Software Design Document

SDD

Project name: LifeBridge

Developers: Nitzan Jarus, Firas Shehady, Bara Haj

Introduction

System Overview

Falls are one of the most common and dangerous incidents among elderly individuals, often leading to serious injuries, long-term complications, or delayed medical response. While various fall detection solutions exist, many rely on cloud-based processing, continuous data transmission, or intrusive monitoring methods that raise privacy concerns and limit usability.

LifeBridge is a local-first fall detection support system designed to address these challenges by providing continuous monitoring, real-time classification, and alert generation without transmitting raw sensor data outside the local environment. The system focuses on early identification of potential fall events using accelerometer-based data, processed entirely on the user's local machine. By prioritizing privacy, responsiveness, and simplicity, LifeBridge aims to support elderly safety while minimizing system complexity and user intervention.

The system operates by simulating or receiving motion sensor data, classifying activity patterns in real time, detecting abnormal motion signatures associated with falls, and presenting the system state clearly through a lightweight user interface. When a potential fall is detected, the system escalates the event and enables alert handling while maintaining strict control over stored data.

Purpose

The purpose of this Software Design Document (SDD) is to describe the internal design, architecture, and implementation decisions of the LifeBridge system, based on the approved Software Requirements Specification (SRS). This document translates functional and non-functional requirements into concrete architectural components, data flows, and algorithmic logic, serving as a bridge between system requirements and implementation.

The SDD is intended to:

- Provide a clear architectural overview of LifeBridge
- Describe the responsibilities and interactions between system components
- Define how fall detection logic is implemented and evaluated
- Demonstrate traceability between requirements, use cases, and design decisions
- Support future implementation, testing, and validation efforts

Scope

This document covers the design of the LifeBridge fall detection system as a standalone, locally operating prototype. The scope includes:

- Simulation and processing of accelerometer-based motion data
- Real-time classification of activity states
- Detection of potential fall events based on configurable thresholds
- Local data handling and minimal persistence of classified results
- A passive user interface that reflects system state and alerts
- Optional external alert integration, triggered only upon fall detection

Out of scope for this version are:

- Continuous cloud connectivity
- Long-term medical analytics
- Integration with wearable hardware beyond simulated input

- Automated emergency response services

Constraints

The design of LifeBridge is influenced by several constraints:

- Privacy constraints: Raw motion sensor data must not be transmitted externally or persistently stored.
- Local execution: All processing and classification must occur on a single local machine.
- Performance constraints: Fall detection must occur with minimal latency to enable timely response.
- Usability constraints: The system must operate passively without requiring constant user interaction.
- Prototype limitations: The system relies on simulated data inputs to represent sensor behaviour.

These constraints guided the architectural and algorithmic decisions described in the following sections.

System Architecture

Architectural Description and Design: Roles, Activities and Data

Architectural Description and Design:

LifeBridge follows a modular, layered architecture that separates data acquisition, processing, storage, and presentation concerns. The system is designed to operate continuously in the background while maintaining clear internal boundaries between responsibilities.

Roles

Although LifeBridge operates primarily as an automated system, the following conceptual roles are defined for clarity:

- User (Elderly Individual)
  The individual whose motion data is monitored. The user does not actively interact with the system during normal operation.
- System
  Responsible for data simulation or ingestion, classification, fall detection, state management, and alert triggering.
- Caregiver / Observer (Optional)
  Receives alerts or system status updates in the event of a detected fall.

Activities

The core system activities include:

- Motion Data Generation / Ingestion
  The system generates simulated accelerometer data representing normal and abnormal motion patterns.
- Real-Time Classification
  Incoming data is continuously analysed to classify the current activity state (e.g., normal movement, inactivity, abnormal spike).
- Fall Detection Logic
  Classification results are evaluated against configurable thresholds to identify potential fall events.
- Event Handling and Escalation
  When a potential fall is detected, the system updates its internal state and triggers alert mechanisms.

- Visualization and Feedback
  System status and alerts are reflected in the user interface using clear visual indicators.

Data

LifeBridge handles the following categories of data:

- Transient Motion Data
  Time-series accelerometer values processed in memory only.
- Classified Events
  Lightweight records representing detected states or incidents, stored locally for short-term reference.
- System Configuration Data
  Threshold values and detection parameters adjustable within defined bounds.

No raw motion data is persistently stored, ensuring compliance with privacy requirements.

Main System Internal Scenario

The primary internal scenario describes the system’s continuous monitoring loop during normal operation:

Motion data is generated or received by the Data Simulation Module.

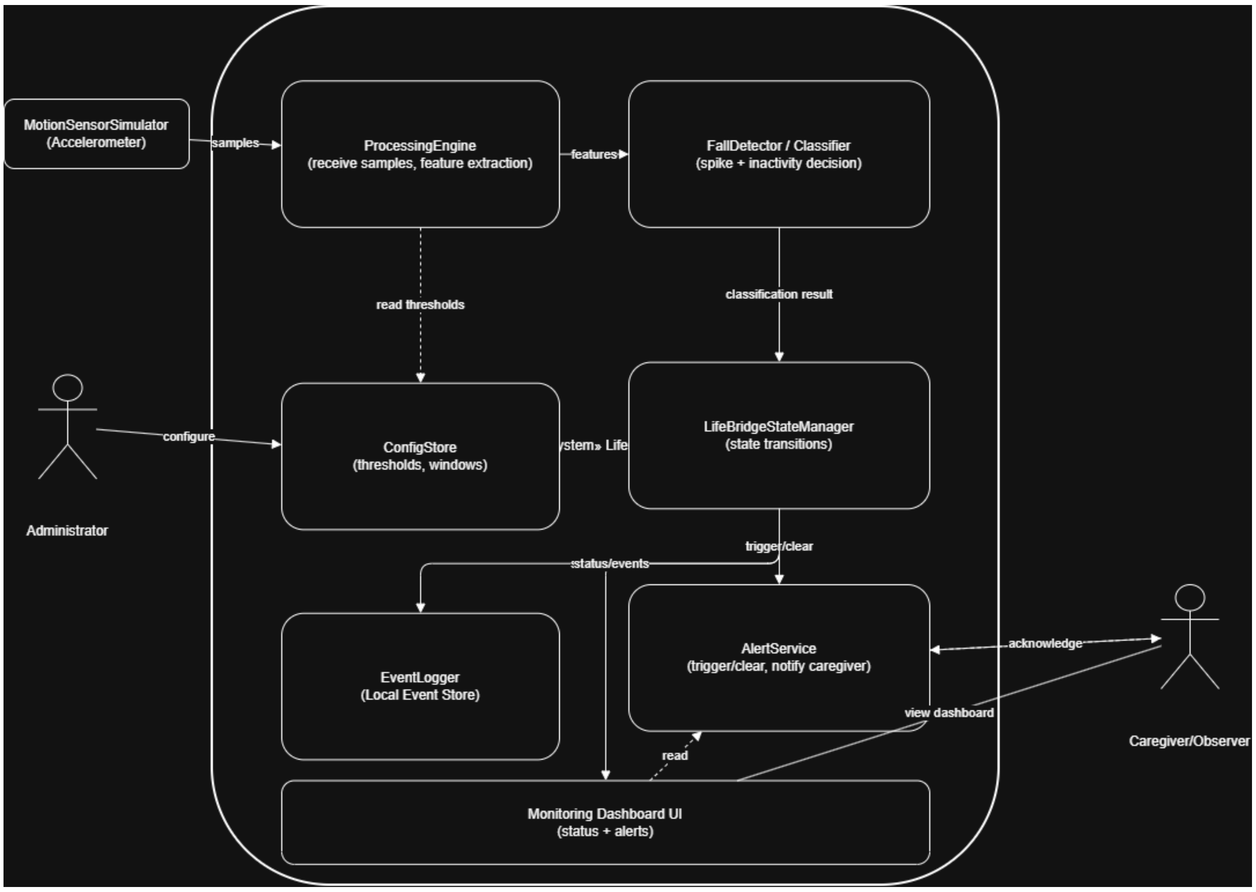
Data is forwarded to the Classification and Processing Engine.

The engine analyses the data stream and assigns activity classifications.

Classified results are evaluated by the Fall Detection Logic.

The system updates its internal state and user interface accordingly.

If a potential fall is detected, an alert event is generated.


MotionSensorSimulator (Accelerometer)
samples
ProcessingEngine (receive samples, feature extraction)
features
FallDetector / Classifier (spike + inactivity decision)
read thresholds
classification result
Administrator
configure
ConfigStore (thresholds, windows)
ystem» Life
LifeBridgeStateManager (state transitions)
trigger/clear
status/events
EventLogger (Local Event Store)
AlertService (trigger/clear, notify caregiver)
acknowledge
view dashboard
Caregiver/Observer
read
Monitoring Dashboard UI (status + alerts)

Additional System Internal Scenarios

Additional internal scenarios include:

- **False Alarm Prevention**:
  Abnormal spikes without subsequent inactivity are classified but not escalated.
- **System Idle State**:
  Prolonged inactivity without abnormal motion is treated as a non-fall scenario.
- **Configuration Adjustment**:
  Threshold parameters are updated without interrupting system operation.

Design

Data Design – Database Description

LifeBridge follows a minimalistic data design approach, driven by privacy, performance, and local-first constraints. The system deliberately avoids long-term storage of raw sensor data and instead maintains lightweight records that represent classified events and system states.

Data Entities

Classified Event

- event_id – unique identifier
- timestamp – time of classification
- event_type – normal activity / abnormal motion / potential fall
- confidence_level – calculated confidence score
- alert_triggered – boolean flag

These records are stored locally and are intended solely for short-term reference, debugging, and system validation.

System Configuration

- Detection thresholds
- Time windows for inactivity detection
- Sensitivity parameters

Configuration data can be adjusted without requiring system restart.

Data Handling Policy

- Raw accelerometer data is processed in-memory only
- No continuous sensor streams are persisted
- Classified results are stored in a local, lightweight format
- Users can clear all stored records at any time

This data design directly supports privacy-related functional and non-functional requirements defined in the SRS

Data Flow – Data Flow Between System Modules

The LifeBridge system follows a linear and deterministic data flow designed to minimize latency and reduce unnecessary processing overhead.

Primary Data Flow:

1. Motion data is generated by the Data Simulation Module
2. Data is passed to the Processing and Classification Engine
3. Classification results are forwarded to the Fall Detection Logic
4. Events are either logged locally or escalated
5. System state updates are reflected in the UI and alert mechanisms

[Insert diagram here]

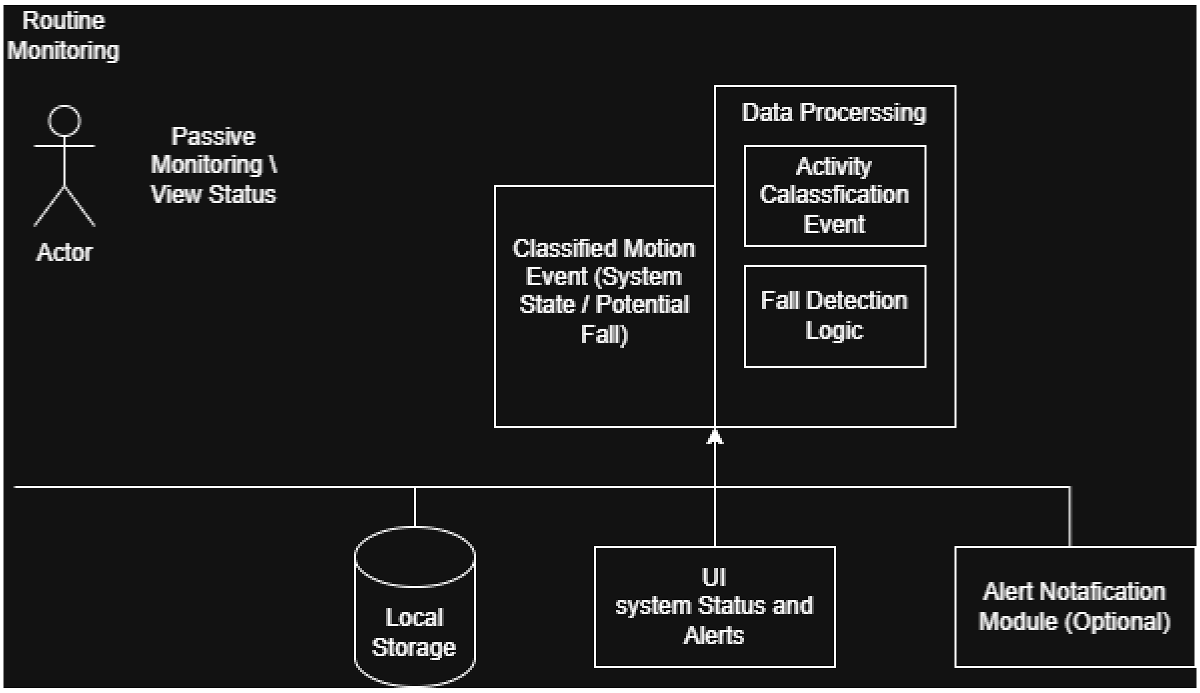
This design ensures predictable behaviour, ease of debugging, and clear separation of concerns between system modules.

Structural Design – Class Diagram

LifeBridge does not rely on a complex object-oriented domain model. Instead, the system is composed of loosely coupled components that communicate through well-defined interfaces and data structures.

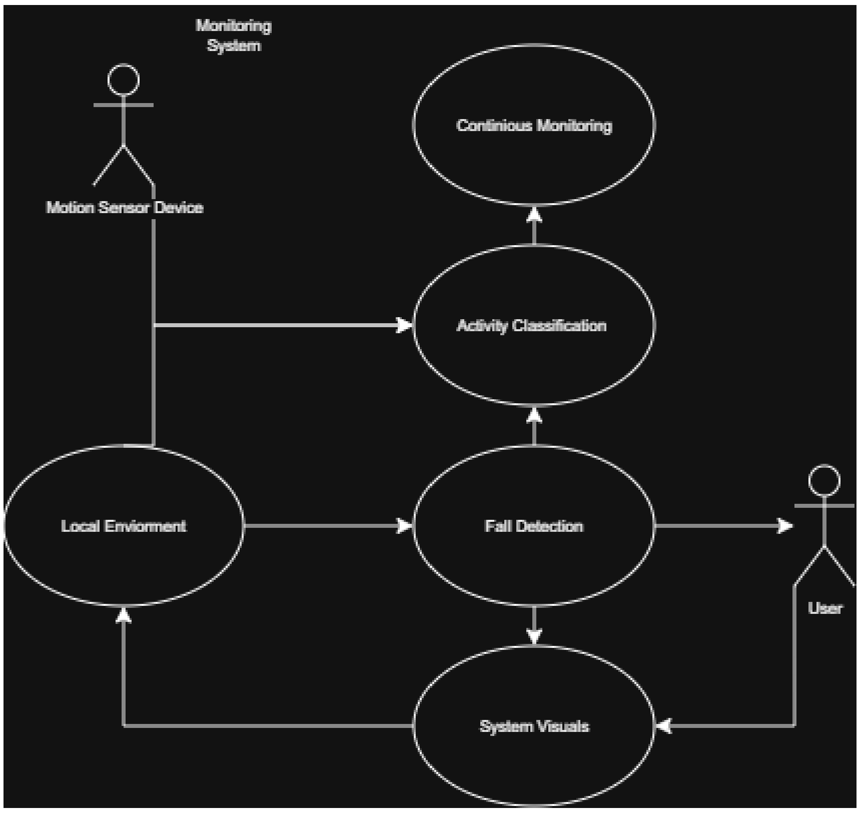
As a result, a traditional class diagram is not essential for understanding system behaviour. The design emphasizes functional components rather than deep inheritance hierarchies.

Interactions Design

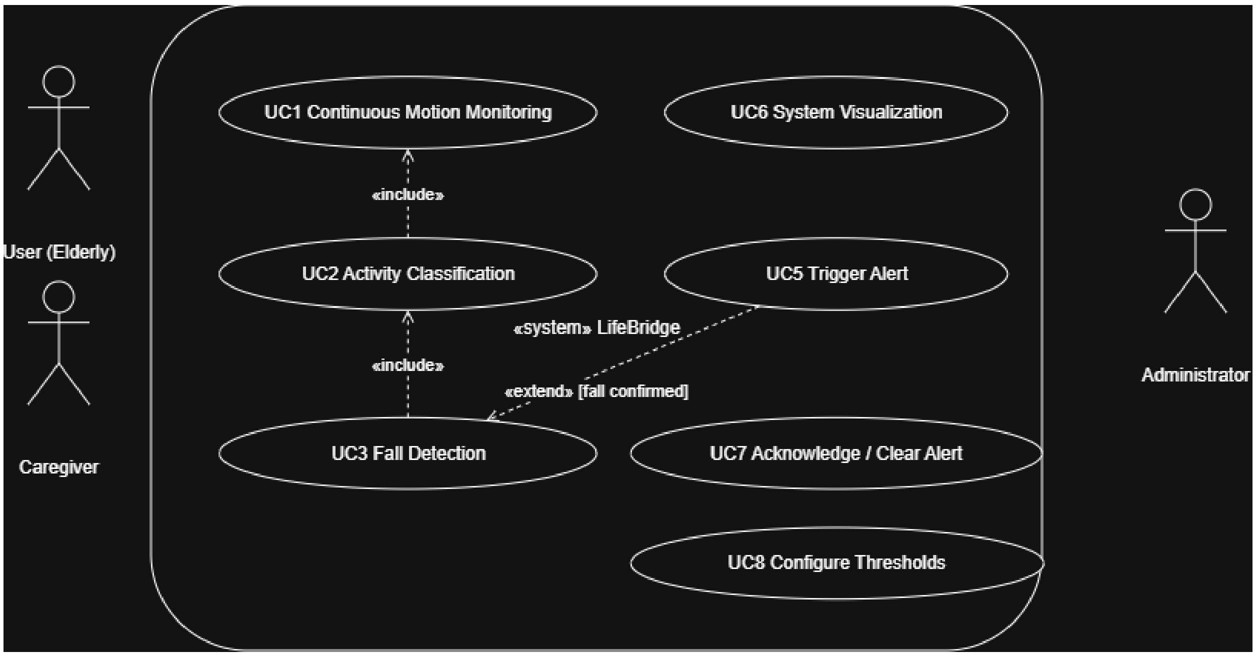

Routine Monitoring
Actor
Passive Monitoring \ View Status
Data Procerssing
Activity Calassfication Event
Fall Detection Logic
Classified Motion Event (System State / Potential Fall)
Local Storage
UI system Status and Alerts
Alert Notafication Module (Optional)

Use Cases

3.4.1 Use Case Diagram



The system supports several core use cases derived directly from the SRS. These use cases describe system behaviour rather than user-driven interaction flows.



UC1 – Continuous Motion Monitoring

- The system continuously processes incoming motion data.
- Activity states are classified in real time.

UC2 – Activity Classification

- Motion data is analysed and assigned to predefined activity categories.

UC3 – Fall Detection

- Abnormal motion patterns followed by inactivity trigger a potential fall event.

UC4 – Local Event Logging

- Classified events are recorded locally without storing raw data.

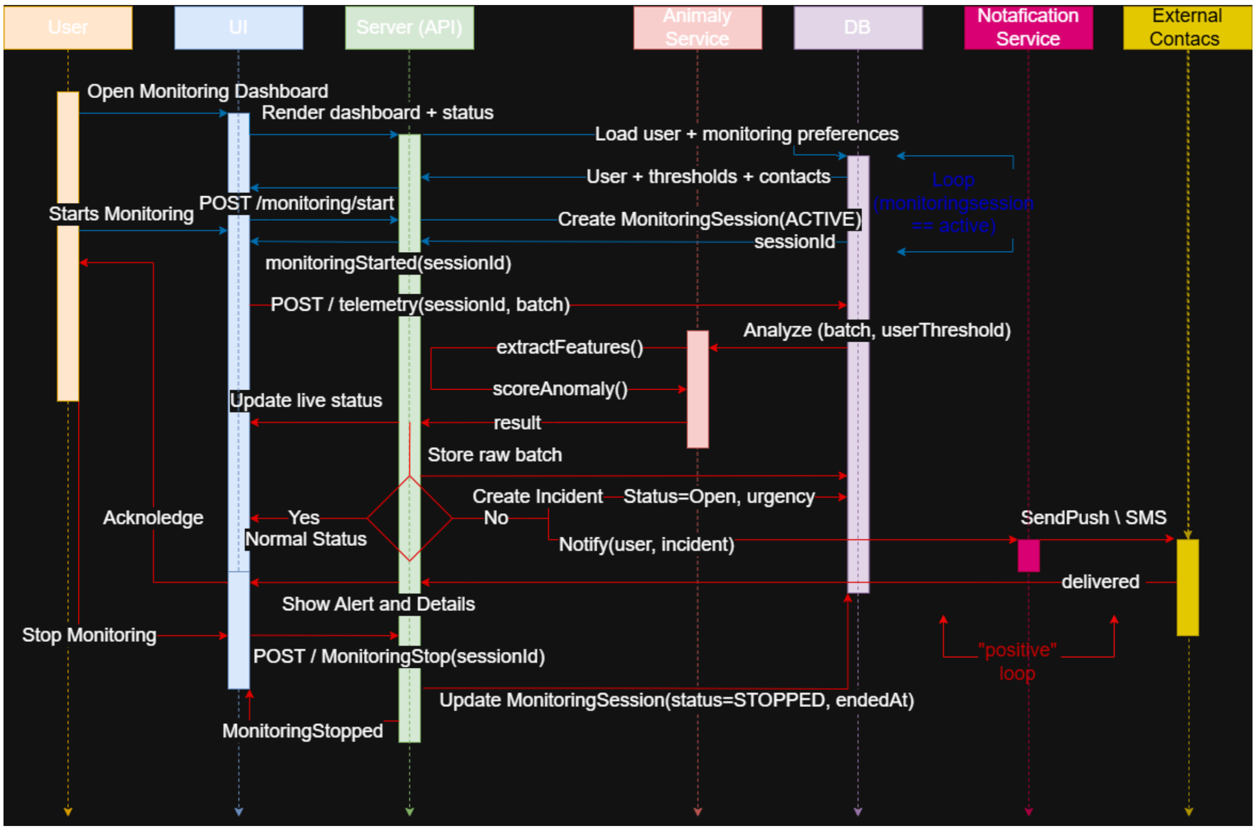
UC5 – Alert Triggering

- A detected fall escalates to an alert event.

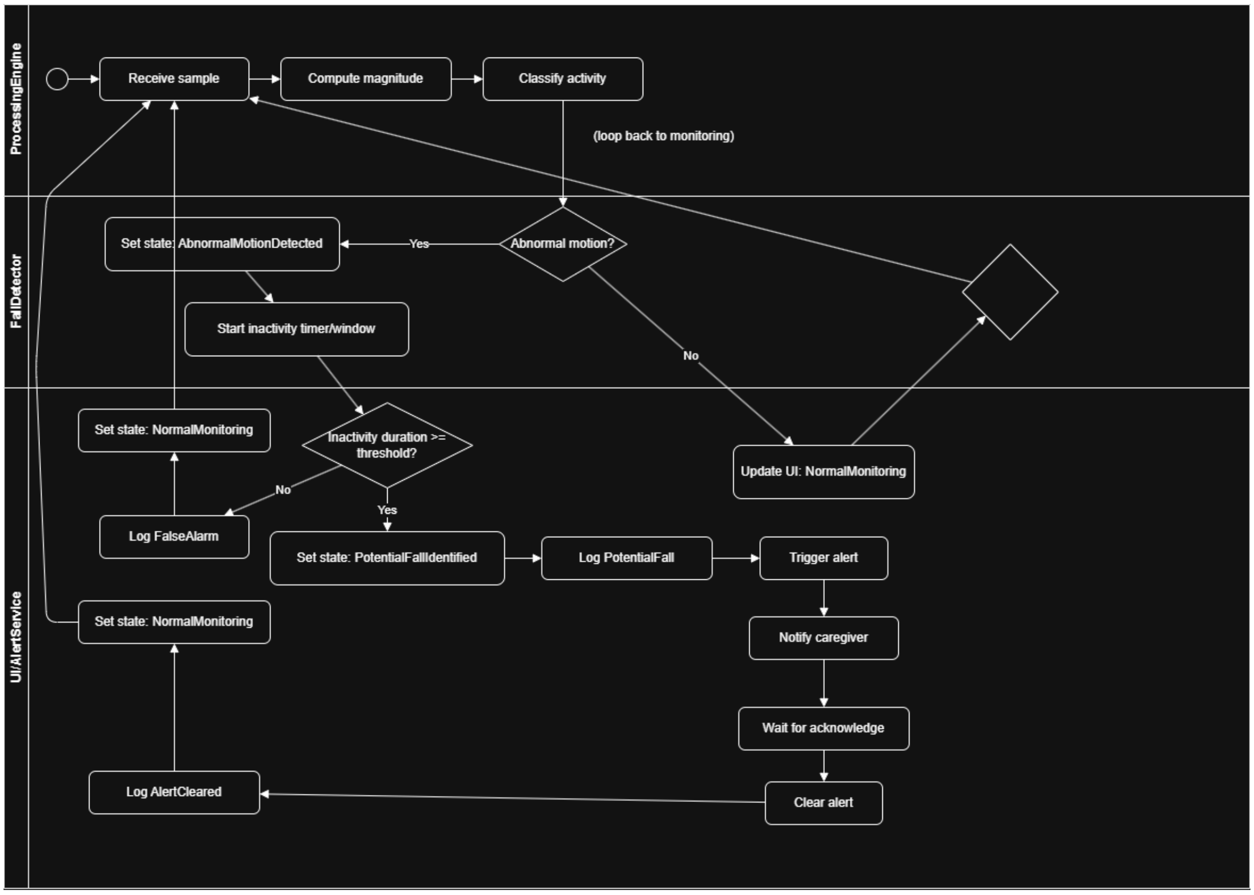
UC6 – System Visualization

- System state and alerts are displayed passively in the UI.

3.4.2 Sequence Diagram



3.4.3 Activity Diagram


ProcessingEngine
FallDetector
UI/AlertService
Receive sample
Compute magnitude
Classify activity
(loop back to monitoring)
Set state: AbnormalMotionDetected
Yes
Abnormal motion?
Start inactivity timer/window
No
Set state: NormalMonitoring
Inactivity duration >= threshold?
Update UI: NormalMonitoring
No
Yes
Log FalseAlarm
Set state: PotentialFallIdentified
Log PotentialFall
Trigger alert
Set state: NormalMonitoring
Notify caregiver
Wait for acknowledge
Log AlertCleared
Clear alert

Description of Algorithmic Components

This section describes the core logic that enables fall detection within LifeBridge.

Fall Detection Algorithm – Overview

The fall detection algorithm is based on identifying a combination of:

- Sudden acceleration spikes
- Followed by a period of abnormal inactivity

This two-stage approach reduces false positives caused by normal daily activities.

Algorithm Flow

1. Incoming accelerometer data is continuously sampled
2. Magnitude values are calculated from multi-axis input

3. Sudden deviations beyond a defined threshold are marked as abnormal motion

4. A time window is initiated to monitor subsequent activity

5. If inactivity persists beyond a defined duration, a potential fall is identified

6. The event is escalated and logged

Thresholds and time windows are configurable to allow system tuning and experimentation.

Design Rationale

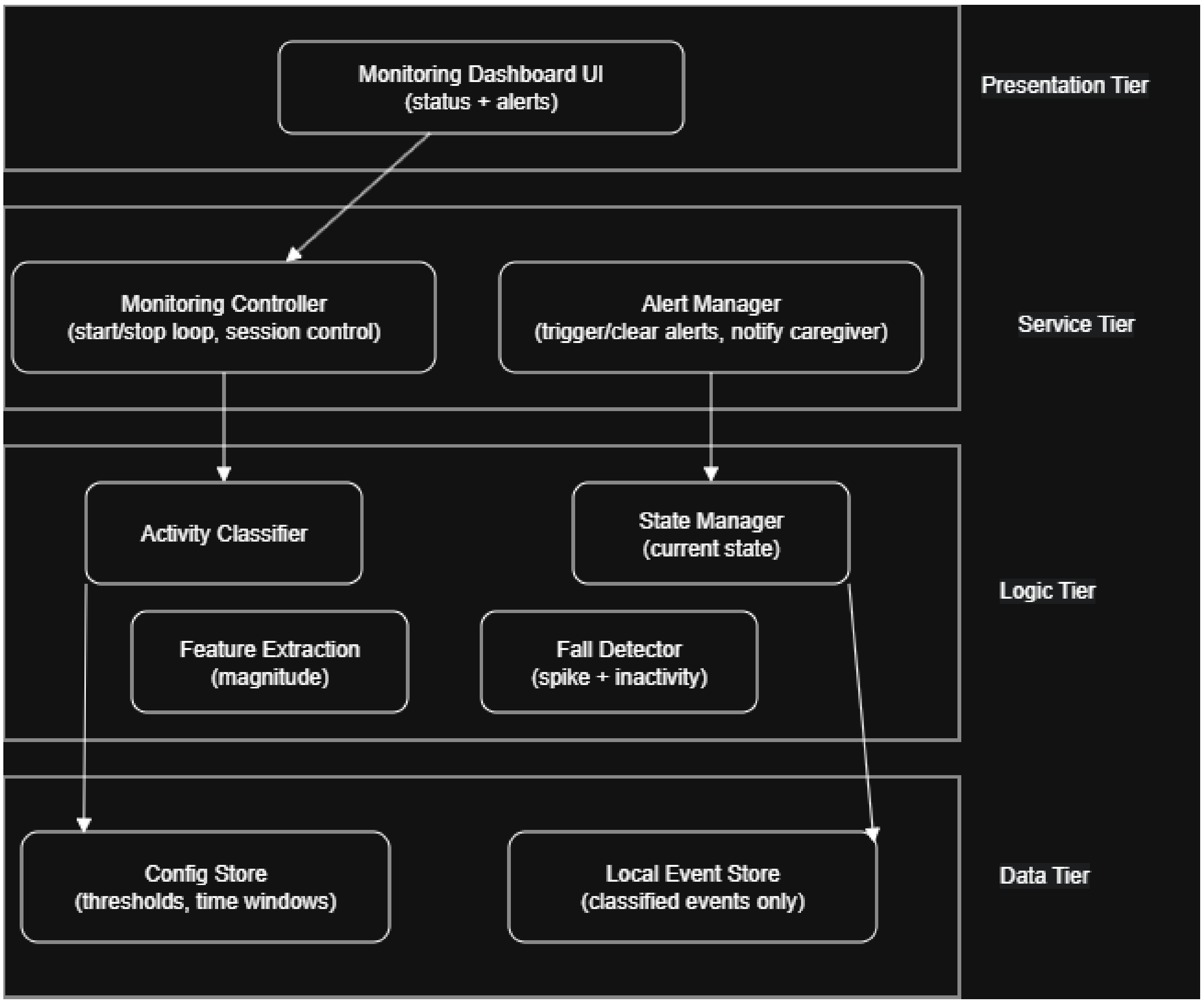
- **Accuracy**: Combining spike detection with inactivity reduces noise sensitivity
- **Performance**: Lightweight calculations ensure low processing delay
- **Privacy**: No raw data persistence
- **Flexibility**: Adjustable parameters support validation and experimentation

This algorithmic design aligns with system performance and privacy constraints defined in the SRS.

Software Architecture Pattern

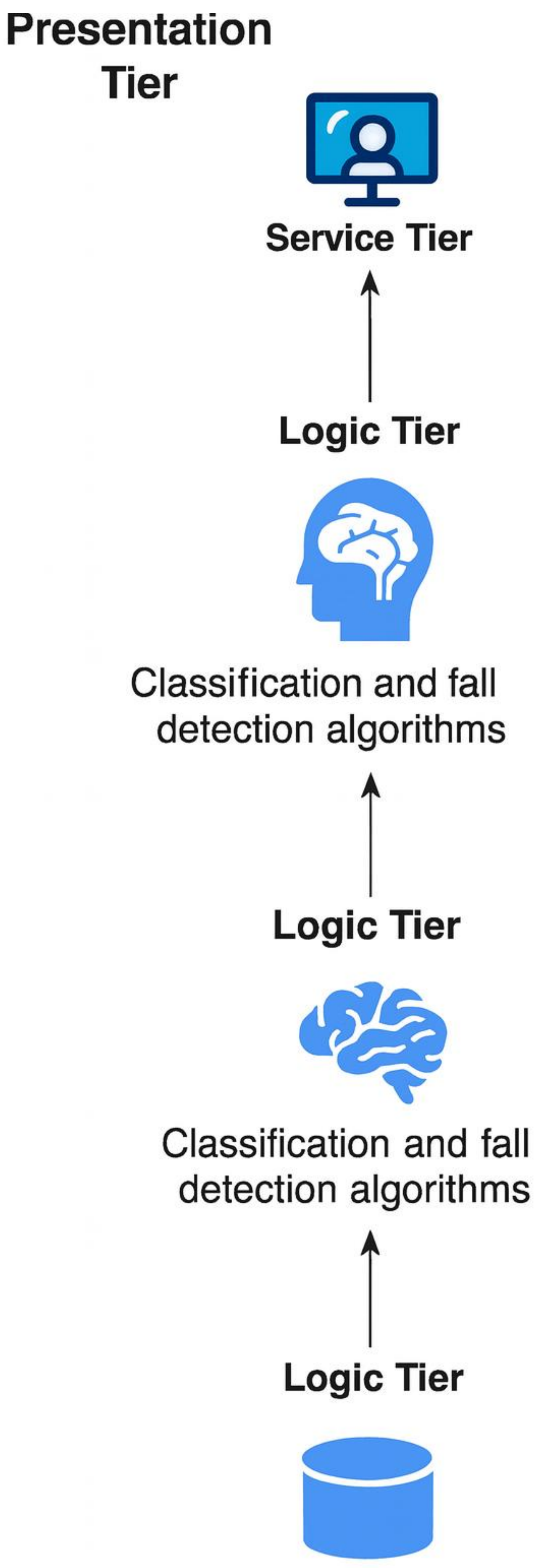
3.5.1 N-Tier Architecture

LifeBridge follows a simplified N-tier architecture:



- **Data Tier**: Local storage of classified events and configuration
- **Logic Tier**: Classification and fall detection algorithms
- **Service Tier**: Event handling and alert management
- **Presentation Tier**: Passive UI visualization

This separation improves maintainability and clarity without introducing unnecessary complexity.

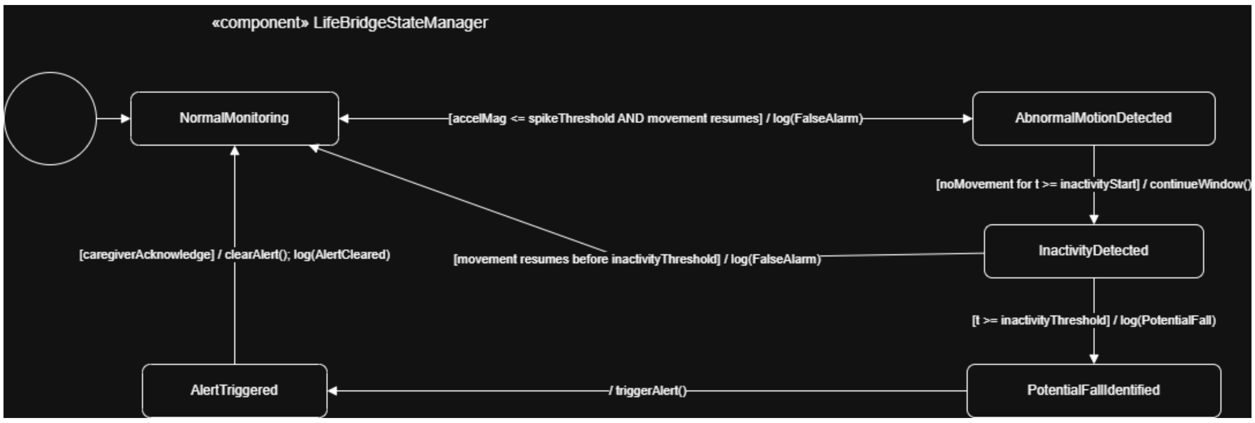

Presentation Tier
Service Tier
Logic Tier
Classification and fall detection algorithms
Logic Tier
Classification and fall detection algorithms
Logic Tier

MVC Structure

The system loosely follows an MVC-inspired structure:

- **Model**: Classified event records and configuration data
- **View**: User interface displaying system status
- **Controller**: Processing engine coordinating data flow and state transitions

The MVC approach is applied conceptually rather than strictly, serving as an organizational guideline.


«component» LifeBridgeStateManager
NormalMonitoring
[accelMag <= spikeThreshold AND movement resumes] / log(FalseAlarm)
AbnormalMotionDetected
[noMovement for t >= inactivityStart] / continueWindow()
InactivityDetected
[movement resumes before inactivityThreshold] / log(FalseAlarm)
[caregiverAcknowledge] / clearAlert(); log(AlertCleared)
[t >= inactivityThreshold] / log(PotentialFall)
PotentialFallIdentified
/ triggerAlert()
AlertTriggered

Validation

Validation and Evaluation Plan

The validation strategy for LifeBridge focuses on ensuring that the system meets its functional and non-functional requirements while operating under realistic constraints. Since the system is intended as a prototype, validation emphasizes correctness, responsiveness, and robustness rather than long-term medical accuracy.

Functional Validation

The following aspects will be validated:

- **Continuous Monitoring**
  Verify that motion data is processed continuously without interruption.
- **Activity Classification Accuracy**
  Ensure that normal motion, abnormal spikes, and inactivity states are classified correctly.
- **Fall Detection Logic**
  Confirm that fall events are detected only when both abnormal motion and subsequent inactivity conditions are met.
- **Alert Triggering**
  Validate that alerts are generated only for confirmed potential fall events.

- **Local Data Handling**
  Ensure that no raw motion data is stored persistently and that only classified events are logged.

Performance Evaluation

Performance validation focuses on responsiveness:

- Measure processing delay between data ingestion and classification
- Ensure fall detection decisions occur within defined time constraints
- Verify that UI updates reflect system state changes in near real time

Reliability and Robustness

- Evaluate system behaviour under noisy or irregular input data
- Verify that configuration changes do not disrupt ongoing monitoring
- Confirm graceful handling of false positives and non-fall scenarios

This validation plan supports iterative refinement of detection thresholds and system behaviour.

Testing Platform

LifeBridge is tested in a controlled local development environment.

Development Environment

- Local machine execution
- Desktop operating system (Windows / macOS / Linux)
- Lightweight runtime environment

Testing Methods

- **Simulation-Based Testing**
  Synthetic accelerometer data is used to represent various motion patterns, including normal activity and fall-like events.
- **Scenario Testing**
  Predefined scenarios derived from the SRS are executed to validate system behaviour.
- **Manual Verification**
  UI state transitions and alert triggers are observed and verified manually.
- **Configuration Testing**
  Threshold and sensitivity adjustments are tested to ensure correct system response.

This testing approach is sufficient for validating system logic and design decisions at the prototype level.

Project Management

Schedule / Gantt

The LifeBridge project followed a phased development approach aligned with the course timeline.

Key phases included:

1. Requirements analysis and SRS preparation
2. System design and architectural planning
3. Algorithm design and validation planning
4. Documentation and refinement

This phased approach ensured steady progress and allowed early validation of design decisions.

Team Roles – Final

The project responsibilities were distributed as follows:

Nitzan Jarus

- Requirements analysis and SRS preparation
- System design and architectural planning
- Algorithmic design and validation logic
- Documentation and final integration

Firas Shehady

- **Testing and validation**
- **development**

Bara Haj

- Testing and validation
- development