State	Potential Causes	Loss Scenario	Safety Constraint	Mitigation Measures
CA1	Data stream	HS1: Inaccurate stream data or Unable to find files or Files consist incorrect data	PC1: Software bugs, incorrect algorithms	LS1: stream inaccurate data	SC1: Files data must be accurate	MM1: improve data stream function
CA2	Data processing by Real-time Data Processor	HS2: Delayed fall detection	PC2: High data load, inefficient algorithms	LS2: Real-time Data Processor delays processing	SC2: Data processing must be timely	MM2: Optimize code MM3: use efficient algorithms
CA3	Data Cleansing by Real-time Data Processor	HS3: Incorrect or incomplete data	PC3: Faulty data cleansing logic	LS3: Real-time Data Processor fails to cleanse	SC3: Data cleansing must ensure accuracy	MM4: Implement rigorous data validation
CA4	Fall Detection by Machine Learning Model	HS4: False positives/negatives	PC4: Model sensitivity, insufficient training data	LS4: ML Model misclassifies falls	SC4: Fall detection must be accurate	MM4: Improve training datasets MM5: refine model MM6: validate model performance
CA5	Data Storage by Database	HS5: Data loss	PC5: Hardware failure, software bugs	LS5: Database loses data	SC5: Data must be securely stored	MM5: Implement robust backup and recovery plans
CA6	Data Retrieval by Database	HS6: Slow response times	PC6: High data load, inefficient queries	LS6: Database retrieval is slow	SC6: Data retrieval must be efficient	MM6: Optimize queries MM7: index database tables
CA7	Alert delivery by Patient Web App	HS7: Missed or delayed alerts	PC7: Network issues, app bugs	LS7: Patient Web App misses alerts	SC7: Alerts must be delivered in real-time	MM8: Improve network reliability MM9: implement error handling
CA8	Data display by Doctor Web App	HS8: Inaccurate or incomplete display	PC8: Data synchronization issues, display bugs	LS8: Doctor Web App displays incorrect data	SC8: Data display must be accurate	MM10: Improve data synchronization
CA9	Unauthorized access to patient data	HS9: Data breach	PC9: Security loopholes, weak authentication	LS9: Unauthorized access to patient data	SC9: Patient data must be protected from unauthorized access	MM11: implement strong authentication
CA10	Poor usability of Patient/Doctor Web App	HS10: User frustration, reduced usage	PC10: Complex design, inadequate user testing	LS10: Patient/ Doctor Web App is not user-friendly	SC10: The interface must be user-friendly	MM12: Conduct usability testing MM13: gather user feedback MM14: iterate on UI designs

System Safety Requirements

1. **Accurate Fall Detection**: Ensure high accuracy in detecting falls using robust algorithms and diverse training datasets.
2. **Timely Alerts**: Ensure the system sends alerts to patients and doctors in real-time.
3. **Data Integrity**: Maintain the integrity of the collected and processed data.
4. **Usability**: Design the system to be user-friendly, particularly for elderly users.

Safety Measures

1. **Data Validation and Cleansing**:
   - Implement rigorous data validation and cleansing processes to ensure data accuracy.
   - Use noise filtering techniques to preprocess sensor data.
2. **Algorithm Optimization**:
   - Continuously refine fall detection algorithms to minimize false positives and negatives.
   - Train models with comprehensive datasets to improve detection accuracy.
3. **System Monitoring**:
   - Monitor system performance regularly to detect and rectify any anomalies.
   - Conduct regular code reviews and performance testing.

Safety Testing

1. **Unit Testing**: Validate each component individually.
2. **Integration Testing**: Ensure all components work together seamlessly.
3. **System Testing**: Test the entire system.
4. **User Testing**: Conduct usability testing with elderly users to ensure the interface is intuitive.

8. Security Plan

(a) Security Analysis Using STPA-SafeSec

STPA-SafeSec integrates security analysis into the STPA (System-Theoretic Process Analysis) framework. The goal is to identify potential security threats and establish corresponding security requirements and constraints for each hierarchy level and component of the system.

System Level

Level	Potential Security Threats	Security Requirements	Security Constraints
System Level	Unauthorized access to the system	Ensure authorized access only	Implement multi-factor authentication (MFA)
	Data interception during transmission	Ensure data confidentiality during transmission	Use end-to-end encryption for data in transit

Component Level

Component	Potential Security Threats	Security Requirements	Security Constraints
Simulation Tool	Injection of malicious data	Validate and sanitize input data	Implement strict input validation and sanitization
Real-time Data Processor	Unauthorized access to processing functions	Restrict access to authorized entities	Implement role-based access control (RBAC)
Machine Learning Model	Model poisoning attacks	Ensure model training data is clean and secure	Validate and sanitize training data, monitor for anomalies
Database	Unauthorized access to stored data	Ensure data confidentiality	Implement access control
	Data breaches	Ensure data integrity and confidentiality	Implement access control
Web App	Unauthorized access to patient data	Ensure patient data privacy	Implement strong authentication and session management
	Cross-site scripting (XSS) attacks	Prevent XSS attacks	sanitize inputs
	Data leakage	Prevent data leakage	Use secure data transmission and storage mechanisms

System Security Requirements

1. **Data Confidentiality**: Protect sensitive patient data from unauthorized access.
2. **Data Integrity**: Ensure data is not altered or tampered with.
3. **Authentication and Authorization**: Implement robust user authentication and role-based access control.

Security Measures

1. Implement strict input validation
2. Implement role-based access control
3. Sanitize inputs

9. Roadmap and Schedule with Milestones

Week	Phase	Team Members Involved	Key Activities	Milestones	Status
1	Setup and Preliminary Development	All	Set up MATLAB environment Develop Simulation Tool, Simulation tool function to read data	Environment setup ,Simulation Tool	Done
2	Core System Development	1-2	Develop Real-time Data Processor	Real-time Data Processor development starts	Done
3	Core System Development	3-4	Implement Machine Learning Model	Machine Learning Model implementation starts	Done
4	Core System Development	5-6	Train and integrate Machine Learning Model	Machine Learning Model training complete	Done
5	Core System Development	7-8	Integrate Simulation Tool, Real-time Data Processor, ML Model	Core system integration starts	Done
6	Web Application Development	1-2-3-4	Develop a Patient Web App with role-based access control	Patient Web App development starts	
7	Integration and Testing	All	Integrate all components Conduct unit testing Perform system testing with simulated data	Integration starts Unit testing complete System testing complete	
8	Deployment and Evaluation	All	Deploy system	System deployment complete	

main.m Script Documentation

streamData.m Function Documentation

processData Functions Documentation

Data Preprocessing and Feature Extraction Documentation

Model Training and Evaluation Documentation