Spring 2024

Jon Pablo

Victor Wong

Olha Hodovaniuk

Kalki Tageja

<Project Name>

Software Requirements Specification

<Group Name> | Group # <#>

Executive Summary

# Background

In multiple industries, sensor data plays a critical role in monitoring and managing resource usage. Efficient resource operations can be improved through the data gathered from these sensors. However, transmitting data from distant sensors to a centralized system poses significant challenges, including issues with data transmission, storage, and security concerns.

# Description

Numerous businesses from different industries have data as a one of its pillars in decision-making. While the data differs between businesses, they all share the same challenges of gathering and managing enormous amounts of data.

DataSense’s aims to simplify the collection, organization, and management of sensor data, minimizing the time, effort and resources required to configure and manage large sensor data.

## Company Value Add

The DataSense system addresses the issue of data management from various sensors, thereby improving operational efficiency and resource management for companies.

## End-User Value Add

The DataSense system provides value to end-users with key features of real-time insights, centralized monitoring, and predictive analytics.

# Scope

## What is Included

* Centralized data management system
* Integration with various sensors (temperature, humidity, soil moisture)
* Real-time data collection and transmission
* Scalable NoSQL database for data storage
* Unified dashboard for monitoring all sensors
* Personalized alert configurations for specific thresholds
* Advanced analytics and machine learning for pattern detection and predictive analysis
* User account management and secure authentication
* API for external integrations
* Comprehensive documentation and user guides

## What is Not Included

* Support for custom sensor types beyond the initial scope
* Long-term maintenance and support for the system
* Advanced features such as real-time collaboration tools and external data source integrations

# Justification

The DataSense project is a comprehensive initiative integrating hardware, software, and networking components. It involves collecting data from sensors, transmitting it to a central server, and performing advanced analytics. This project provides practical experience in IoT, cloud computing, and data management, including using Raspberry Pi, developing with Next.js and Node.js, and managing NoSQL databases.

The complexity and interdisciplinary nature of the project, requiring skills in system integration, data security, and advanced analytics, justify its qualification for two course credits. It challenges the team to apply and expand their knowledge, offering significant learning and development opportunities.

Table of Contents

[Executive Summary 1](#_Toc114498083)

[Background 1](#_Toc114498084)

[Description 1](#_Toc114498085)

[Company Value Add 1](#_Toc114498086)

[End-User Value Add 1](#_Toc114498087)

[Scope 1](#_Toc114498088)

[What is Included 1](#_Toc114498089)

[What is Not Included 1](#_Toc114498090)

[Justification 1](#_Toc114498091)

[Section 1 – Introduction and Overview 5](#_Toc114498092)

[1.1 Document Authors 5](#_Toc114498093)

[1.2 Document Revision History 5](#_Toc114498094)

[1.3 Document Purpose 5](#_Toc114498095)

[1.4 Audience 5](#_Toc114498096)

[1.4.1 Intended Document Audience 5](#_Toc114498097)

[1.4.2 Intended Application Audience 5](#_Toc114498098)

[1.5 Group Agreement 5](#_Toc114498099)

[Team # 5](#_Toc114498100)

[Project Title 5](#_Toc114498101)

[Project Time Frame 5](#_Toc114498102)

[Team Members 6](#_Toc114498103)

[Team Leadership 6](#_Toc114498104)

[Team Functions/Roles 6](#_Toc114498105)

[Development Environment 6](#_Toc114498106)

[Team Meetings 6](#_Toc114498107)

[Team Problems 6](#_Toc114498108)

[Team Commitment 6](#_Toc114498109)

[Section 2 – Project Overview 7](#_Toc114498110)

[2.1 Project Proposal 7](#_Toc114498111)

[2.1.1 Project Background 7](#_Toc114498112)

[2.1.2 Problem Statement 7](#_Toc114498113)

[2.1.3 Product Vision 7](#_Toc114498114)

[2.2 Stakeholders and Users 7](#_Toc114498115)

[2.3 Requirements 7](#_Toc114498116)

[2.3.3 Requirements Prioritization 7](#_Toc114498117)

[2.3.1 End-User Interface 7](#_Toc114498118)

[Functional Requirements 7](#_Toc114498119)

[Non-Functional Requirements 8](#_Toc114498120)

[2.3.2 Backoffice Administrative Area 8](#_Toc114498121)

[Functional Requirements 8](#_Toc114498122)

[Non-Functional Requirements 8](#_Toc114498123)

[2.3.3 Reporting Systems 8](#_Toc114498124)

[Functional Requirements 8](#_Toc114498125)

[Non-Functional Requirements 9](#_Toc114498126)

[2.3.4 Hosting Services 9](#_Toc114498127)

[Functional Requirements 9](#_Toc114498128)

[Non-Functional Requirements 9](#_Toc114498129)

[2.3.5 Database Services 9](#_Toc114498130)

[Functional Requirements 9](#_Toc114498131)

[Non-Functional Requirements 9](#_Toc114498132)

[2.3.6 – another area if needed to copy the template for if required…. 10](#_Toc114498133)

[Functional Requirements 10](#_Toc114498134)

[Non-Functional Requirements 10](#_Toc114498135)

[2.4 Project Scope 10](#_Toc114498136)

[2.5 System Risks 11](#_Toc114498137)

[2.6 Operating Environment 11](#_Toc114498138)

[2.7 UI/UX Interface Mock-ups 11](#_Toc114498139)

[Section 3 – Process and Data Modelling 12](#_Toc114498140)

[3.1 Workflow Diagrams 12](#_Toc114498141)

[3.1.1 Application Navigation 12](#_Toc114498142)

[3.1.2 End User Workflow 12](#_Toc114498143)

[3.1.3 Adminstration Workflow 12](#_Toc114498144)

[3.2 Data Modelling and Flow 12](#_Toc114498145)

[3.2.3 Data Flow Diagrams 12](#_Toc114498146)

[3.3 Use Case Scenarios 12](#_Toc114498147)

[3.3.1 UC1 - Scenario 1 12](#_Toc114498148)

[Scenario Name 12](#_Toc114498149)

[Actors 12](#_Toc114498150)

[Stakeholders and Interests 12](#_Toc114498151)

[Description 12](#_Toc114498152)

[Assumptions, Constraints, and/or Pre-Conditions 12](#_Toc114498153)

[Trigger – Starting Point 12](#_Toc114498154)

[Relationships 12](#_Toc114498155)

[Normal Flow of Events 13](#_Toc114498156)

[Sub-Flows 13](#_Toc114498157)

[Alternate/Exceptional Flows 13](#_Toc114498158)

[3.3.2 UC2 - Scenario 2 13](#_Toc114498159)

[3.3.3 UC3 - Scenario 3 13](#_Toc114498160)

[3.3.4 UC4 - Scenario 4 13](#_Toc114498161)

[3.3.5 UC# - until there are no more scenarios 13](#_Toc114498162)

[3.4 Activity Diagrams 13](#_Toc114498163)

[3.4.1 AD1 - <name> 13](#_Toc114498164)

[3.4.2 AD2 - <name> 13](#_Toc114498165)

[3.4.3 AD3 - <name> 13](#_Toc114498166)

[3.5 Business Rules 14](#_Toc114498167)

[Section 4 – Domain Class 15](#_Toc114498168)

[Section 5 – Database 15](#_Toc114498169)

[Section 6 – Project Management 15](#_Toc114498170)

[6.1 Work Breakdown Structure 15](#_Toc114498171)

[6.2 Milestones 15](#_Toc114498172)

[6.3 Acceptance Criteria 15](#_Toc114498173)

[6.4 Implementation Schedule 15](#_Toc114498174)

[Section 7 – Client/Faculty Sign-off 15](#_Toc114498175)

Section 1 – Introduction and Overview

# 1.1 Document Authors

* Jon Pablo, Project Manager
* Victor Wong, Hardware Development Lead
* Kalki Tageja, Quality Assurance Lead
* Olha Hodovaniuk, Software Development Lead

# 1.2 Document Revision History

|  |  |  |
| --- | --- | --- |
| **WEEK** | **DATE** | **Revisions** |
| 3 | May 08, 2024 | Forming a group |
| 4 | May 15, 2024 | Prelim Proposal – Topic: Sensor, alcohol, and game website |
| 5 | May 22, 2024 | Business Case Proposal – Topic: Sensor (Name the project DataSense) |
| 6 | May 28, 2024 | Development environment additions, problem statement + stakeholder, end-user revision |
| 7 | June 11, 2024 | Product requirements added |
| 8 | June 18, 2024 | Scope and WireFrame Prototype |
| 9 | July 02, 2024 | DFD and Activity Diagrams |
| 10 | July 09, 2024 | Business Rules, Use Cases |
| 11 | July 15, 2024 | Prototype Presentation - Figma |
| 12 | July 23, 2024 | Class Diagrams |
| 13 | July 30, 2024 | Database Diagrams |
| 14 | Aug 06, 2024 | Tasks and Milestones |

# 1.3 Document Purpose

The purpose of this document is to outline an 8-month project proposal and workflows for the development and implementation of a sensor system, covering both hardware and software aspects. This document aims to:

* Define project objectives, scope, and deliverables.
* Present a structured timeline with key milestones.
* Specify hardware and software requirements.
* Describe detailed development, testing, and implementation workflows.
* Assign roles and responsibilities to team members.
* Identify and propose mitigation strategies for potential risks.
* Provide a budget and resource allocation plan.
* Outline quality assurance and testing protocols.
* Establish a communication plan for stakeholders.
* Define evaluation criteria and reporting methods.

This document serves as a comprehensive guide for all stakeholders, ensuring a clear understanding and execution methods of the project.

# 1.4 Audience

The intended audience can be broken into two categories: intended document audience and intended application audience.

## 1.4.1 Intended Document Audience

The intended document audience include project stakeholders and project developers.

## 1.4.2 Intended Application Audience

The intended application audience includes the personnel responsible for data collection, management, and analysis of various sectors of industries.

# 1.5 Group Agreement

### Team 6

### Project Title

### DataSense (Sensor Data Management System)

### Project Time Frame

May 21, 2024 - December 15, 2024

### Team Members

Jon Pablo

Victor Wong

Kalki Tageja

Olha Hodovaniuk

### Team Leadership

Jon Pablo

### Team Functions/Roles

|  |  |
| --- | --- |
| **Name** | **Role(s)** |
| Jon Pablo | Project Manager |
| Victor Wong | Hardware Development Lead |
| Olha Hodovaniuk | Software Development Lead |
| Kalki Tageja | Quality Assurance Lead |

### Development Environment

##### Project Management

* Weekly standups will be conducted every Wednesday from 11:40pm – 1:25pm (EST) via Microsoft Teams.
* GitHub Projects will be the primary tool for tracking tasks, issues and pull requests.
* Jira will be the secondary tool for tracking tasks, issues and for creating and maintaining the project Gantt chart.

#### Software

* TypeScript (v5.4.5)
* MicroPython (v1.22.2)
* Next (v14.2.3)
* Tailwind UI (v3.4)
* NextAuth (v4.24.7)

#### Hardware

* Raspberry Pi Pico W
* DHT11 Temperature, Humidity Sensor
* U7Q9 Moisture Meter Detection Module

#### Text Editor

* Visual Studio Code (v1.89.1)

#### Visual Studio Code Extensions

* Python (v2024.6.0)
* Pylance (v2024.5.1)
* Python Debugger (v2024.6.0)
* MicroPico (v3.7.6)
* VS Code ES7+ React/Redux/React-Native/JS snippets (v4.4.3)
* ESLint (v2.4.4)
* Prettier – Code Formatter (v10.4.0)
* Thunder Client (v2.24.7)
* GitHub Actions (v0.26.2)

### Team Meetings

* Monday: 3:20pm (EST), project update discussion with Clint Macdonald, in person. Recurring weekly.
* Wednesday, 11:40am – 1:25pm (EST), team standup via MS Teams. Recurring weekly.
* Wednesday, 12:40pm – 12:55pm (EST), project update discussion with Clint Macdonald via MS Teams. Recurring Weekly.

### Team Problems

A potential problem the team may face is that the project scope might be too broad, with numerous features to add, which can complicate development and risk missing deadlines. To resolve this, the team should focus on building a Minimum Viable Product (MVP) that includes only the core functionalities essential for the project's success. Prioritizing the MVP ensures the delivery of a practical and usable product within the set time limit. Once the MVP is in place and functioning, additional features can be gradually integrated to enhance the user experience without overwhelming the development process.

### Team Commitment

The undersigned members agree to collaborate effectively and diligently on the DataSense project until its completion at the end of the PRJ666 semester. We acknowledge that both as a team and as individuals, we are equally accountable for the quality and success of all deliverables.

|  |  |  |
| --- | --- | --- |
| **Name** | **Date** | **Signature** |
| Jon Pablo | 22 May 2024 | *Jon* |
| Victor Wong | 22 May 2024 | *Victor* |
| Kalki Tageja | 22 May 2024 | *Kalki* |
| Olha Hodovaniuk | 22 May 2024 | *Olha* |

Section 2 – Project Overview

# 2.1 Project Proposal

## 2.1.1 Project Background

In many industries, the effective management of sensor data is crucial for optimizing operations and resource usage. However, the current methods of handling sensor data are often fragmented and inefficient, posing challenges in data transmission, storage, and security.

The DataSense project aims to address these issues by developing centralized system for managing sensor data. This project focuses on integrating various sensors that record environmental variables into a unified platform, enabling analysis of time-series data, real-time monitoring, and personalized alerts. Sensor flexibility allows for custom monitoring solutions – making it available to multiple industries.

## 2.1.2 Problem Statement

Multiple businesses from varying industries rely on data from multiple sources for making informed decision-making to monitoring and managing business operations. However, collecting and managing multiple data sources becomes a challenge when data originates from multiple sources (e.g. sensors). The problem extends when communicating the data from the sensor to an off-location server, to storage of large data, and security.

## 2.1.3 Product Vision

The proposed solution aims to consolidate sensor data into a centralized system focused on management and flexibility where a collection of sensors can be used in combination with each other depending on the environment. Within the centralized system, a unified dashboard simplifies the monitoring of all sensors in one view. Users can configure personalized alerts for when thresholds have been exceeded. Furthermore, the system is open to advanced analytics, leveraging machine learning for pattern detection and predictive analysis.

# 2.2 Stakeholders and Users

The stakeholders in the sensor project include the development team responsible for feature creation, development, testing, maintenance and provide user guidelines. The project mentor, Clint Macdonald, who provides feedback and guidance of project direction through weekly meetings.

The end-users currently include, but not limited to, hobby gardeners and DIYers. End-users will be the primary driver for feedback and serve as the basis for feature creation. By engaging with a diverse group of users, the project aims to understand various use cases and challenges. This feedback will help refine the sensor's features and ensure it meets a wide range of needs.

# 2.3 Requirements

The following requirements tables list the requirements; sorted by each functional area of the system. Requirements are divided into a top level of classification: Functional and Non-Function Requirements.

**Functional**: Things the system must do, tasks user can complete within the system

**Non-Functional**: Properties the system must have: Operational, Performance, & Security Requirements

## 2.3.3 Requirements Prioritization

Each requirement is classified under a level of priority within the scope of the project: *<A good place to start is with is the following, feel free to modify this to match your project>*

* (MH) MUST HAVE – the requirement must be present in the current version design
* (SH) SHOULD HAVE – the requirement should be present, but if time does not permit, then could be delayed to a future version
* (NH) NICE TO HAVE – the requirement would be nice to have, but is not mandatory, and could be pushed off to a future version, if implemented at all. Most often these are quality of life kind of features.

## 2.3.1 Front End, User Interface

### Functional Requirements

|  |  |  |
| --- | --- | --- |
| **Req. #** | **Requirement** | **Priority** |
| N1 | Button to opt out of cloud storage | MH |
| N2 | User signup via email and password | MH |
| N3 | User signup via Google | NH |
| N4 | Handle existing user login with incorrect password | MH |
| N5 | Handle non-existent user login | MH |
| R64 | Export sensor data as CSV | MH |
| R65 | Export sensor data as JSON | MH |
| N6 | Create temperature sensor data interface | MH |
| N7 | Create humidity sensor data interface | MH |
| N8 | Create moisture sensor data interface | MH |
| N9 | Create user interface | MH |
| N10 | Handle 5xx responses | MH |
| N11 | Handle 4xx responses | MH |
| N12 | Handle 3xx responses | MH |
| N12 | Handle 2xx responses | MH |
| R78 | Parse HTTP headers | MH |
| R79 | Parse HTTP body | MH |
| R80 | Parse HTTP query parameters | MH |
| N14 | Parse cookies | MH |
| R82 | Assign user session | MH |
| R83 | Update user session | MH |
| R84 | Delete user session | MH |
| R85 | Read user session | MH |
| R90 | Assign Raspberry Pi to user | MH |
| R91 | Unassign Raspberry Pi from user | MH |
| R92 | Give name to Raspberry Pi | SH |
| R93 | Remove name from Raspberry Pi | SH |
| R94 | Edit name of Raspberry Pi | SH |
| N15 | Graph temperature data | MH |
| N16 | Graph humidity data | MH |
| N17 | Graph moisture data | MH |
| N18 | Change X-axis scale | SH |
| N19 | Change Y-axis scale | SH |
| R142 | Display temperature data in a table | MH |
| N20 | Display temperature data in a graph | MH |
| R143 | Display humidity data in a table | MH |
| N21 | Display humidity data in a graph | MH |
| R144 | Display moisture data in a table | MH |
| N22 | Display moisture data in a graph | MH |
| N23 | User set a threshold for a data measurement | MH |
| N24 | User removes a threshold for a data measurement | MH |
| N25 | User enables data related alerts | SH |
| N26 | User disables data related alerts | SH |
| N27 | Collapse hamburger menu on mobile view | SH |
| N28 | Open hamburger menu on mobile view | SH |
| N29 | Remove navbar on mobile view | SH |
| N30 | Quick navigation on bottom of screen on mobile view | NH |
| N31 | Navbar login is clickable from navbar when no there is no user session | MH |
| N32 | Navbar registration is clickable from navbar when no there is no user session | MH |
| N33 | Forget password is clickable on login page | MH |
| N34 | Reset password is clickable on login page | MH |
| N35 | Home is clickable on navbar | MH |
| N36 | Sensor details page link clickable on navbar | MH |
| N37 | Button to retrieve live data from sensor | MH |
| N38 | Instantiate a web socket between user and Raspberry Pi | MH |
| N39 | Link to view historical data | MH |
| N40 | Link to manage user alerts and notifications page | SH |
| N41 | Link to user’s profile page | MH |
| N42 | Link to about page on footer | NH |
| N43 | Link to content page on footer | NH |
| N44 | Link to help page on footer | NH |
| N45 | Link to FAQ on footer | NH |
| N46 | Link to social media on footer | NH |
| N47 | Button for user to delete account | SH |
| N48 | Confirmation prompt to delete account | SH |
| N49 | Pop-up message boxes for system responses | MH |
| N50 | Animate graph data | NH |
| N51 | Animate loading screens | NH |

### Non-Functional Requirements

|  |  |  |
| --- | --- | --- |
| **Req. #** | **Requirement** | **Priority** |
| R135 | DateTime data always on X-axis | SH |
| R136 | Sensor data always on Y-axis | SH |
| N52 | Display client-side errors to user | MH |
| N53 | Display success prompts to user | MH |
| N54 | Display sensor data in table and graph formats | MH |
| N55 | Page layout differs depends on screen size | MH |
| N56 | UI adheres to accessibility standards | MH |
| N57 | Process for users to provide feedback | SH |
| N58 | Process for users to report bugs | SH |
| N59 | User has the option to opt out of data being stored in the cloud | MH |

## 2.3.2 Back End, Server

### Functional Requirements

|  |  |  |
| --- | --- | --- |
| **Req. #** | **Requirement** | **Priority** |
| N60 | User login via Google | NC |
| N61 | User registration via Google | NC |
| N62 | User login via email and password | MH |
| N63 | User registration via email and password | MH |
| N64 | Handle wrong password error | MH |
| N65 | Handle non-existent username error | MH |
| N66 | Users should be able to reset their password | SH |
| R120 | Assign user session | MH |
| R121 | Update user session | MH |
| R122 | Delete user session | MH |
| R123 | Read user session | MH |
| N67 | Authentication protected dashboard endpoint | MH |
| N68 | Authentication protected sensor detail endpoint | SH |
| N69 | Authentication protected data livestream endpoint | SH |
| N70 | Authentication protected data history endpoint | SH |
| N71 | Authentication protected alerts and notifications endpoint | SH |
| N72 | Instantiate a web-socket between server and IoT device | SH |
| N73 | Authentication protected account settings page | SH |
| N74 | Authenticated protected GET endpoint for temperature data | MH |
| N75 | Authenticated protected GET endpoint for humidity data | MH |
| N76 | Authenticated protected GET endpoint for moisture data | MH |
| R137 | Change unit of measure in X-axis | SH |
| R138 | Change unit of measure in Y-axis | SH |
| N77 | Empty response if no data available | SH |
| N78 | Error message if cannot return data | MH |
| R144 | Set data threshold | MH |
| R145 | Remove data threshold | MH |
| R146 | Turn on alert for data threshold | SH |
| R147 | Turn off alert for data threshold | SH |
| R112 | Handle error code 5xx | MH |
| R113 | Handle error code 4xx | MH |
| R114 | Handle error code 3xx | MH |
| R37 | Receive temperature data from Raspberry Pi | MH |
| R38 | Receive humidity data from Raspberry Pi | MH |
| R39 | Receive moisture data from Raspberry Pi | MH |
| R40 | Parse temperature data | MH |
| R41 | Parse humidity data | MH |
| R42 | Parse moisture data | MH |
| R46 | Handle temperature sensor data corruption | MH |
| R47 | Handle humidity sensor data corruption | MH |
| R48 | Handle moisture sensor data corruption | MH |
| R52 | Configure Google auth callback | NC |
| R53 | Configure email auth callback | MH |
| R54 | Persist Google user | NC |
| R55 | Persist Email user | MH |
| R56 | Delete Google user | NC |
| R57 | Delete Email user | MH |
| R58 | Update username | SH |
| R59 | Add profile picture | SH |
| R60 | Edit profile picture | SH |
| R61 | Remove profile picture | SH |
| R64 | Export sensor data as CSV | MH |
| R65 | Export sensor data as JSON | MH |
| R67 | Interface temperature sensor data | MH |
| R69 | Interface humidity sensor data | MH |
| R71 | Interface moisture sensor data | MH |
| R73 | Interface user data | MH |
| R78 | Parse HTTP headers | MH |
| R79 | Parse HTTP body | MH |
| R80 | Parse HTTP query parameters | MH |
| R81 | Parse request cookies | MH |
| R86 | Assign JWT | MH |
| R87 | Update JWT | MH |
| R88 | Delete JWT | MH |
| R89 | Read JWT | MH |
| R90 | Assign Raspberry Pi to user | MH |
| R91 | Unassign Raspberry Pi from user | SH |
| R92 | Give name to Raspberry Pi | SH |
| R93 | Remove name from Raspberry Pi | SH |
| R94 | Edit name of Raspberry Pi | SH |
| R95 | Store all sensor data if user opts out of cloud storage | MH |
| R96 | Notify user when SD card reaches X capacity | SH |
| N79 | Hashing identifiable information | MH |
| N80 | Authenticated protected PUT route to delete user account | MH |
| N81 | Pagination of large data | SH |
| N82 | Authenticated filtering of GET data requests | SH |
| N83 | Data compression of large data | SH |
| N84 | Session endpoint tracking | SH |

### Non-Functional Requirements

|  |  |  |
| --- | --- | --- |
| **Req. #** | **Requirement** | **Priority** |
| N85 | User can opt out of cloud-based storage | MH |
| N86 | User can register using multiple services | SH |
| N87 | Push notifications/alerts via email | SH |
| N88 | Large data can be easily manipulated when viewing | SH |
| N89 | Data can be downloaded in different formats | MH |

## 2.3.3 Cloud Infrastructure

### Functional Requirements

|  |  |  |
| --- | --- | --- |
| **Req. #** | **Requirement** | **Priority** |
| N90 | Attach logging service to server instance | MH |
| N91 | Attach server monitoring to server instance | MH |
| N92 | Configure push notifications for critical server actions | SH |
| N93 | Configure API performance monitoring | SH |

### Non-Functional Requirements

|  |  |  |
| --- | --- | --- |
| **Req. #** | **Requirement** | **Priority** |
| N94 | Ability to monitor the health of the main server | MH |
| N95 | Send push notifications when server in critical health | SH |

## 2.3.4 Physical Hardware

### Functional Requirements

|  |  |  |
| --- | --- | --- |
| **Req. #** | **Requirement** | **Priority** |
| R24 | Measure DHT11 dimensions | SH |
| R25 | Measure moisture sensor dimensions | SH |
| R26 | Measure Raspberry Pi dimensions | SH |
| R27 | Add case mounting points | NC |
| R28 | Add case ventilation | NC |
| N96 | Add case connector openings | NC |
| N97 | 3D print case prototype | NC |

### Non-Functional Requirements

|  |  |  |
| --- | --- | --- |
| **Req. #** | **Requirement** | **Priority** |
| N98 | IoT device is housed, protected from environmental elements | SH |
| N99 | Sensors can be removed | SH |
| N100 | Sensors can be added to Raspberry Pi | SH |

## 2.3.4 Embedded Software

### Functional Requirements

|  |  |  |
| --- | --- | --- |
| **Req. #** | **Requirement** | **Priority** |
| R1 | Detect temperature from sensor | SH |
| R2 | Detect humidity from sensor | SH |
| R3 | Detech moisture from sensor | SH |
| R4 | Calibrate temperature sensor | SH |
| R5 | Calibrate humidity sensor | SH |
| R6 | Calibrate moisture sensor | SH |
| R7 | Read temperature from sensor | SH |
| R8 | Read humidity from sensor | SH |
| R9 | Read moisture from sensor | SH |
| N101 | Write temperature data locally to SD card | SH |
| N102 | Write humidity data locally to SD card | SH |
| N103 | Write moisture data locally to SD card | SH |
| R17 | Handle temperature sensor malfunction | SH |
| R18 | Handle humidity sensor malfunction | SH |
| R19 | Handle moisture sensor malfunction | SH |
| R20 | Handle WiFi absence | SH |
| N104 | Send local data to main server | SH |
| R22 | Connect Raspberry Pi to WiFi | SH |
| N105 | Disable sensor | SH |
| N106 | Enable sensor | SH |

### Non-Functional Requirements

|  |  |  |
| --- | --- | --- |
| **Req. #** | **Requirement** | **Priority** |
| N107 | Have “emergency” procedure when unexpectedly shuts down | SH |
| N108 | Have setup process when power is restored | SH |

## 2.3.5 Continuous Integration, Continuous Deployment

### Functional Requirements

|  |  |  |
| --- | --- | --- |
| **Req. #** | **Requirement** | **Priority** |
| N109 | Create testing branch for requirement testing | SH |
| N110 | Create linting tests, enforcing coding standards | SH |
| N111 | Include layer libraries in testing suite | SH |
| N112 | Configure GitHub labs to automatically test upon pull request | SH |
| N113 | Generate a coverage report after merging a pull request | SH |

### Non-Functional Requirements

|  |  |  |
| --- | --- | --- |
| **Req. #** | **Requirement** | **Priority** |
| N114 | Implement a system to prevent merge conflicts | SH |
| N115 | Have a deployment system to minimize bugs on production branch | SH |

## 2.3.6 User Documentation

### Functional Requirements

|  |  |  |
| --- | --- | --- |
| **Req. #** | **Requirement** | **Priority** |
| N116 | Provide user with quick-start guide | SH |
| N117 | Provide tutorials on how to use the site | SH |
| N118 | Provide hardware specifications for troubleshooting | SH |

### Non-Functional Requirements

|  |  |  |
| --- | --- | --- |
| **Req. #** | **Requirement** | **Priority** |
| N119 | Provide users with resources when troubleshooting the IoT device | SH |

## 2.3.7 Machine Learning

### Functional Requirements

|  |  |  |
| --- | --- | --- |
| **Req. #** | **Requirement** | **Priority** |
| N120 | Integrate forecasting model into server | SH |
| N121 | Collect temperature training data | SH |
| N122 | Collect humidity training data | SH |
| N123 | Collect moisture training data | SH |
| N124 | Train forecasting model on training data | SH |
| N125 | Validate model with hold-off training set | SH |

## 2.3.7 Database Services

### Functional Requirements

|  |  |  |
| --- | --- | --- |
| **Req. #** | **Requirement** | **Priority** |
| R66 | Model temperature sensor data | MH |
| R68 | Model humidity sensor data | MH |
| R70 | Model moisture sensor data | MH |
| R72 | Model user data | MH |
| R43 | Persist temperature data | SH |
| R44 | Persist humidity data | SH |
| R45 | Persist moisture data | SH |
| R54 | Persist Google user | NC |
| R55 | Persist Email user | SH |
| R56 | Delete Google user | NC |
| R57 | Delete Email user | SH |
| R58 | Update username | SH |
| R59 | Add profile picture | SH |
| R60 | Edit profile picture | SH |
| R61 | Remove profile picture | SH |
| R90 | Assign Raspberry Pi to user | MH |
| R91 | Unassign Raspberry Pi from user | MH |
| R92 | Give name to Raspberry Pi | MH |
| R93 | Remove name from Raspberry Pi | MH |
| R94 | Edit name of Raspberry Pi | MH |
| R229 | Implement data archival for long-term storage | SH |
| R238 | Implement automated backups for data protection | SH |

### Non-Functional Requirements

|  |  |  |
| --- | --- | --- |
| **Req. #** | **Requirement** | **Priority** |
| N126 | Have backup system in case of a service outage | SH |
| N127 | Archival system to store old data | SH |

# 2.4 Project Scope

The DataSense project is designed to streamline how industries manage sensor data. By bringing different sensors into one unified platform, it allows for real-time monitoring, detailed analysis, and custom alerts. This centralized system simplifies data handling, enhances security, and adapts to various industry needs. Ultimately, DataSense aims to make data management more efficient and insightful, helping businesses optimize their operations.

The current version of the application will include the following features:

* **Centralized Sensor Management**: Integration of various sensors for temperature, humidity, and moisture into a unified platform.
* **User Authentication and Session Management:** account signup and login, session handling, and user authentication for dashboard access.
* **Data Export and Visualization:** Export sensor data as CSV and JSON, and display data in tables and graphs with monitoring capabilities.
* **Error Handling and Notifications:** Comprehensive handling of HTTP responses and error messages.

The following features will not be included in the current version, but may be considered in a future version of the software:

* **Advanced Analytics and Forecasting:** Integration of machine learning models for predictive analysis and pattern detection using collected sensor data.
* **Additional Authentication Methods:** Google-based user signup and login, and other third-party authentication services.
* **Dynamic Data Handling:** Features like changing axis scales on graphs, advanced data manipulation.

**Conclusion:**

The project scope ensures a shared understanding of the deliverables, timelines, and constraints among all stakeholders. It forms the foundation for project planning, execution, and monitoring, ensuring the project is completed successfully and meets all defined objectives.

# 2.5 System Risks

All system design and implementation processes have associated risks. Identifying and managing these risks is crucial for ensuring the successful delivery of the project. Below is a detailed analysis of the potential risks and the responses to mitigate them:

|  |  |  |  |
| --- | --- | --- | --- |
| Risk | Impact | Likelihood | Response |
| Inaccurate sensor Data Collection | inaccurate sensor data could lead to incorrect analytics and decision-making. | Medium | Implement data validation and verification processes. Regularly calibrate and maintain sensors to ensure accuracy. |
| Integration Challenges with Various Sensors | Delays in project timelines if sensors do not integrate seamlessly | Medium | Establish clear integration protocols and conduct initial integration tests early in the project. Work closely with sensor vendors to resolve any compatibility issues. |
| Real-Time Data Transmission Latency | Inaccurate real-time data reporting, which could affect decision-making | Medium | Optimize data transmission protocols and use high-speed communication channels. Implement redundant data transmission paths to ensure reliability. |
| Scalability of NoSQL Database | Performance issues as data volume grows, potentially leading to system slowdowns | Medium | Conduct scalability testing and implement database optimization techniques. Use a distributed database architecture to manage large data volumes efficiently. |
| User Authentication and Data Security | Risk of data breaches and unauthorized access | High | Implement multi-factor authentication and regular security audits. Use encryption for data storage and transmission. Conduct regular penetration testing to identify and address vulnerabilities. |
| Advanced Analytics and Machine Learning Accuracy | Inaccurate predictions and analysis could lead to poor decision-making | Medium | Use well-established machine learning algorithms and continuously refine models based on new data. Validate model outputs with domain experts to ensure accuracy. |
| API for External Integrations | Potential security vulnerabilities and data breaches through API endpoints | Medium | Secure API endpoints with authentication and encryption. Implement rate limiting and continuous monitoring for unusual activity. Conduct regular security assessments of API integrations. |
| User Account Management | Mismanagement could lead to unauthorized access or data breaches | Medium | Implement role-based access controls and regular audits of user permissions. Provide training for administrators on best practices in user account management. |
| Documentation and User Guide Completeness | Inadequate documentation could lead to user confusion and increased support requests | Low | Allocate sufficient time for comprehensive documentation. Involve end-users in reviewing documentation to ensure clarity and completeness. |
| Budget Overruns | Project costs exceeding the initial budget, impacting financial viability | Medium | Implement strict budget monitoring and control mechanisms. Have contingency plans and buffers in place for unexpected expenses. |
| Timeline Delays | Project not delivered on time, impacting stakeholder satisfaction and project outcomes | Medium | Develop a detailed project plan with realistic timelines. Monitor progress closely and implement corrective actions promptly when deviations occur. |
| Long-Term Maintenance and Support Needs | Potential issues with system upkeep and evolution beyond initial deployment | Low | Plan for a separate support agreement post-deployment. Establish a maintenance plan and allocate resources for ongoing support and system updates. |
| Environmental Factors Affecting Sensors | Extreme weather conditions could impact sensor functionality and data accuracy. | Low | Use sensors rated for extreme conditions. Implement protective measures and redundant sensors to ensure continuous operation. |
| Database Breaches | Unauthorized access to sensitive data can lead to data loss and legal consequences. | High | Use strong encryption for data at rest and in transit. Implement access controls and regular security audits. Conduct penetration testing and vulnerability assessments regularly. |
| Data Loss | Loss of critical data can disrupt business operations and result in significant financial loss. | Medium | Implement regular data backup and recovery procedures. Use redundant storage solutions and test backups periodically to ensure data can be restored. |
| Unauthorized Data Modification | Unauthorized changes to the database can corrupt data and disrupt operations. | Medium | Implement role-based access controls and audit trails. Use version control for database changes and monitor for unauthorized modifications. |

Proactively identifying and addressing potential risks ensures the project is delivered successfully on time, and within budget. Regular risk assessments and updates to the risk management plan will be essential throughout the project lifecycle.

# 2.6 Operating Environment

The DataSense project is designed to work seamlessly across different aspects that cater to users, administrators, hosting infrastructure, and more. Here’s how each component contributes to its functionality:

**Users**

* **User Interface:** Accessible via major web browsers (Chrome, Firefox, Safari, Edge) on desktops, tablets, and smartphones.
* **User Roles:** Different user roles with specific access levels (e.g., regular users, administrators) to control data access and system functionalities.
* **Authentication:** Secure authentication mechanisms using NextAuth, supporting email/password and potentially OAuth integrations in the future.

**Administrators**

* **Admin Dashboard:** A comprehensive admin dashboard for managing users, monitoring sensor data, and system maintenance.
* **User Management:** Tools for administrators to add, remove, or update user roles and permissions.

**Hosting and Infrastructure**

* **APIs:** Utilizes custom and third-party APIs for data management, authentication, and additional functionalities.

**Database**

* **Data Security:** Encryption of data at rest and in transit, with secure access controls.
* **Database Management System**: Robust database management with support for NoSQL databases, ensuring efficient data storage and retrieval.

**Development and Maintenance Tools**

* **Development Tools:** Utilizes Visual Studio Code, GitHub for version control, and Jira for project management.
* **Documentation:** Comprehensive user and developer documentation, including quick start guides and tutorials.
* **Continuous Integration/Continuous Deployment (CI/CD):** Automated testing and deployment pipelines to ensure code quality and rapid delivery.

**Hardware Components**

* 8GB SD Card
* Breadboard
* SD Card Module
* Raspberry Pi Pico W
* Dupont Wire Bundle
* DHT11 Temperature and Humidity Sensor
* Soil Moisture Meter Module

# 2.7 UI/UX Interface Mock-ups

The following screenshots are an initial mock-up of the screens to be provided within the application. They are initially creating using wireframes and a content review, and later created with more defined graphics, look and feel, in addition to other user experience considerations.

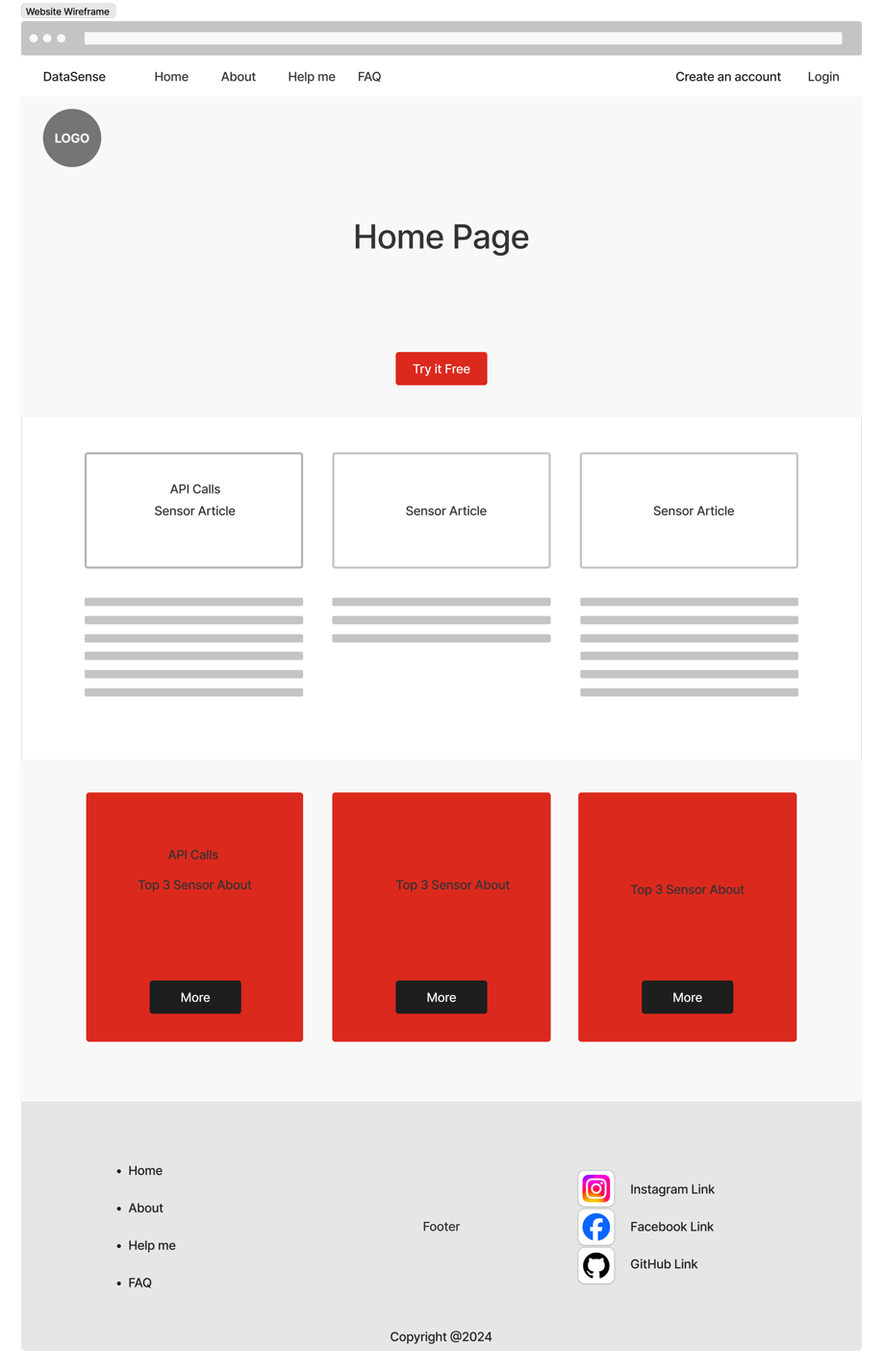
The team is using Figma to construct the content views and workflows.

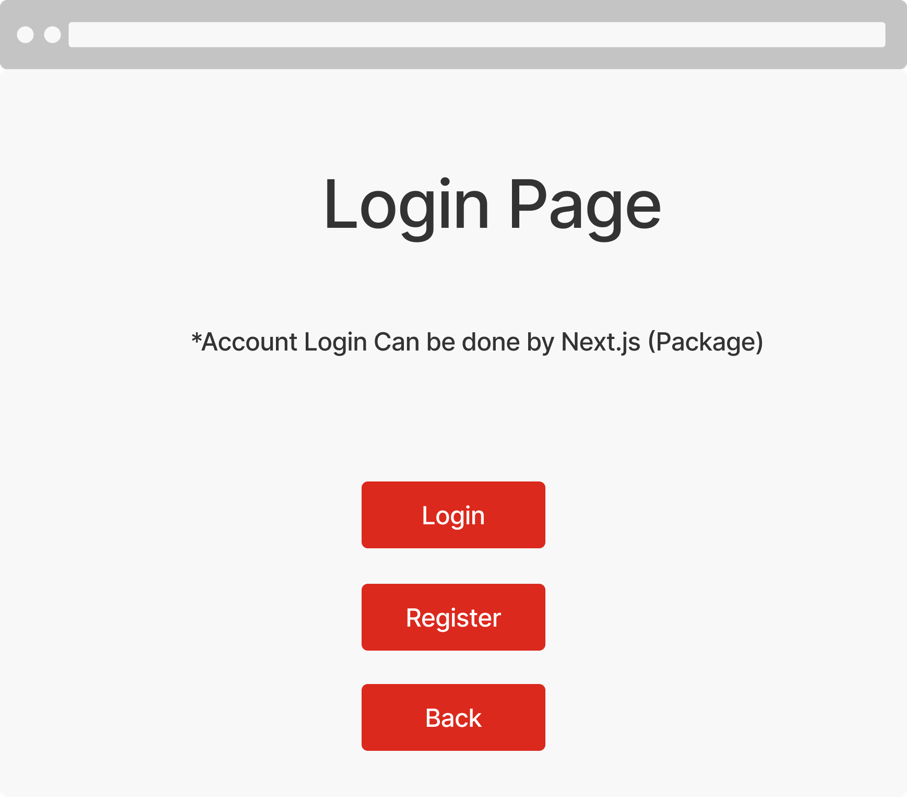
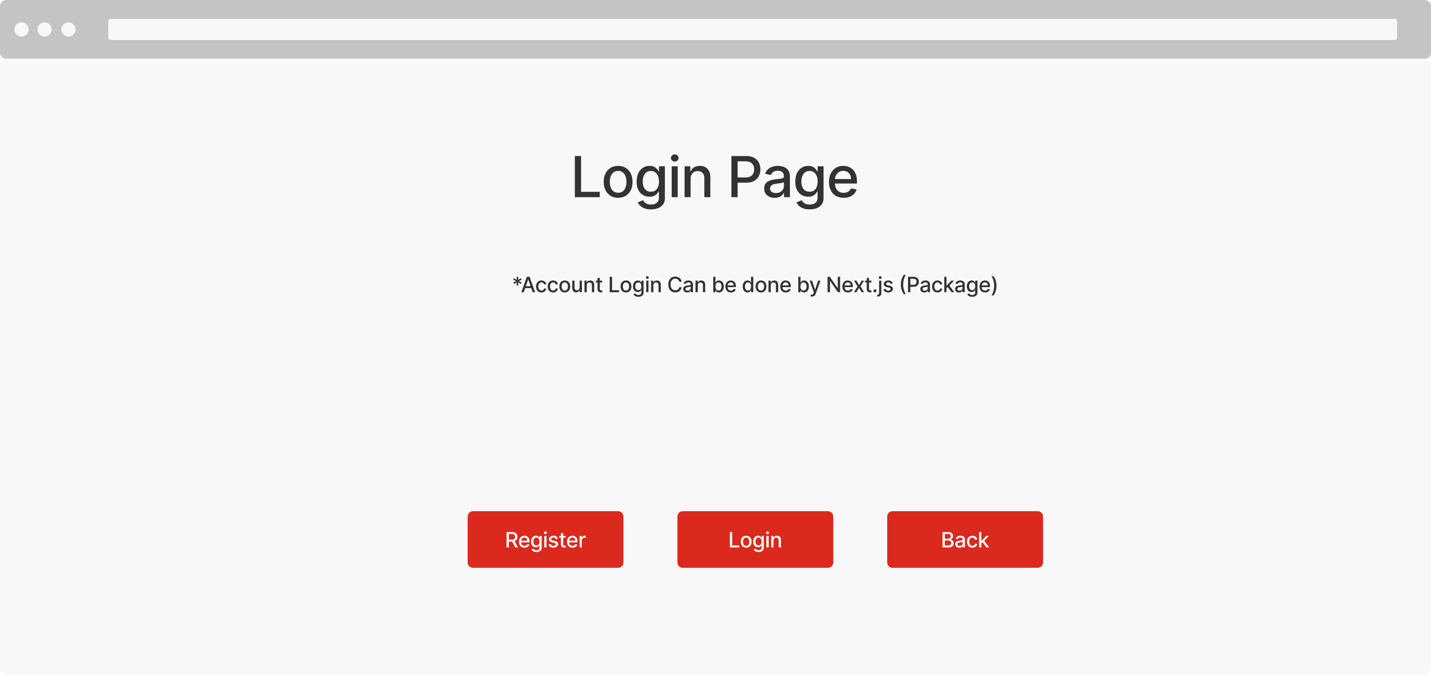
Link:  
<https://www.figma.com/board/hJOFehipJJC7eZLJJQweGn/PRJ566?node-id=0-1&t=QfCt5zUIHcxEFqRM-0>

Based on AODA compliance, the team has chosen some primary and secondary colors for their website. The ideas might change later based on client requirements or design change orders.

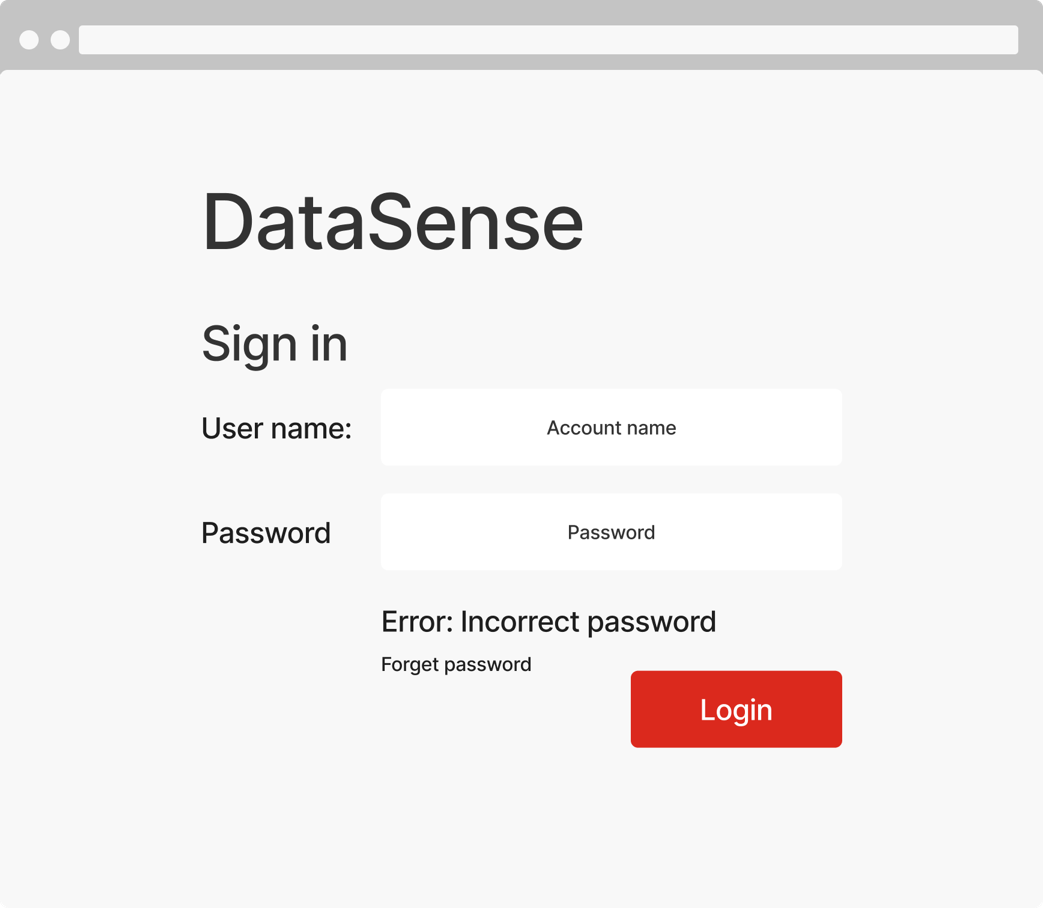
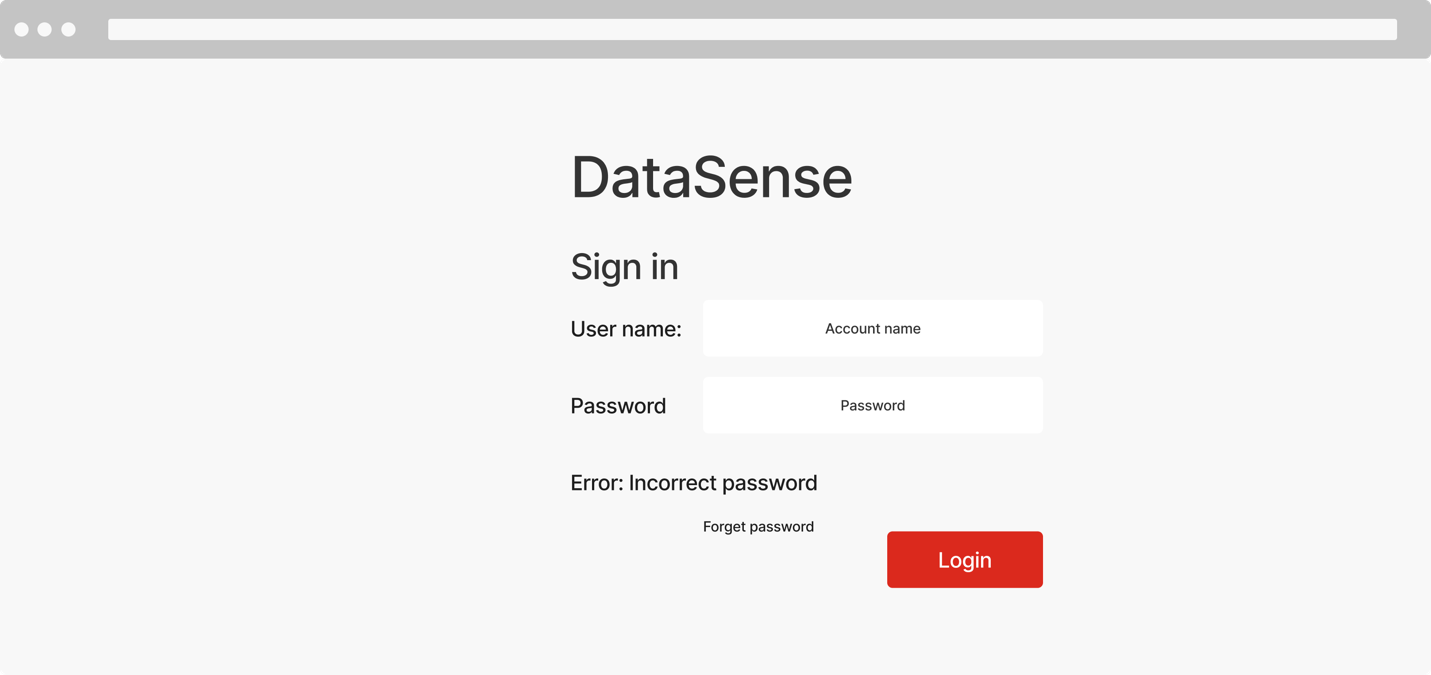
|  |  |
| --- | --- |
| Color | Code |
| Red | #DB291D |
| Grey | #F8F8F8 |
| Dark Grey | #F0F0F0 |
| White | #FFFFFF |

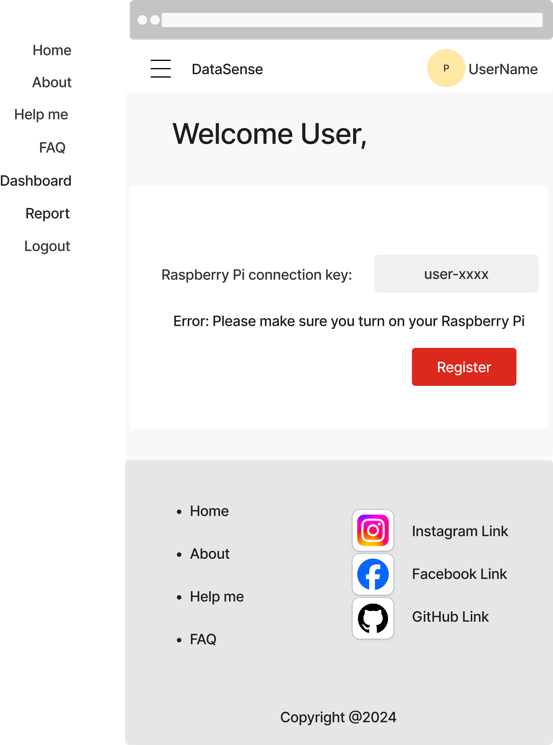
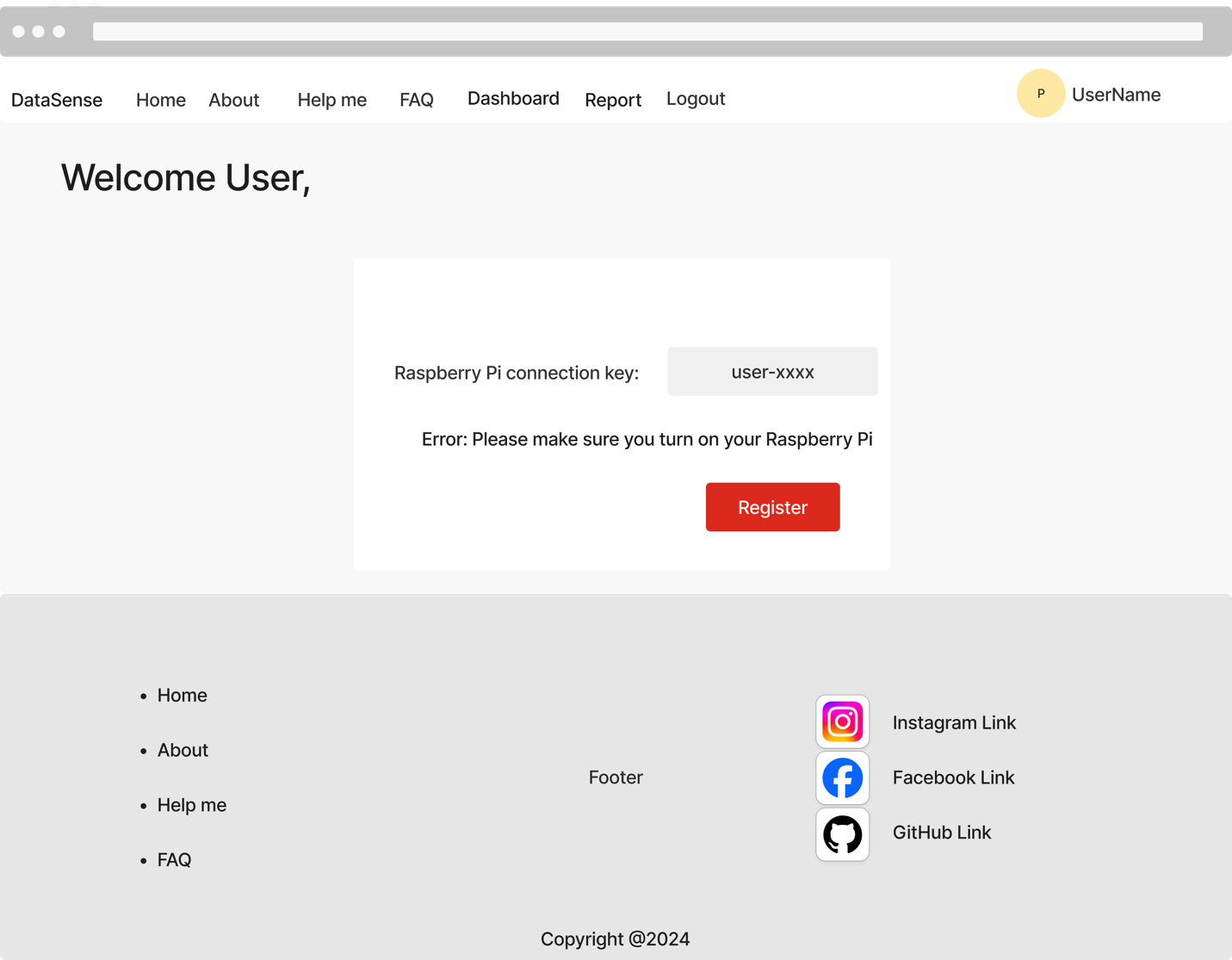
Home Page:

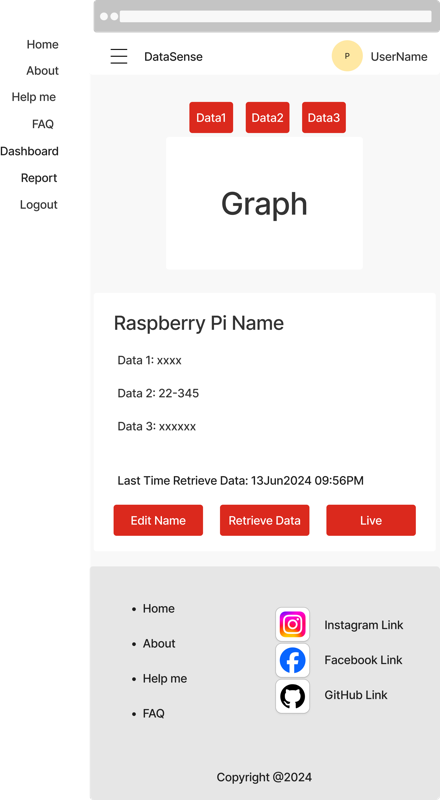
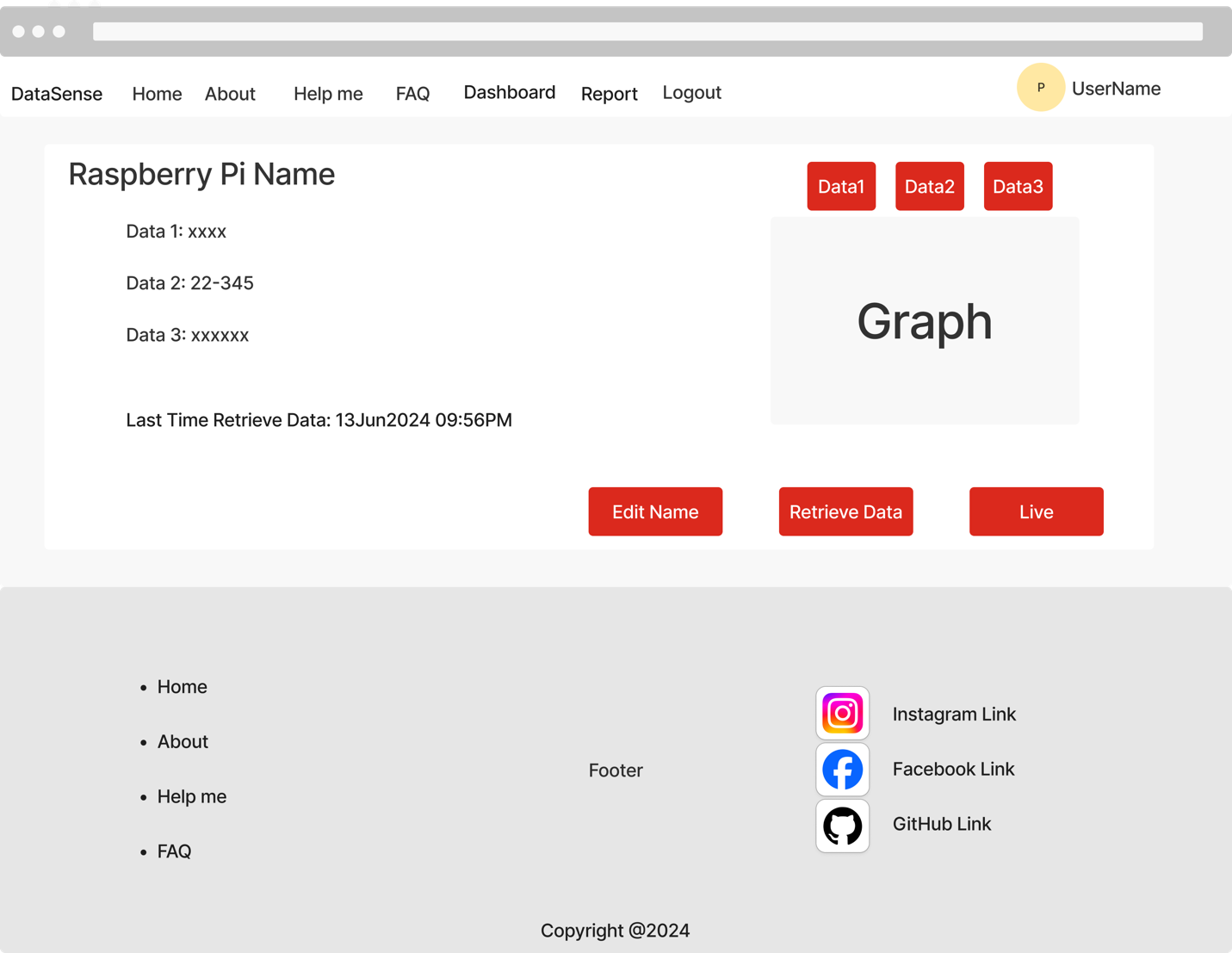


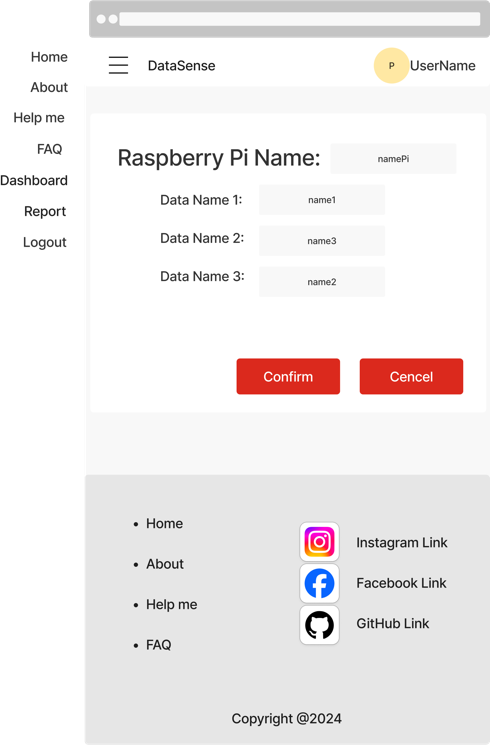
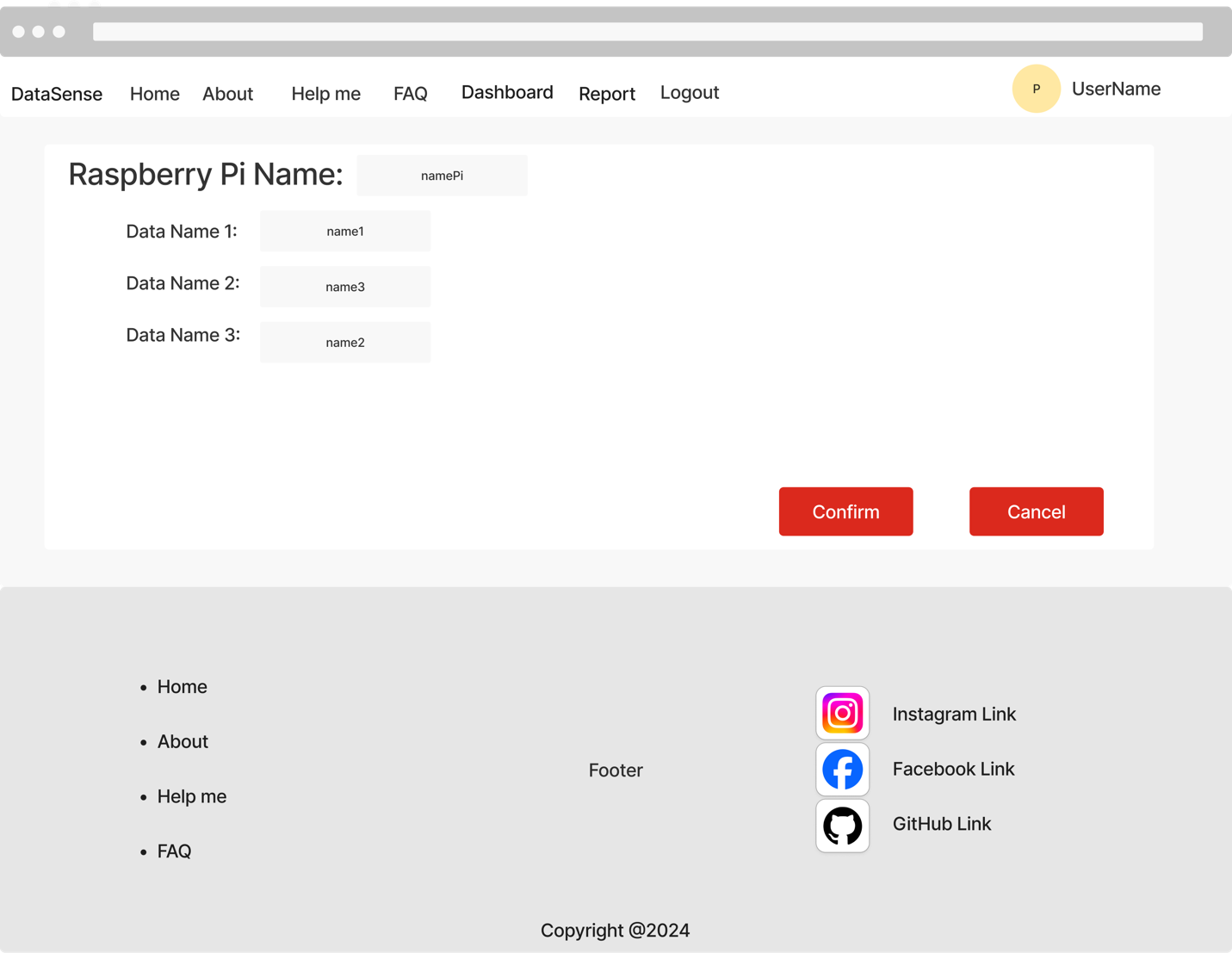
Login menu Page:  


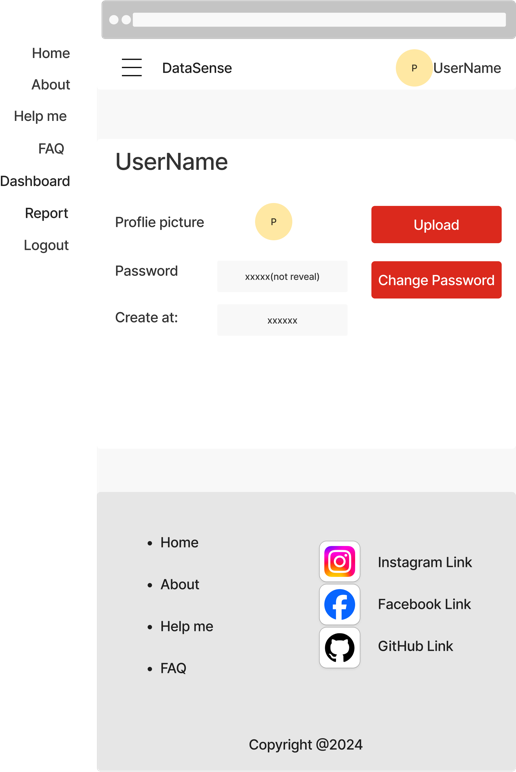
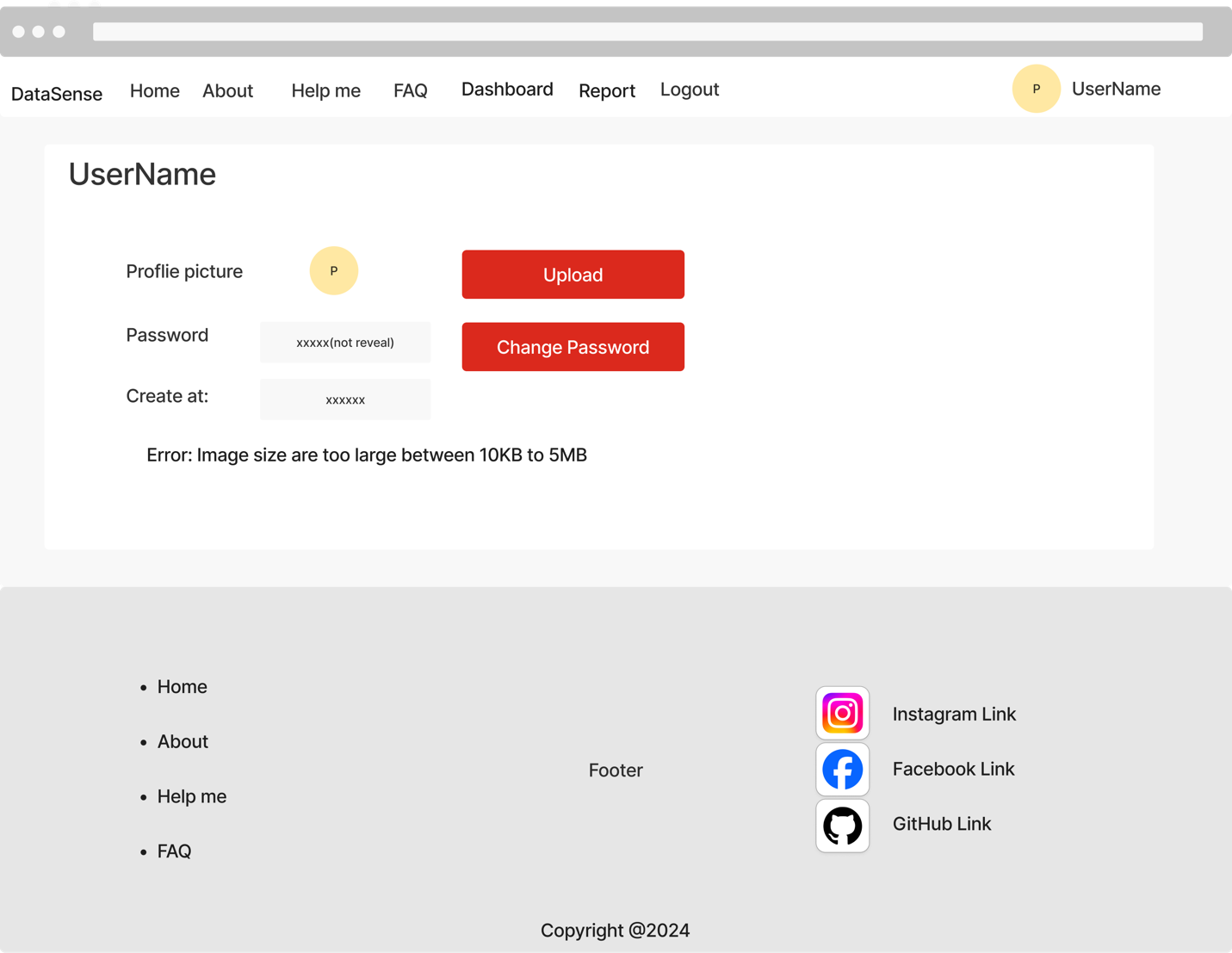
Login Page:



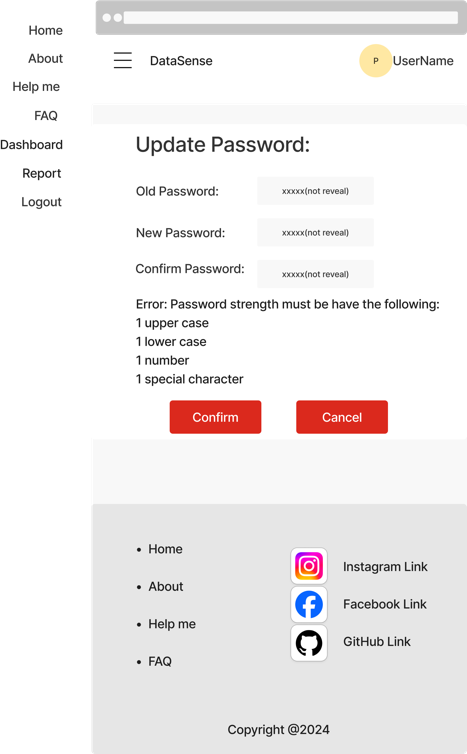
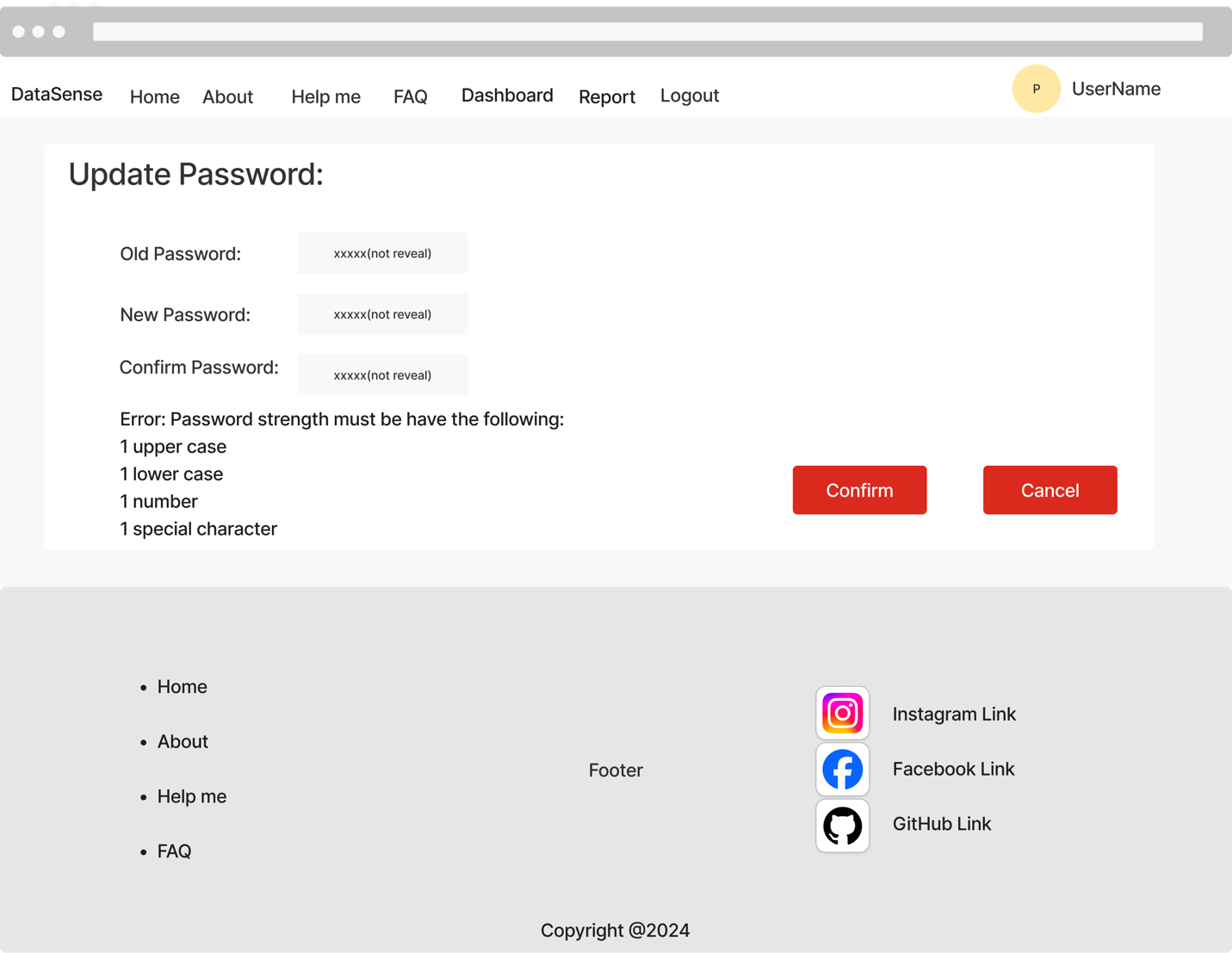
User Dashboard Page (without Raspberry Pi)

User Dashboard Page (with Raspberry Pi):  


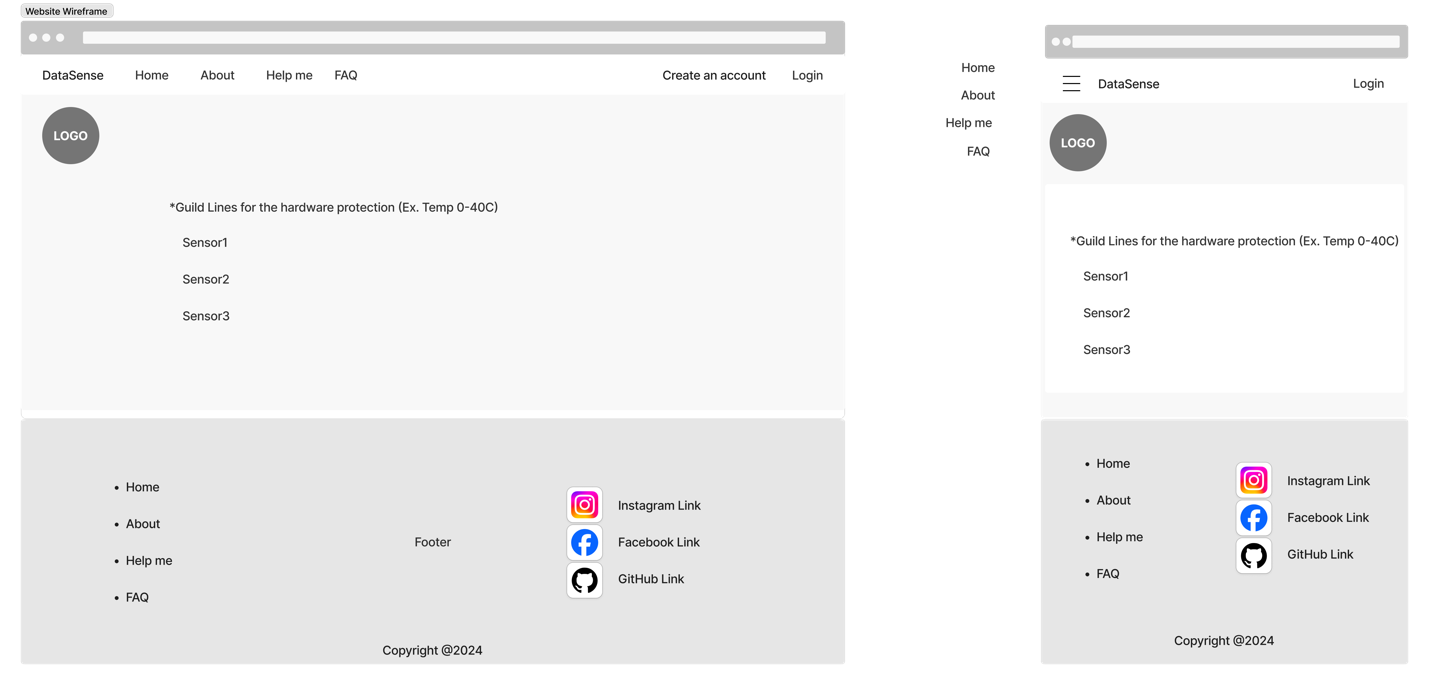
User Edit Name Page (with Raspberry Pi):  


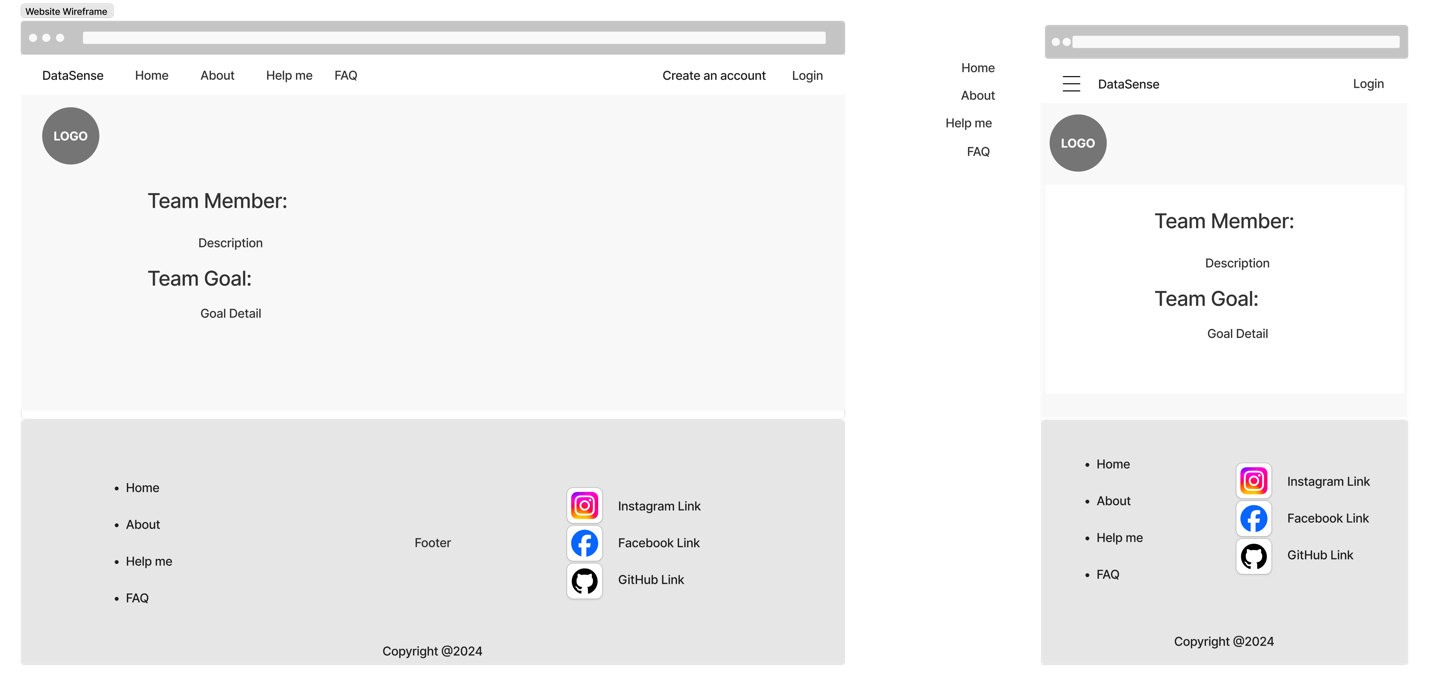
User Account Page (with/without Raspberry Pi):  


User Change Password Page (with/without Raspberry Pi):

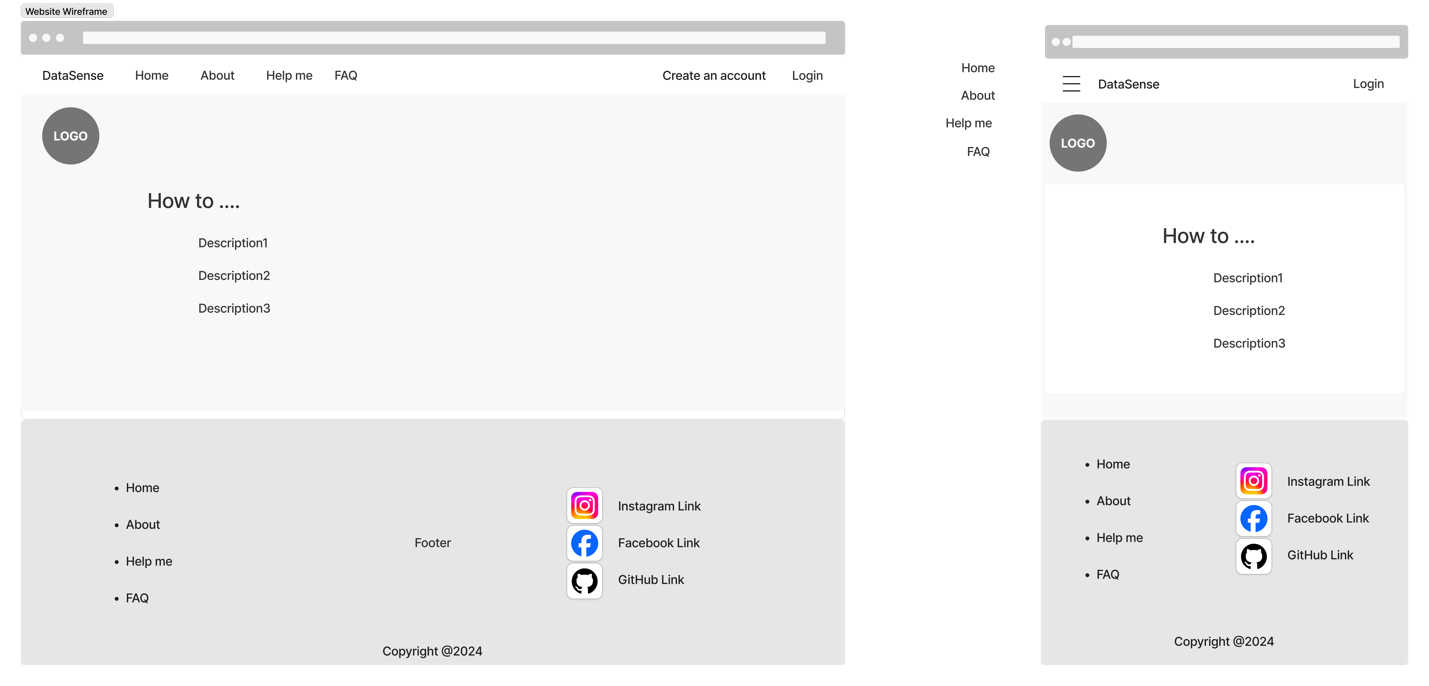


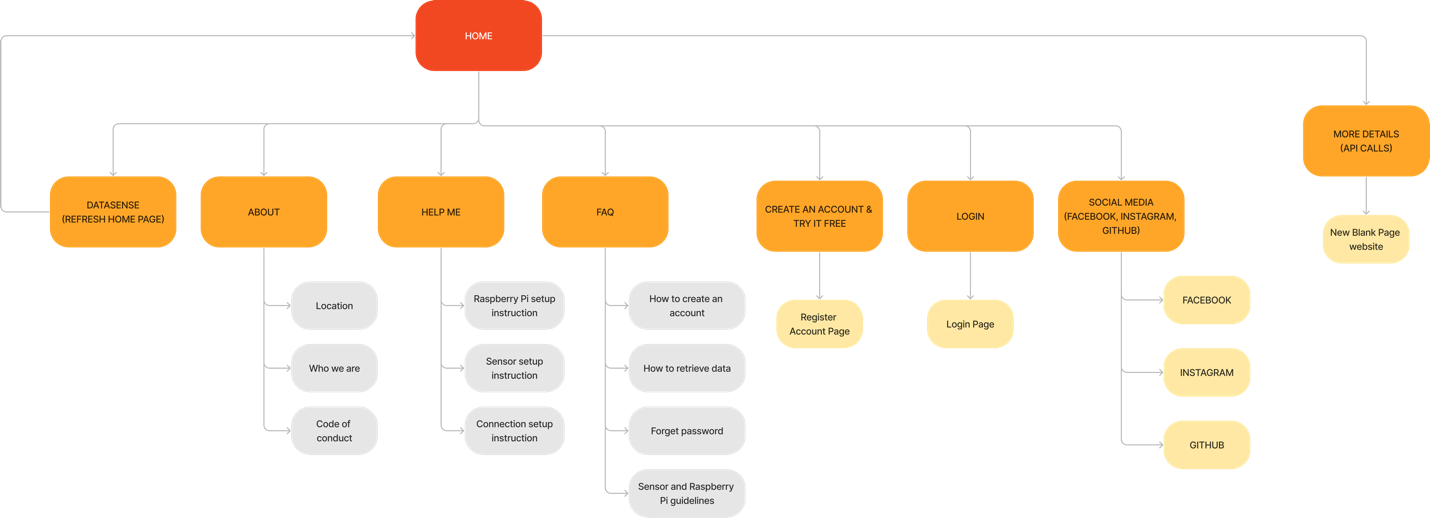
FAQ Page:

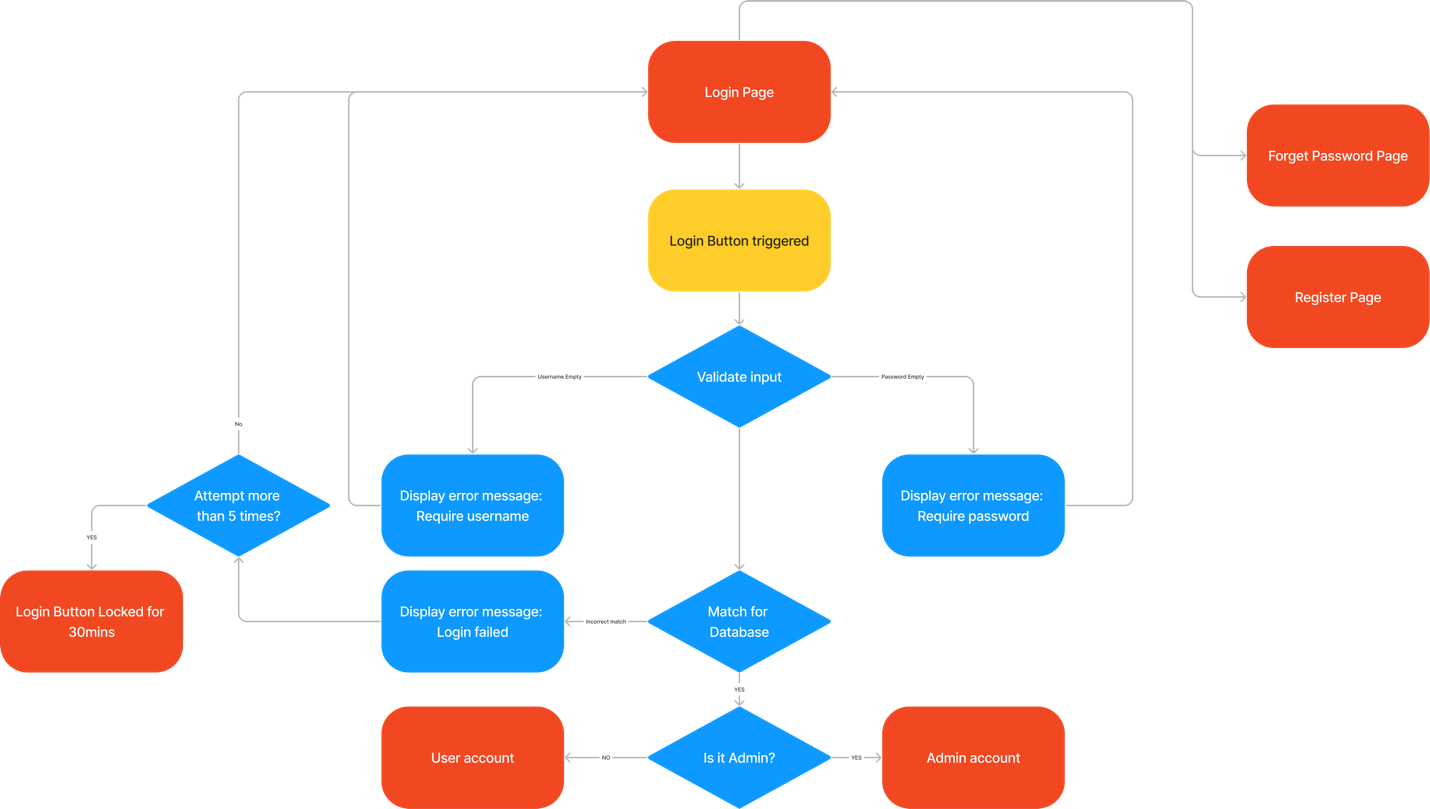
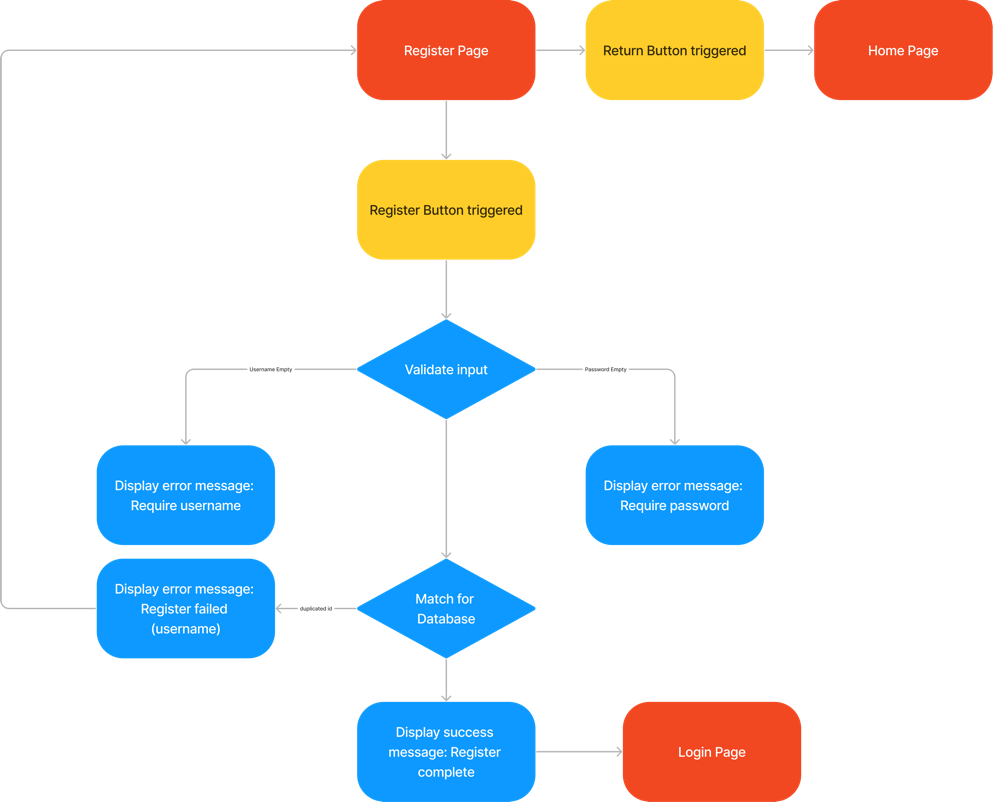
About Page:

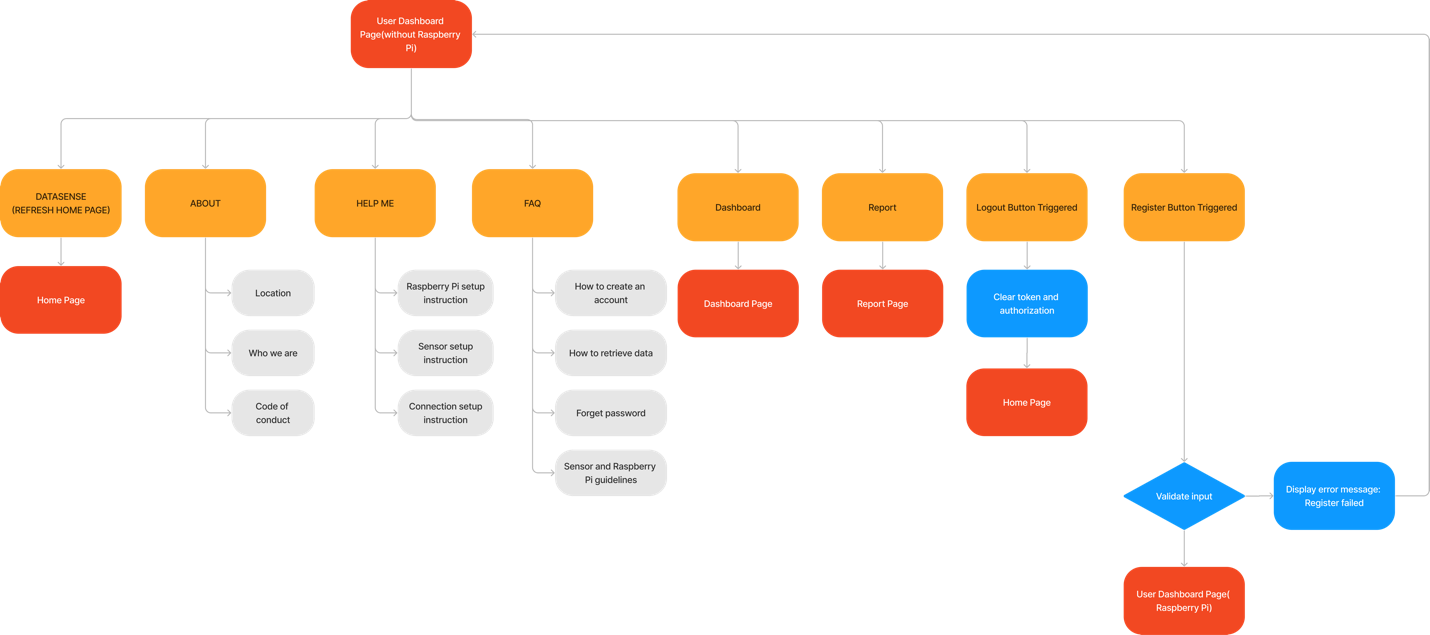


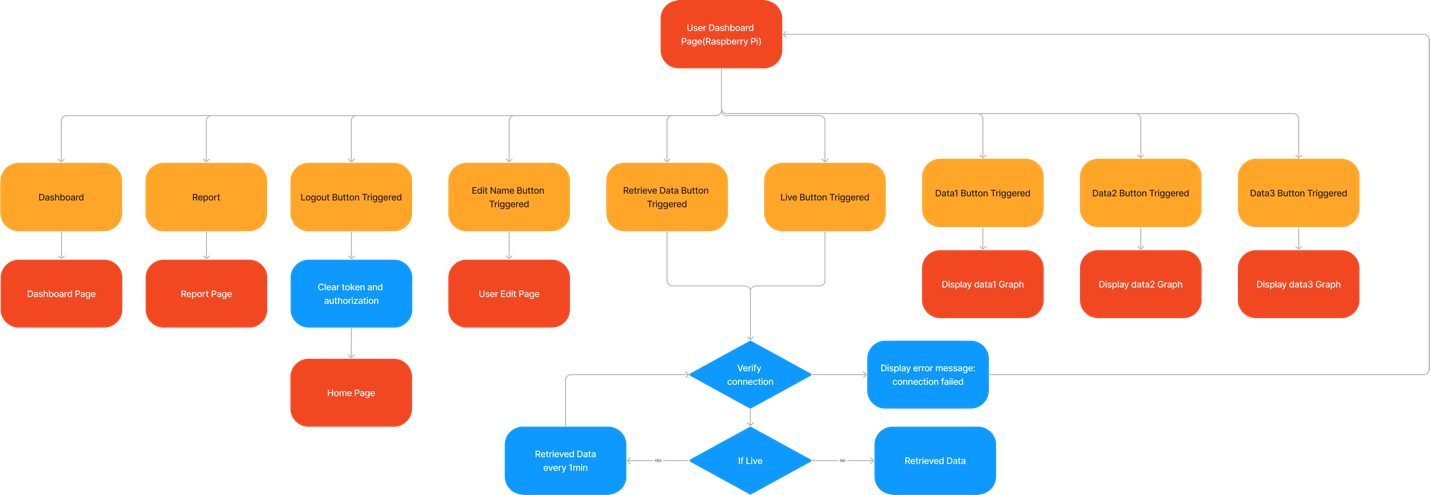
Help Page:



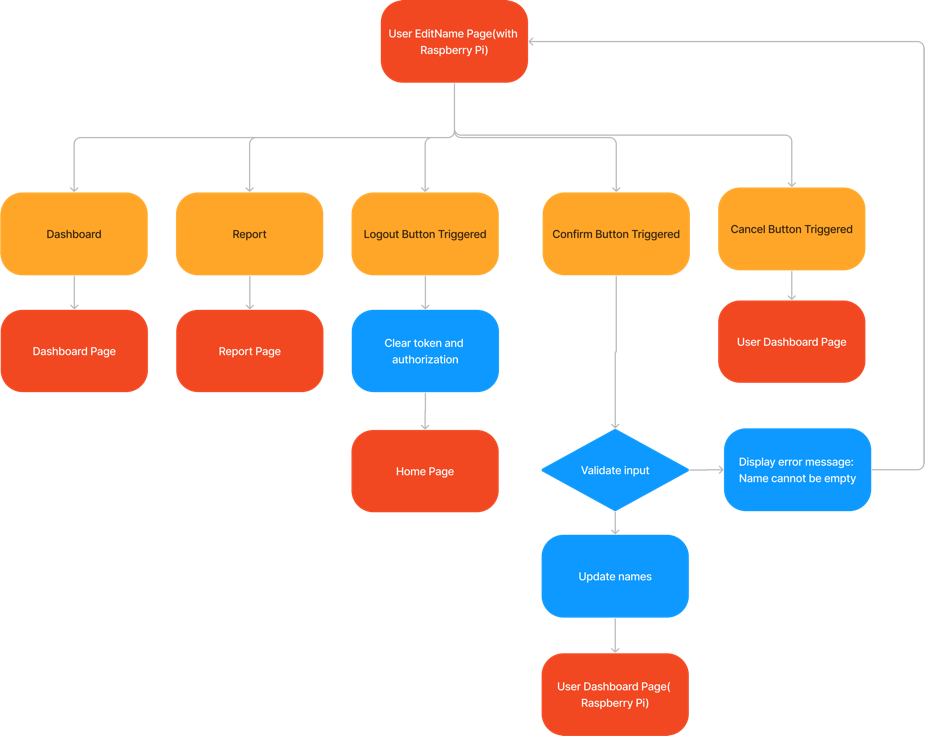
Home workflow diagram:  


Login workflow Diagram:  
Register workflow Diagram:  


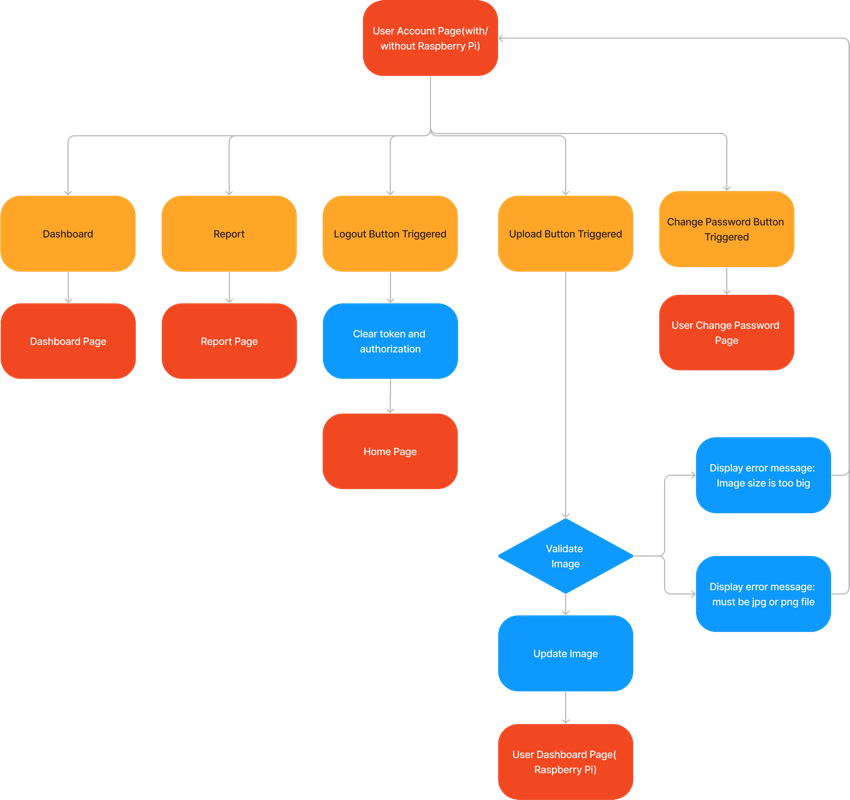
User Dashboard (without Raspberry Pi) workflow diagram:  
  
User Dashboard Page (Raspberry Pi) workflow diagram:



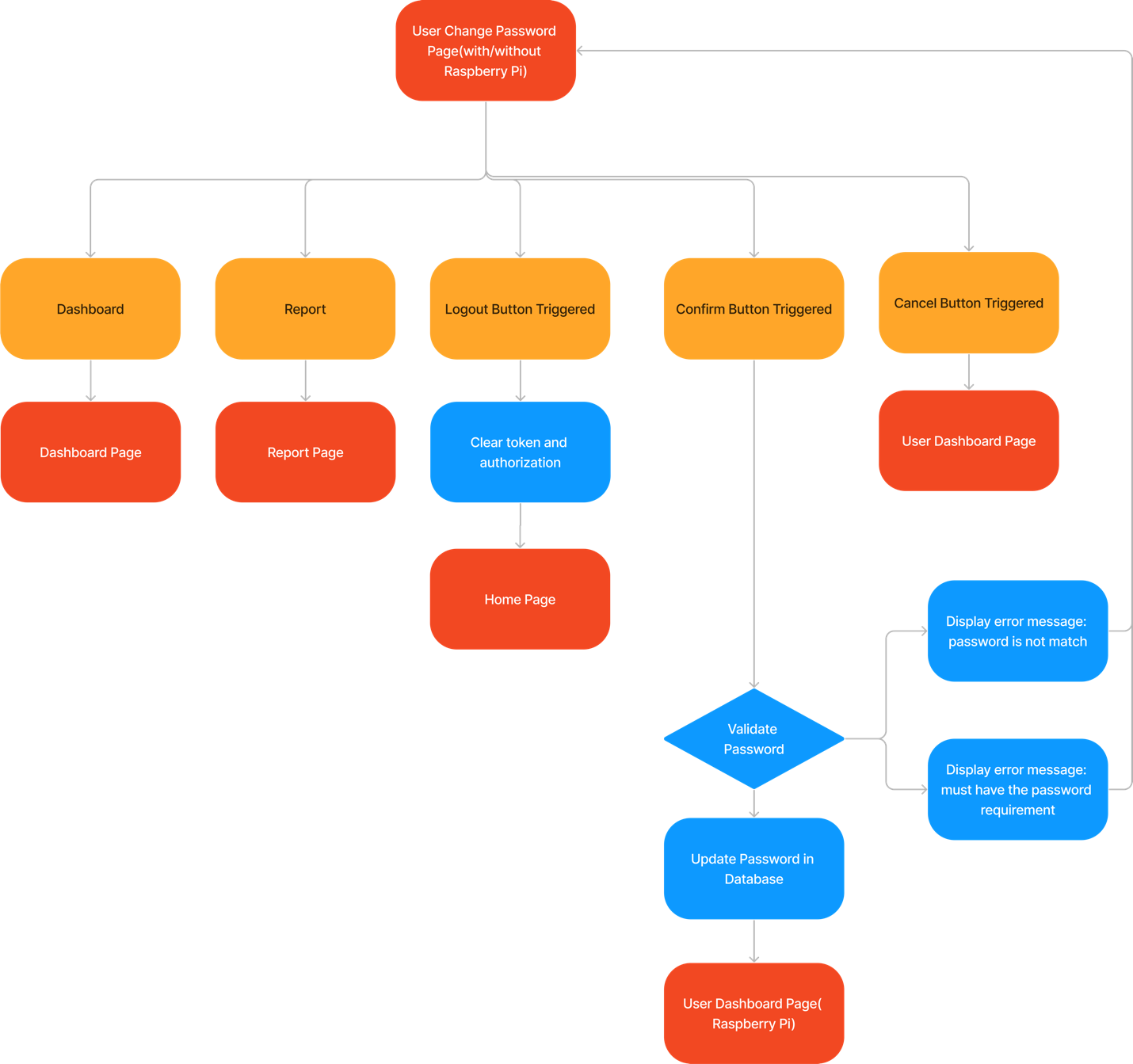
User Edit Name Page (with Raspberry Pi) workflow diagram:



User Account Page (with/without Raspberry Pi) workflow diagram:



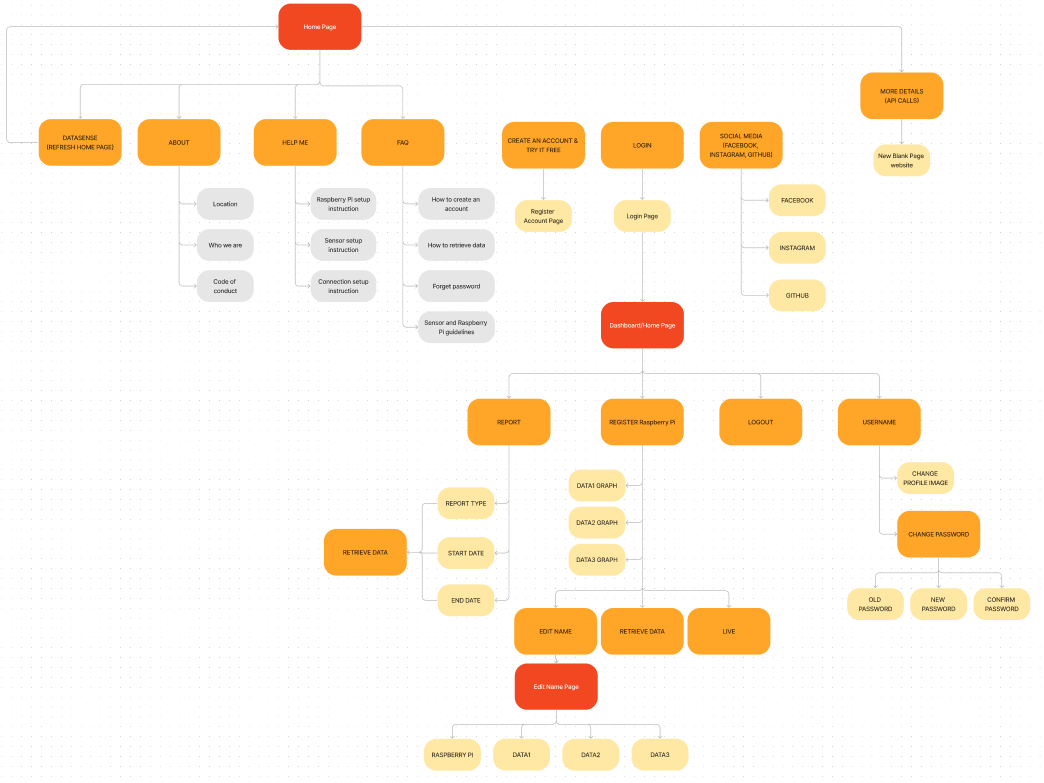
User Change Password Page(with/without Raspberry Pi) workflow diagram:



Section 3 – Process and Data Modelling

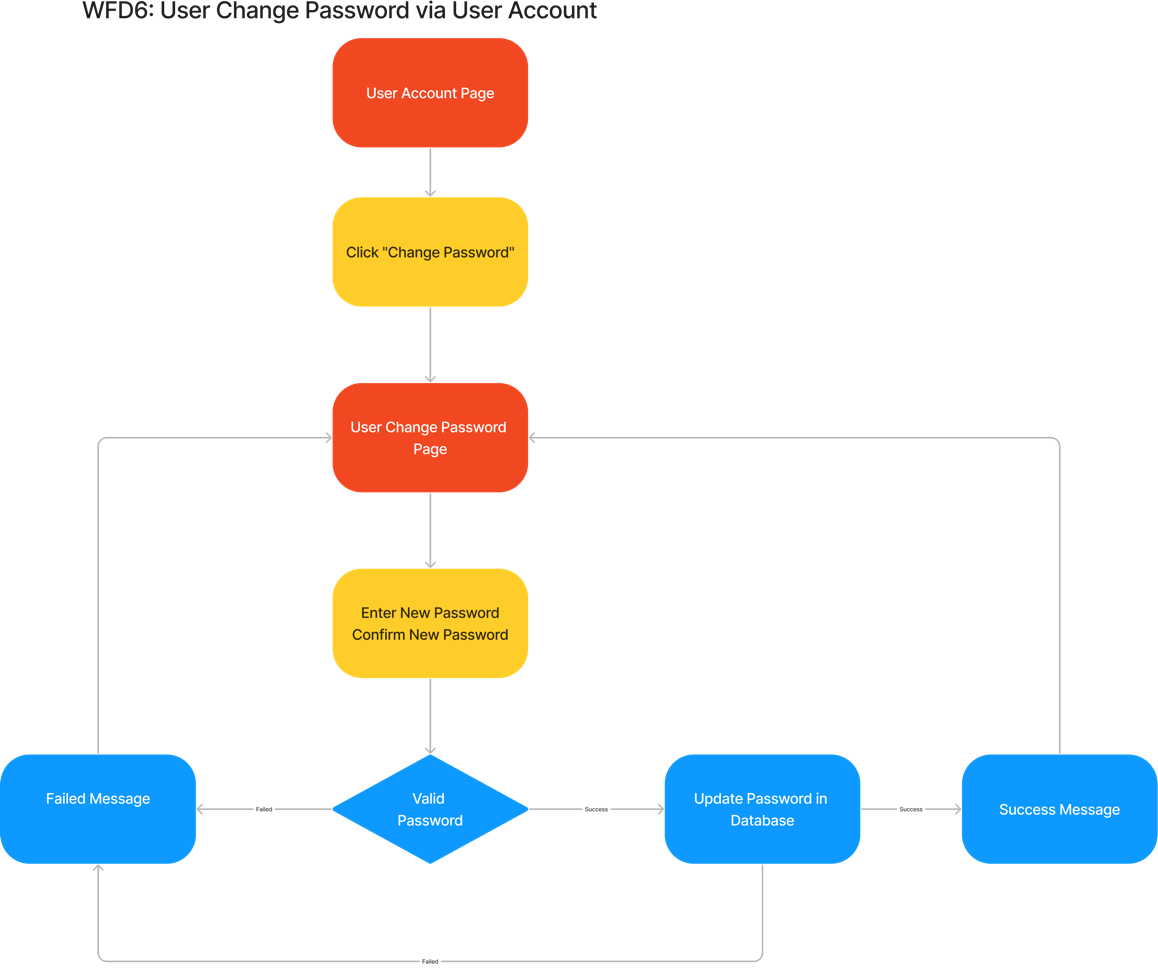
# 3.1 Workflow Diagrams

## 3.1.1 Application Navigation

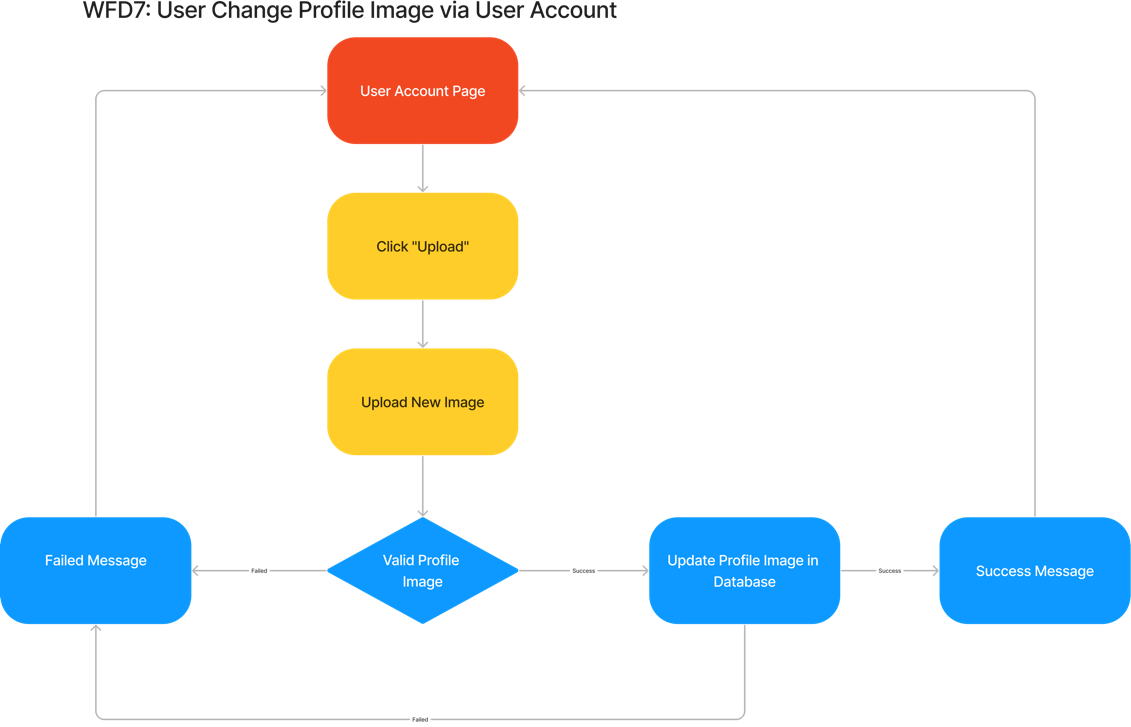


## 3.1.2 End User Workflow

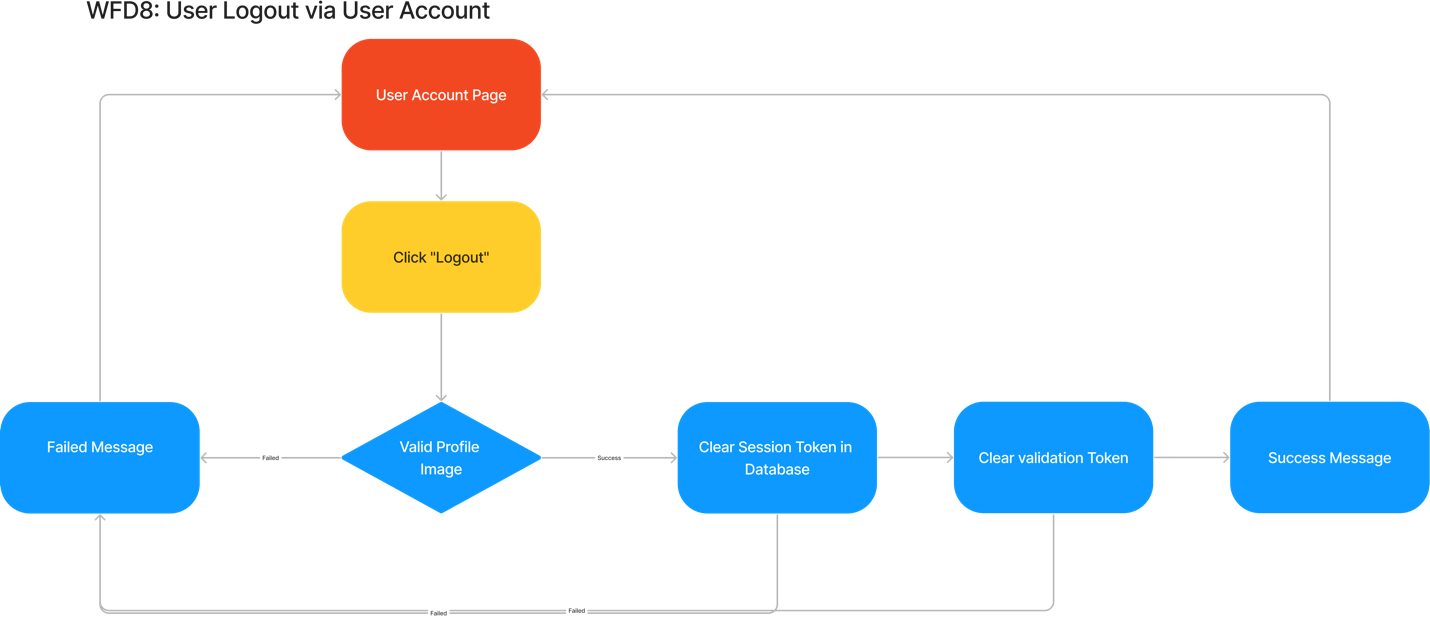
## 3.1.2.6 - WFD6: User Change Password via User Account



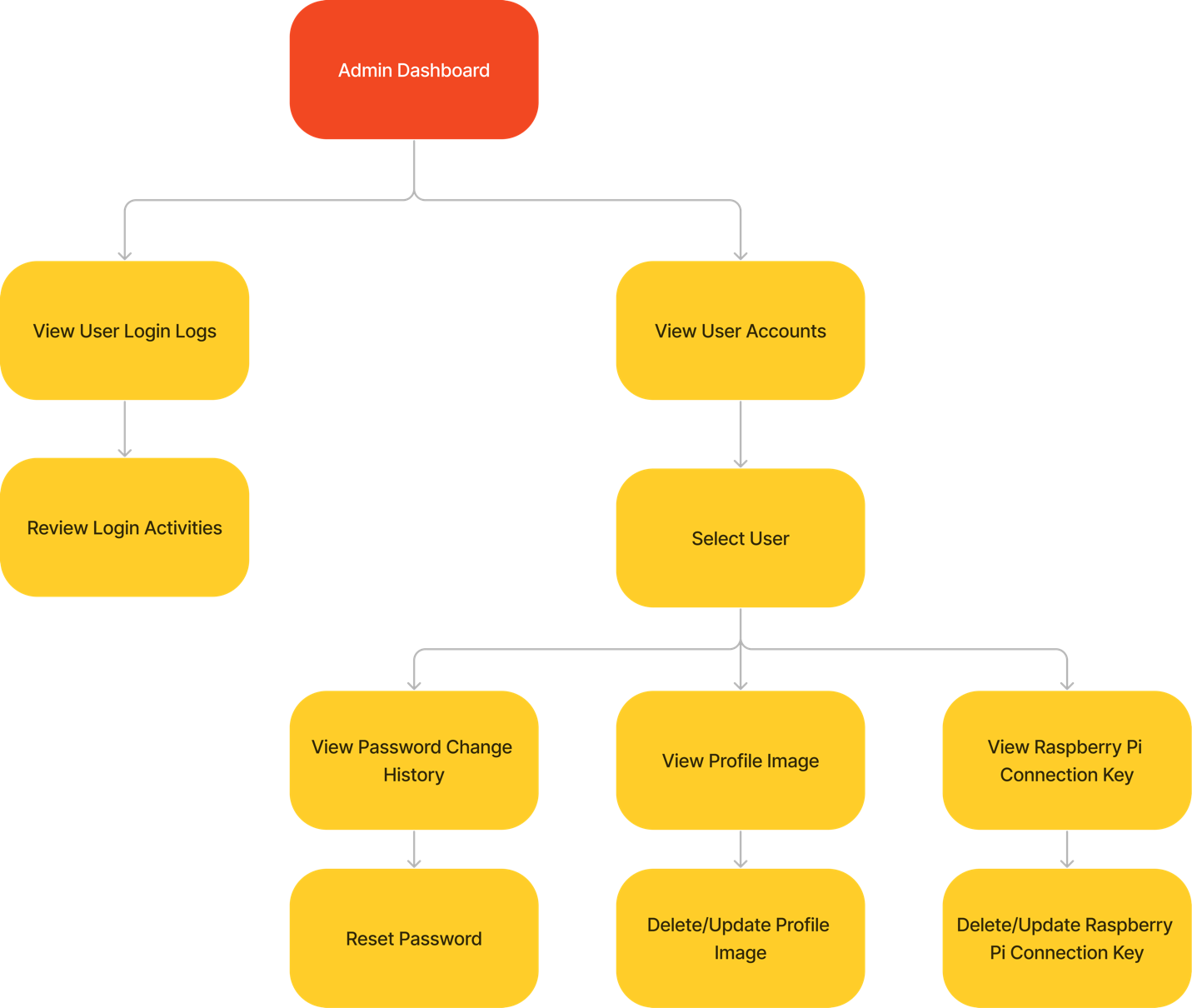
## 3.1.2.7 - WFD7: User Change Profile Image via User Account



## 3.1.2.8 - WFD8: User Logout via User Account



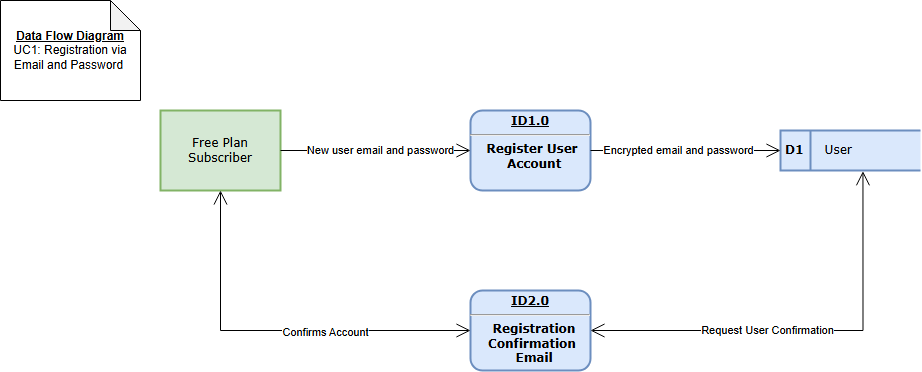
## 3.1.3 Administration Workflow



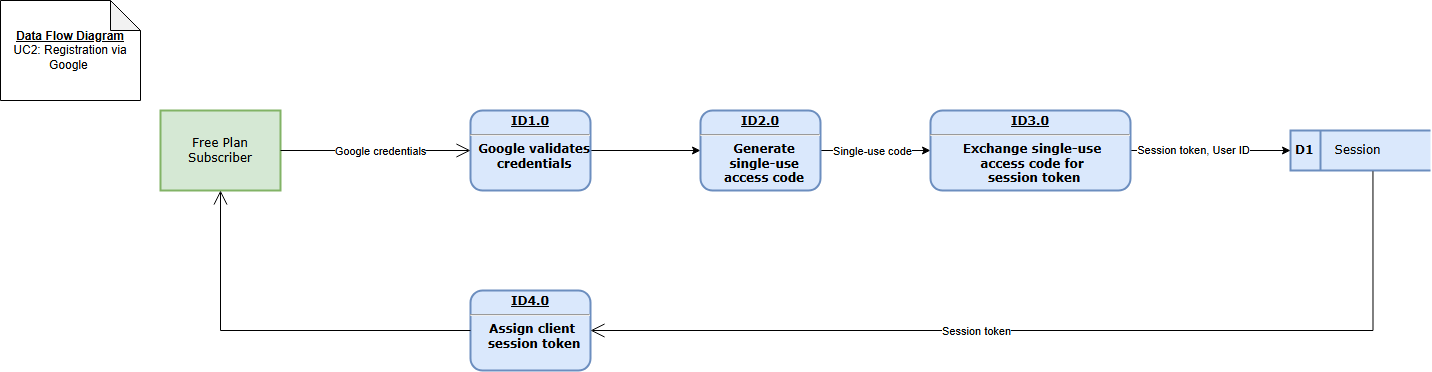
# 3.2 Data Modelling and Flow

|  |  |
| --- | --- |
| **Use Case Scenario #** | **Description** |
| DFD1 | Registration via Email and Password |
| DFD2 | Registration via Google Auth |
| DFD3 | User Adds Raspberry Pi |
| DFD4 | User Login via Email and Password |
| DFD5 | User Login via Google Auth |
| DFD6 | User Change Password via User Account |
| DFD7 | User Change Profile Image via User Account |
| DFD8 | User Logout via User Account |
| DFD9 | View Raspberry Pi Data |
| DFD10 | Delete User via Email and Password |
| DFD11 | Update User Profile Information |
| DFD12 | Configure Personalized Alerts |
| DFD13 | User Generates Custom Reports |

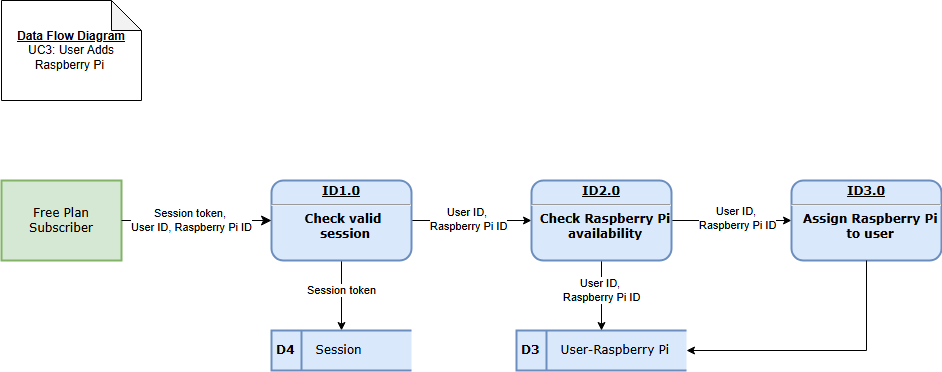
## 3.2.1 DFD1 - Registration via Email and Password



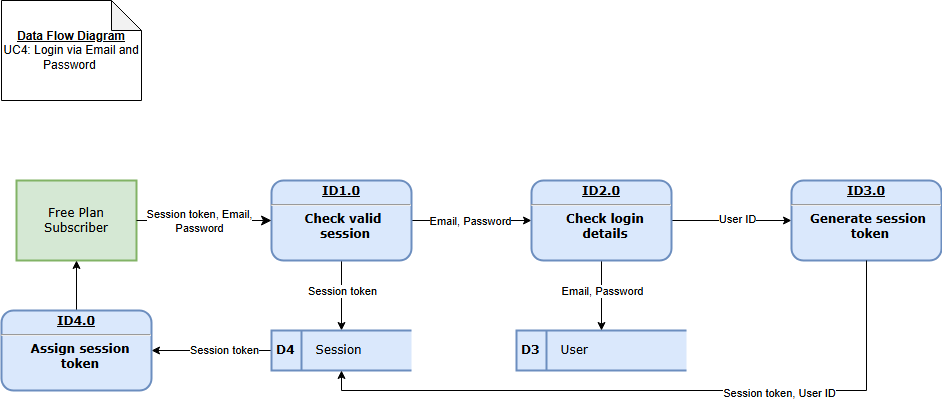
## 3.2.2 DFD2 - Registration via Google Auth



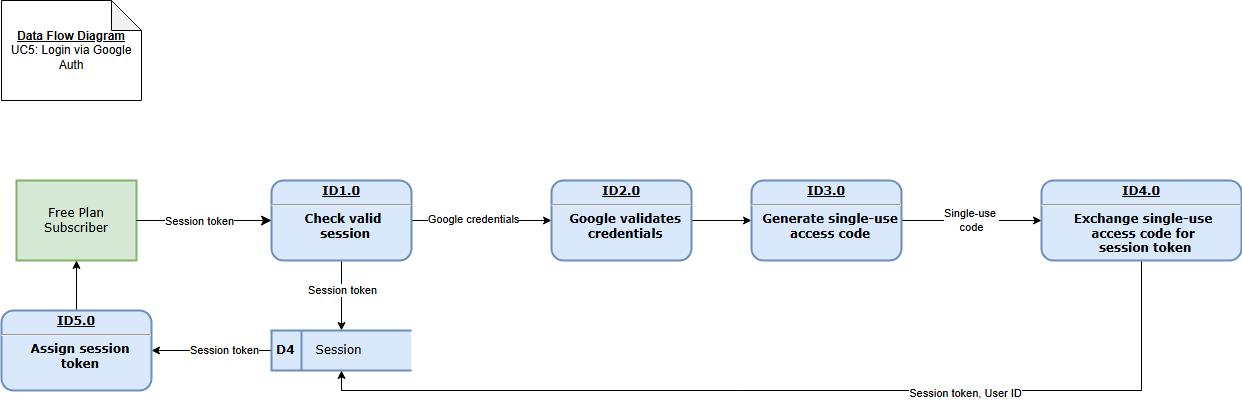
## 3.2.3 DFD3 – User Adds a Raspberry Pi



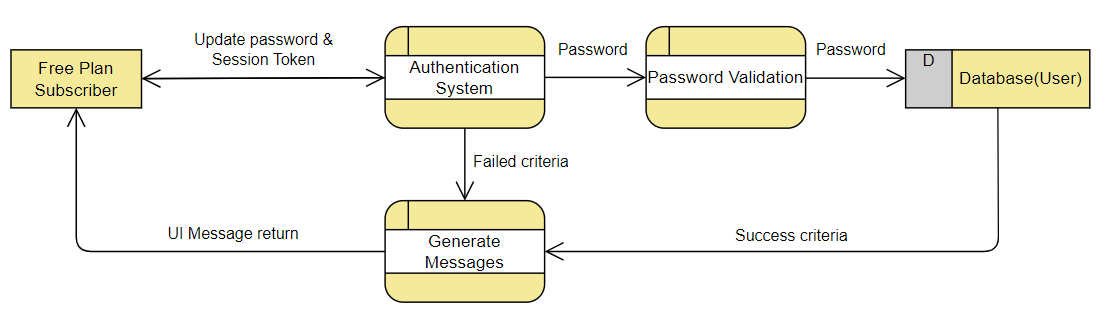
## 3.2.4 DFD4 - User Login via Email and Password



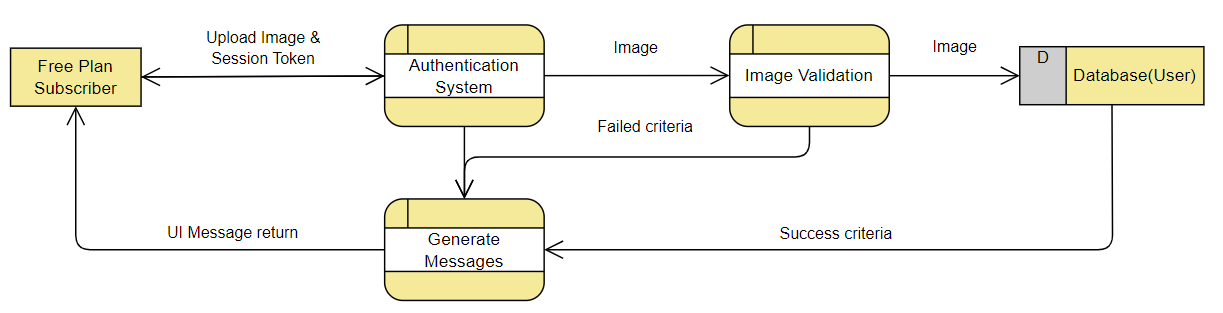
## 3.2.5 DFD5 - User Login via Google Auth



## 3.2.6 DFD6 - User Change Password via User Account

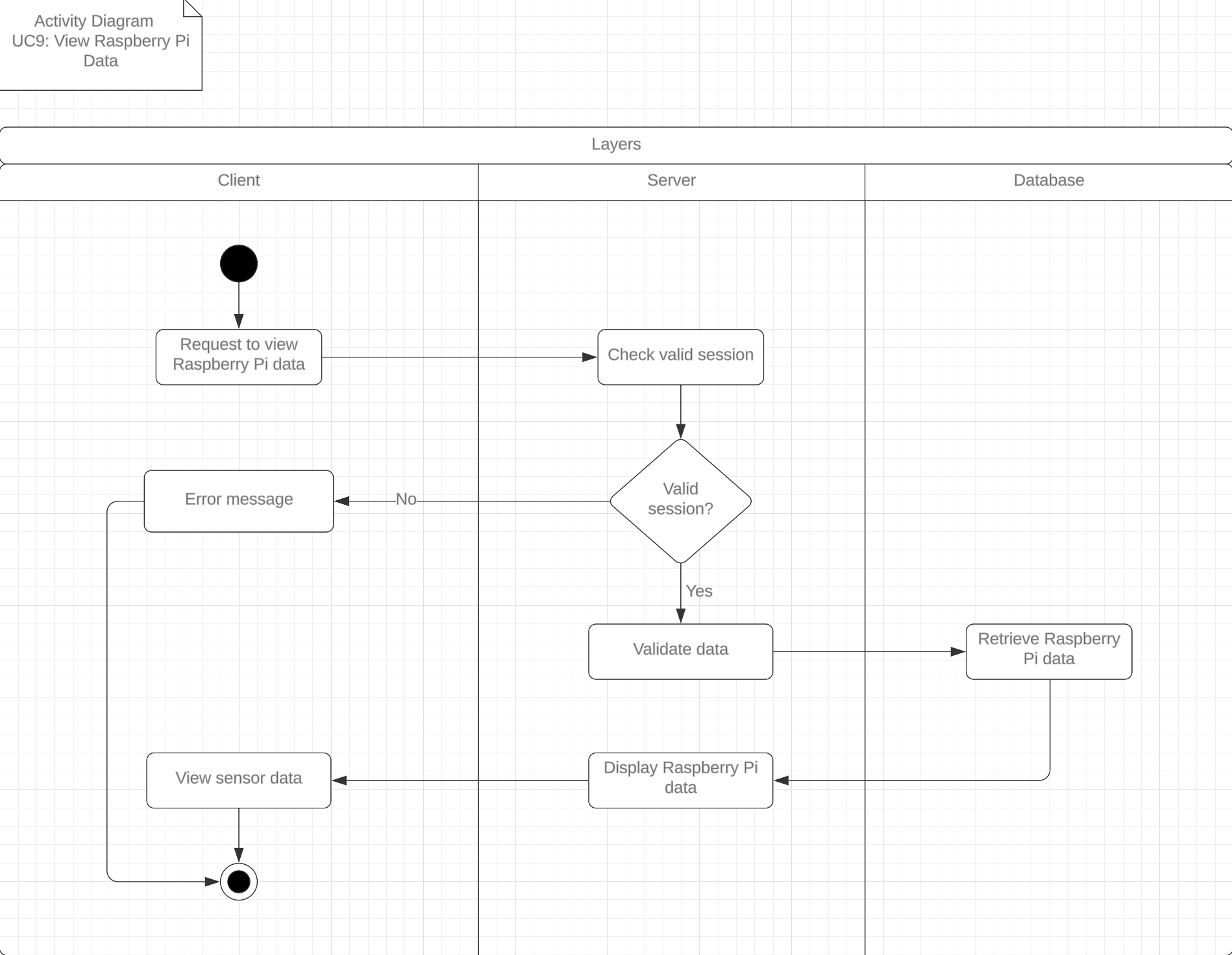


## 3.2.7 DFD7 - User Change Password via User Account

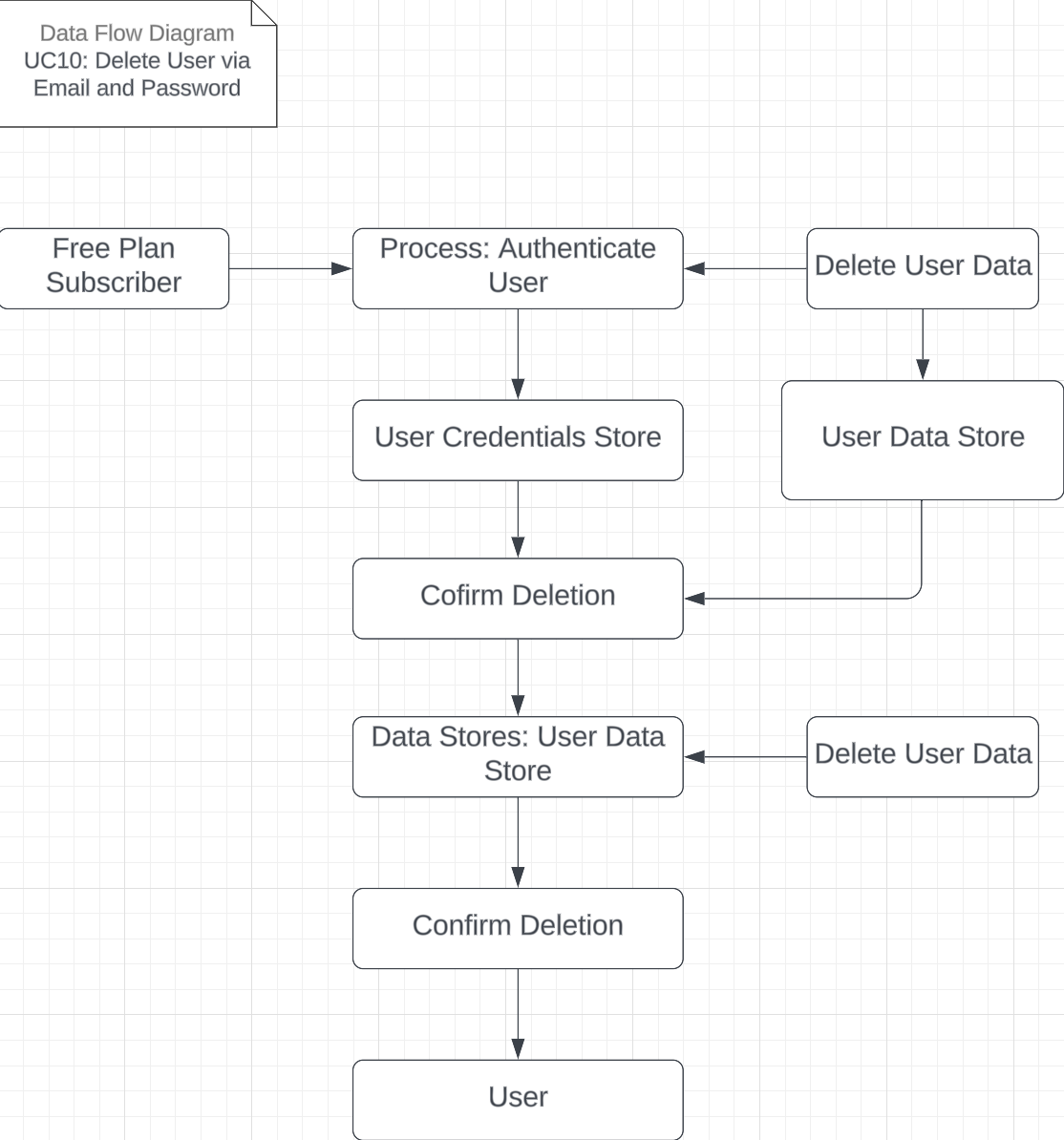


## 3.2.8 DFD8 - User Logout via User Account

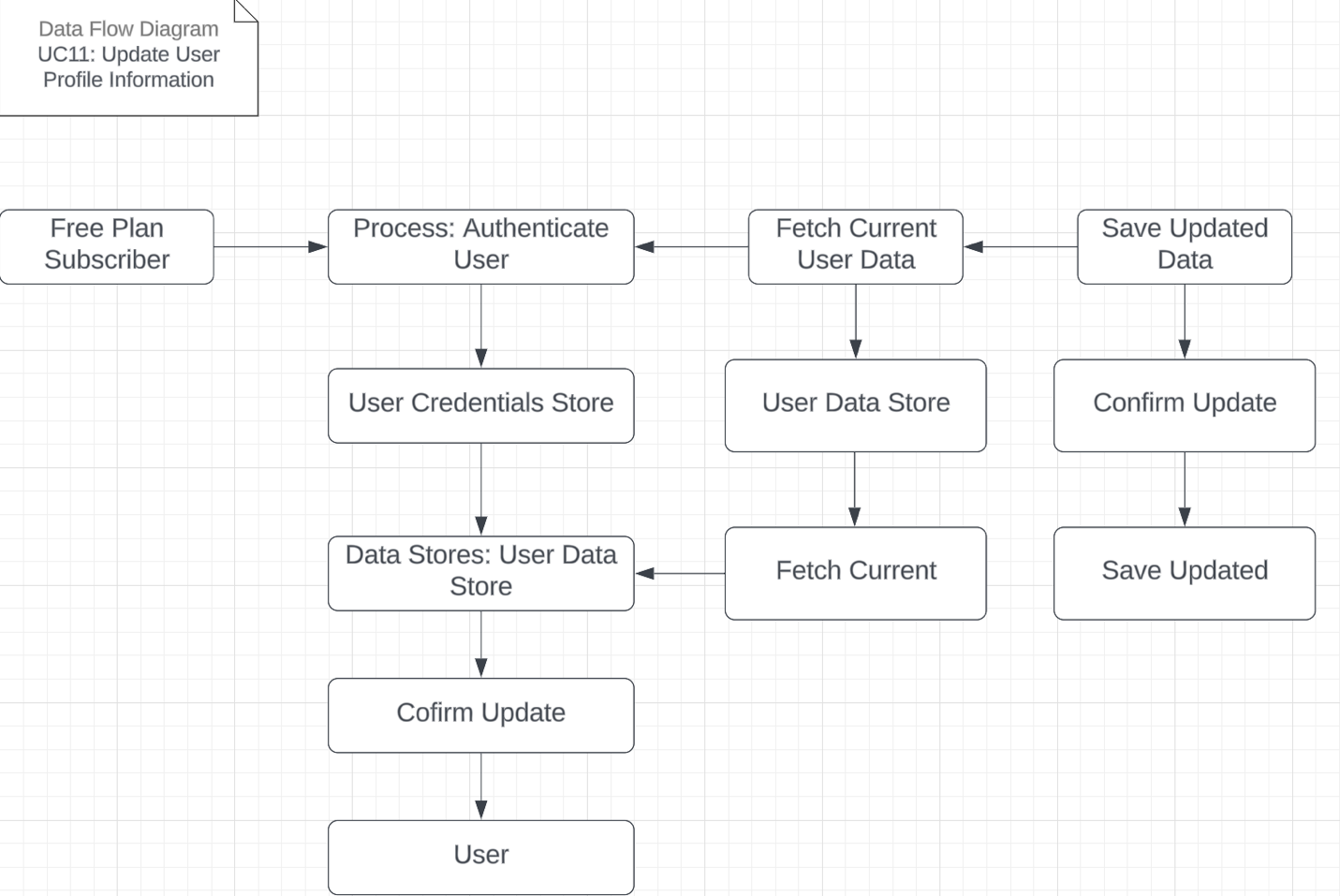
## 3.2.9 DFD9 – View Raspberry Pi Data



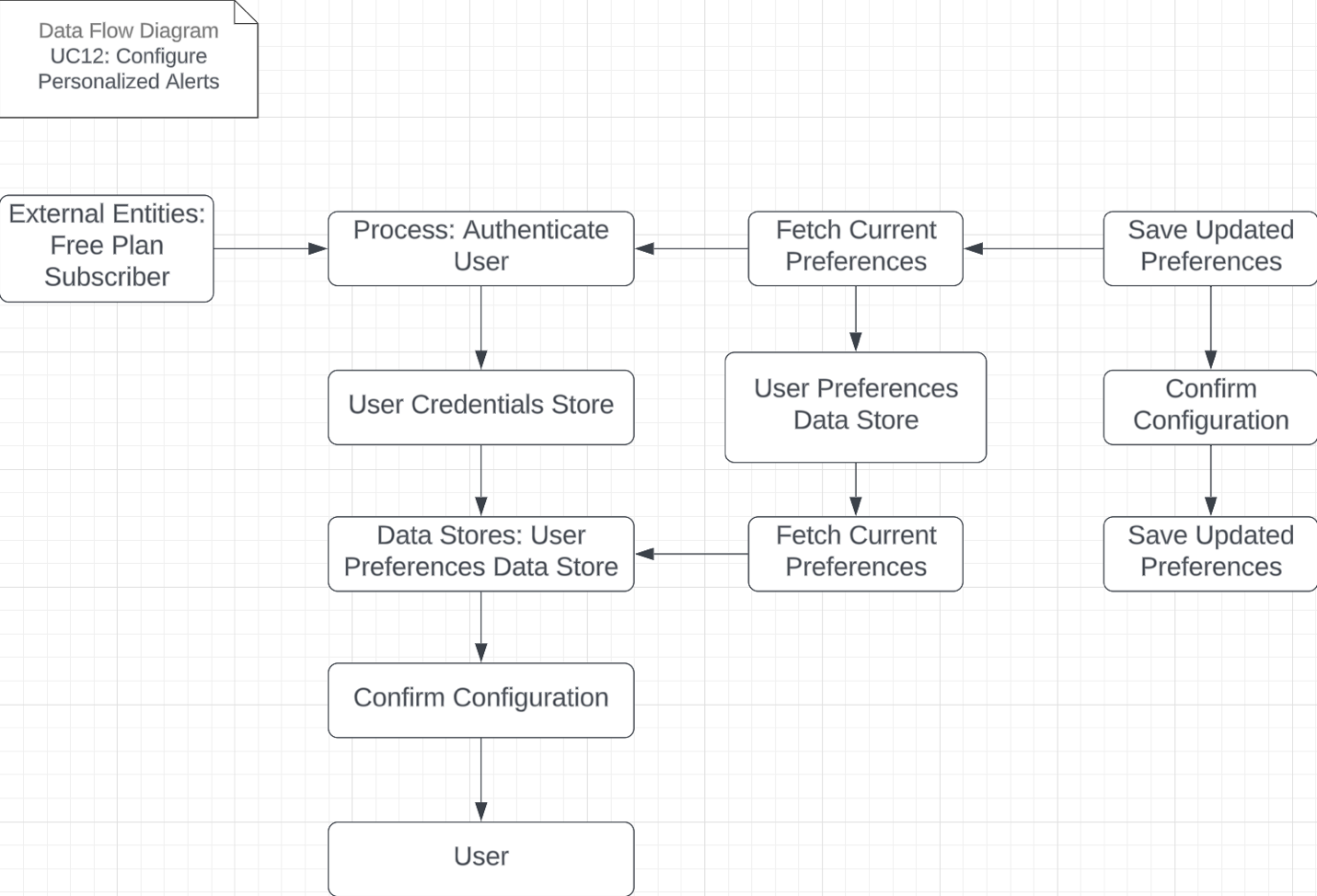
## 3.2.10 DFD10 – Delete User via Email and Password



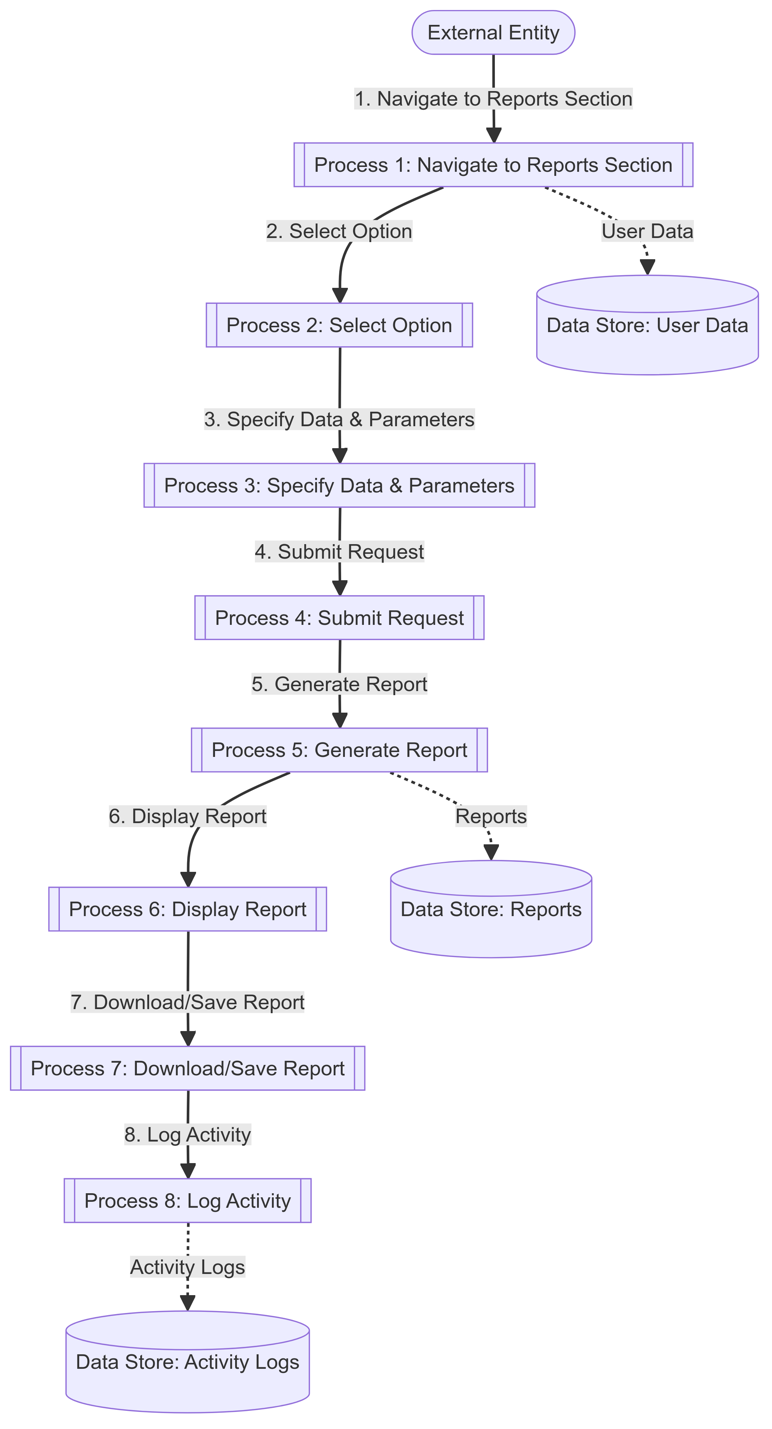
## 3.2.11 DFD11 – Update Username



## 3.2.12 DFD12 – Configure Personalized Alerts



## 3.2.13 DFD13 – User Generates Custom Reports



## 3.3 Use Case Scenarios

The following Use Case Scenarios (UCSs) have been identified. The following table is a summary of the UCSs followed by detailed descriptions.

|  |  |
| --- | --- |
| **Use Case Scenario #** | **Description** |
| UC1 | Registration via Email and Password |
| UC2 | Registration via Google Auth |
| UC3 | User Adds Raspberry Pi |
| UC4 | User Login via Email and Password |
| UC5 | User Login via Google Auth |
| UC6 | User Change Password via User Account |
| UC7 | User Change Profile Image via User Account |
| UC8 | User Logout via User Account |
| UC9 | View Raspberry Pi Data |
| UC10 | Delete User via Email and Password |
| UC11 | Update User Profile Information |
| UC12 | Configure Personalized Alerts |
| UC13 | User Generates Custom Reports |

## 3.3.1 UC1 – Registration via Email and Password

|  |  |
| --- | --- |
| Scenario Name | Registration via Email and Password |
| Actors | Free plan subscriber |
| Stakeholders and Interests | Development team |
| Description | Registration process of a new user |
| Assumptions, Constraints, and/or Pre-Conditions | * User does not have an account * Password must meet the criteria:   + 1 number   + 1 special character   + 1 uppercase   + 1 lowercase   + 8+ characters |
| Trigger – Starting Point | User is currently on the sign-up page |
| Relationships | N/A |
| Normal Flow of Events | 1. Registration page 2. Input email 3. Input password 4. Receive registration confirmation link 5. Click on registration confirmation link 6. Redirected to login page |
| Sub-Flows | N/A |
| Alternate/Exceptional Flows | S-1, 2a1: User does not click on confirmation link   1. The user does not complete their registration by not confirming their account by email link |

## 3.3.2 UC2 – Registration via Google Auth

|  |  |
| --- | --- |
| Scenario Name | Registration via Google Auth |
| Actors | Free plan subscriber |
| Stakeholders and Interests | Development team |
| Description | Registration process of a new user using Google OAuth |
| Assumptions, Constraints, and/or Pre-Conditions | * User does not have an account * User has a Google account |
| Trigger – Starting Point | User is currently on the sign-up page |
| Relationships | N/A |
| Normal Flow of Events | 1. The user is on the registration page 2. The user clicks on “Register with Google” 3. Redirected to Google OAuth    1. Select Google account    2. Input Google credentials       1. Input GMail       2. Input password 4. If Google cannot authenticate user    1. Return the step 3    2. If the user is locked out of Google OAuth:       1. S-1, 2a1 sub-flow is performed 5. Redirected to login page |
| Sub-Flows | N/A |
| Alternate/Exceptional Flows | S-1, 2a1: Cancel Registration   1. The user cancels or exits the Google OAuth workflow 2. The user is redirected to the registration page |

## 3.3.3 UC3 – User Adds Raspberry Pi

|  |  |
| --- | --- |
| Scenario Name | User Adds Raspberry Pi |
| Actors | Free plan subscriber |
| Stakeholders and Interests | Development team, Admin |
| Description | User adds a Raspberry Pi to their account, enabling them to view the sensor data connected to the Raspberry Pi |
| Assumptions, Constraints, and/or Pre-Conditions | * User is currently logged in * User has a valid session token |
| Trigger – Starting Point | User is prompted to enter the Raspberry Pi ID |
| Relationships | N/A |
| Normal Flow of Events | 1. The user is on the “Add Raspberry Pi” page 2. The inputs the Raspberry Pi ID 3. Submits “Add Raspberry Pi” form 4. If the Raspberry Pi ID registration fails:    1. S-1, 2a1 sub-flow is performed 5. Else If: User cancels Raspberry Pi registration    1. S-2, 2a2 sub-flow is performed 6. Else:    1. The user receives a success message 7. Redirect to “My Devices” page |
| Sub-Flows | Authorized Request (Appendix A, Figure 1) |
| Alternate/Exceptional Flows | S-1, 2a1: Raspberry Pi Registration Fails   1. The user receives error message 2. Prompted to input another Raspberry Pi ID   S-2 , 2a2: User Cancels Raspberry Pi Registration   1. Redirect to “My Devices” Page |

## 3.3.4 UC4 – User Login via Email and Password

|  |  |
| --- | --- |
| Scenario Name | User Login via Email and Password |
| Actors | Free plan subscriber |
| Stakeholders and Interests | Development team, Admin |
| Description | Existing user who previously registered with email logs into their account |
| Assumptions, Constraints, and/or Pre-Conditions | * User created their account using their email |
| Trigger – Starting Point | User is currently on the login page |
| Relationships | N/A |
| Normal Flow of Events | 1. The user is on the “Login” page 2. The user inputs their email 3. The user inputs their password 4. Submits login form 5. If login fails    1. S-1, 2a1 sub-flow is performed 6. Else If: User cancels Raspberry Pi registration    1. S-2, 2a2 sub-flow is performed 7. Else:    1. The user receives a success message 8. Redirect to “Dashboard” page |
| Sub-Flows | Authorized Request (Appendix A, Figure 1) |
| Alternate/Exceptional Flows | S-1, 2a1: Login Credentials are Incorrect   1. The user receives error message 2. Prompted to input their email and password   S-2, 2a2 User Cancels Raspberry Pi Registration   1. Redirect to “Home” Page |

## 3.3.5 UC5 – User Login via Google Auth

|  |  |
| --- | --- |
| Scenario Name | User Login via Google Auth |
| Actors | Free plan subscriber |
| Stakeholders and Interests | Development team, Admin |
| Description | Existing user who previously registered with Google logs into their account |
| Assumptions, Constraints, and/or Pre-Conditions | * User created their account using Google |
| Trigger – Starting Point | User is currently on the login page |
| Relationships | N/A |
| Normal Flow of Events | 1. The user is on the “Login” page 2. The user clicks on “Login with Google” 3. Redirected to Google OAuth    1. Select Google account    2. Input Google credentials       1. Input GMail       2. Input password 4. If Google cannot authenticate user    1. Return the step 3    2. If the user is locked out of Google OAuth:       1. S-1, 2a2 sub-flow is performed 5. Redirected to “Dashboard” page |
| Sub-Flows | Authorized Request (Appendix A, Figure 1) |
| Alternate/Exceptional Flows | S-1, 2a1: Login Credentials are Incorrect   1. The user receives error message 2. Prompted to input their email and password   S-2, 2a2 User is Locked out of Google OAuth   1. The user exits the Google OAuth workflow 2. The user if redirected to the “Login” page |

## 3.3.6 UC6 – User Change Password via User Account

|  |  |
| --- | --- |
| Scenario Name | User Change Password via User Account |
| Actors | Free plan subscriber |
| Stakeholders and Interests | Development team, Admin |
| Description | An existing user changes/updates their password in their account. |
| Assumptions, Constraints, and/or Pre-Conditions | * User is currently logged in * Password must meet the criteria:   + 1 number   + 1 special character   + 1 uppercase   + 1 lowercase   + 8+ characters * Confirmed password validation   + The confirmed password must match the new password exactly. * The userID password must be updated in the 'user' table. |
| Trigger – Starting Point | User is currently on the user account page |
| Relationships | N/A |
| Normal Flow of Events | 1. The user is on the “User Account” page 2. The user clicks on “Change Password” 3. Redirected to “User Change Password” Page 4. Type in old password, new password, and confirm password 5. System checks if the old password matches, the new password and confirm password match, and the new password meets the specified criteria:  * If Failed:  1. Return to “User Change Password” Page 2. Display the proper error message  * If Passed:  1. Update the password in the 'user' table. 2. User is redirected back to the “User Account” Page. 3. Display a success message indicating the password change was successful. |
| Sub-Flows | Authorized Request (Appendix A, Figure 1) |
| Alternate/Exceptional Flows | * Password does not meet the specified criteria. * Confirmed password does not match the new password. * System error while updating the password in the database. |

## 3.3.7 UC7 – User Change Profile Image via User Account

|  |  |
| --- | --- |
| Scenario Name | User Change Profile Image via User Account |
| Actors | Free plan subscriber |
| Stakeholders and Interests | Development team, Admin |
| Description | An existing user changes/updates their profile image in their account. |
| Assumptions, Constraints, and/or Pre-Conditions | * User is currently logged in * Profile image must meet the criteria:   + Image Format - JPEG and PNG   + Image Size - 5MB max   + Image Dimensions (min:100x100 pixel, max:1000x1000 pixel)   + Image Content (No offensive or inappropriate content) * The userID profile image must be updated in the 'user' table. |
| Trigger – Starting Point | User is currently on the user account page |
| Relationships | N/A |
| Normal Flow of Events | 1. The user is on the “User Account” page 2. The user clicks on “Upload” 3. User selects and uploads a new profile image. 4. System validates the image format, size, dimensions, and content:  * If Failed:  1. Display the proper error message  * If Passed:  1. Update the profile image in the 'user' table. 2. User is redirected back to the “User Account” Page. 3. Display a success message indicating the profile image change was successful. |
| Sub-Flows | Authorized Request (Appendix A, Figure 1) |
| Alternate/Exceptional Flows | * Image format is not supported. * Image size exceeds the maximum limit. * Image dimensions do not meet the criteria. * Offensive or inappropriate content detected in the image. * System error while updating the profile image in the database. |

## 3.3.8 UC8 – User Logout via User Account

|  |  |
| --- | --- |
| Scenario Name | User Logout via User Account |
| Actors | Free plan subscriber |
| Stakeholders and Interests | Development team, Admin |
| Description | An existing user logs out of their account from the website. |
| Assumptions, Constraints, and/or Pre-Conditions | * User is currently logged in * Session token is valid |
| Trigger – Starting Point | The user clicks the logout button. |
| Relationships | N/A |
| Normal Flow of Events | 1. The user clicks the "Logout" button on the website. 2. The system invalidates the user's session token. 3. The user is redirected to the “Home” page. 4. Display a message confirming the user has successfully logged out. |
| Sub-Flows | Authorized Request (Appendix A, Figure 1) |
| Alternate/Exceptional Flows | * Session token has expired. * Network or server error during the logout process. |

## 3.3.9 UC9 – View Raspberry Pi Data

|  |  |
| --- | --- |
| Scenario Name | View Raspberry Pi Data |
| Actors | Free plan subscriber |
| Stakeholders and Interests | Development team, Admin |
| Description | An existing user views data from their Raspberry Pi |
| Assumptions, Constraints, and/or Pre-Conditions | * User has a Raspberry Pi device linked to their account |
| Trigger – Starting Point | User is currently on the login page |
| Relationships | N/A |
|  | Authorized Request (Appendix A, Figure 1) |
| Alternate/Exceptional Flows | * User has no Raspberry Pi devices linked |

## 3.3.10 UC10 – Delete User via Email and Password

|  |  |
| --- | --- |
| Scenario Name | Delete User via Email and Password |
| Actors | Free plan subscriber |
| Stakeholders and Interests | Development team, Admin |
| Description | An existing user who password logs into their account and deletes their user profile |
| Assumptions, Constraints, and/or Pre-Conditions | * User created their account using email and password and user is authenticated successfully |
| Trigger – Starting Point | User is currently on the login page |
| Relationships | N/A |
| Sub-Flows | Authorized Request (Appendix A, Figure 1) |
| Alternate/Exceptional Flows | * User changes their mind and cancels the delete operation |

## 3.3.11 UC11 – Update User Profile Information

|  |  |
| --- | --- |
| Scenario Name | Update User Profile Information |
| Actors | Free plan subscriber |
| Stakeholders and Interests | Development team, Admin |
| Description | An existing user updates their profile information |
| Assumptions, Constraints, and/or Pre-Conditions | * User is authenticated successfully |
| Trigger – Starting Point | User is currently on the login page |
| Relationships | N/A |
| Sub-Flows | Authorized Request (Appendix A, Figure 1) |
| Alternate/Exceptional Flows | * User fails to authenticate and user inputs invalid profile information |

## 3.3.12 UC12 – Configure Personalized Alerts

|  |  |
| --- | --- |
| Scenario Name | Configure Personalized Alerts |
| Actors | Free plan subscriber |
| Stakeholders and Interests | Development team, Admin |
| Description | An existing user configures personalized alerts for their Raspberry Pi data |
| Assumptions, Constraints, and/or Pre-Conditions | * User created their account and user has a Raspberry Pi device linked to their account |
| Trigger – Starting Point | User is currently on the login page |
| Relationships | N/A |
| Sub-Flows | Authorized Request (Appendix A, Figure 1) |
| Alternate/Exceptional Flows | * User inputs invalid alert configuration |

## 3.3.13 UC13 – User Generates Custom Reports

|  |  |
| --- | --- |
| Scenario Name | User Generates Custom Reports |
| Actors | User, System |
| Stakeholders and Interests | User: Wants to generate custom reports with specific data and parameters.  System: Ensures the report generation process is accurate and efficient. |
| Description | The user generates custom reports by selecting specific data points and parameters. |
| Assumptions, Constraints, and/or Pre-Conditions | User is logged into their account.  System supports custom report generation. |
| Trigger – Starting Point | User navigates to the reports section and selects the option to generate a custom report. |
| Relationships | None |
| Normal Flow of Events | 1. User navigates to the reports section.  2. User selects the option to generate a custom report.  3. System prompts the user to specify data points and parameters.  4. User enters the required details and submits the request.  5. The system generates the custom report based on the provided parameters.  6. System displays the generated report to the user.  7. Users can download or save the report.  8. System logs the report generation activity. |
| Alternate/Exceptional Flows | If the report generation fails, the system notifies the user and prompts them to adjust the parameters. |

## 3.3.14 UC14 – Administrator Manages System Backups

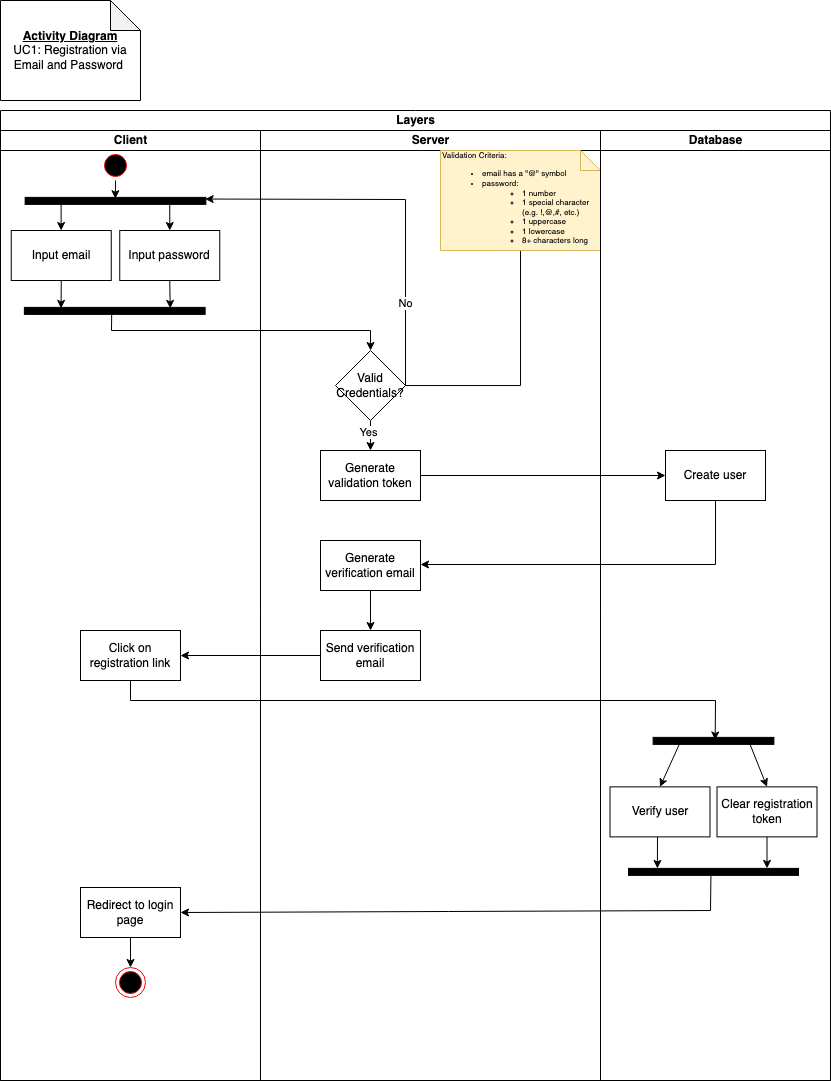
|  |  |
| --- | --- |
| Scenario Name | Administrator Manages System Backups |
| Actors | Administrator, System |
| Stakeholders and Interests | Administrator: Wants to ensure regular backups of system data.  System: Performs backups and ensures data integrity. |
| Description | The administrator schedules and manages system backups to ensure data integrity and availability. |
| Assumptions, Constraints, and/or Pre-Conditions | The administrator is logged into the admin account.  System supports backup management. |
| Trigger – Starting Point | Administrator navigates to the backup management section. |
| Relationships | None |
| Normal Flow of Events | 1. Administrator navigates to the backup management section.  2. System displays the current backup schedule and status.  3. Administrator schedules new backups or manages existing ones.  4. System performs backups according to the schedule.  5. Administrator verifies the backup status and data integrity.  6. System logs the backup activity and any errors encountered. |
| Alternate/Exceptional Flows | If a backup fails, the system notifies the administrator and attempts to retry. |

# 3.4 Activity Diagrams

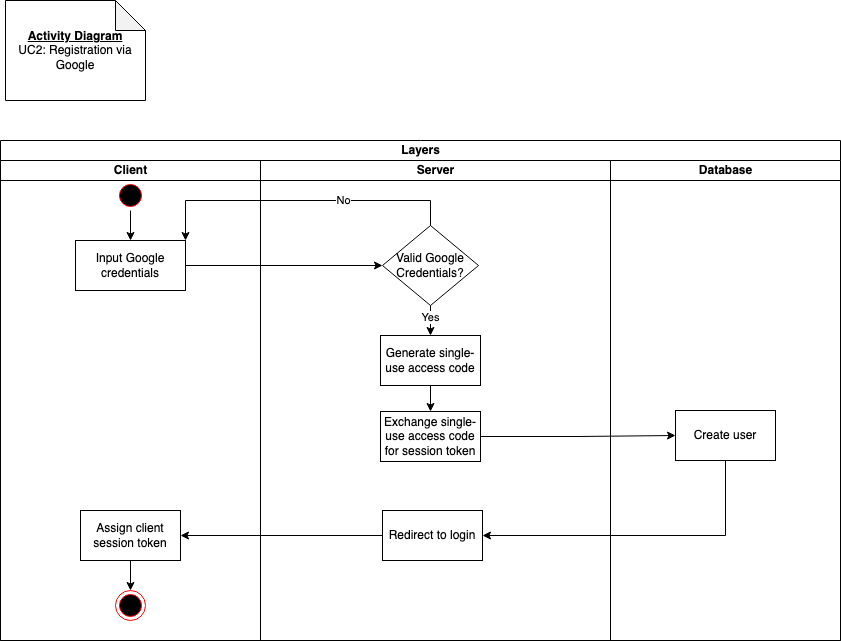
The following is a summary table of the Activity Diagrams provided followed sub-sections of the actual diagrams.

|  |  |  |
| --- | --- | --- |
| **Activity Diagram #** | **Description** | **Related**  **UCS #** |
| AD1 | Registration via Email and Password | UC1 |
| AD2 | Registration via Google Auth | UC2 |
| AD3 | User Adds Raspberry Pi | UC3 |
| AD4 | User Login via Email and Password | UC4 |
| AD5 | User Login via Google Auth | UC5 |
| AD6 | User Change Password via User Account | UC6 |
| AD7 | User Change Profile Image via User Account | UC7 |
| AD8 | User Logout via User Account | UC8 |
| AD9 | View Raspberry Pi Data | UC9 |
| AD10 | Delete User via Email and Password | UC10 |
| AD11 | Update User Profile Information | UC11 |
| AD12 | Configure Personalized Alerts | UC12 |
| AD13 | User Generates Custom Reports | UC13 |

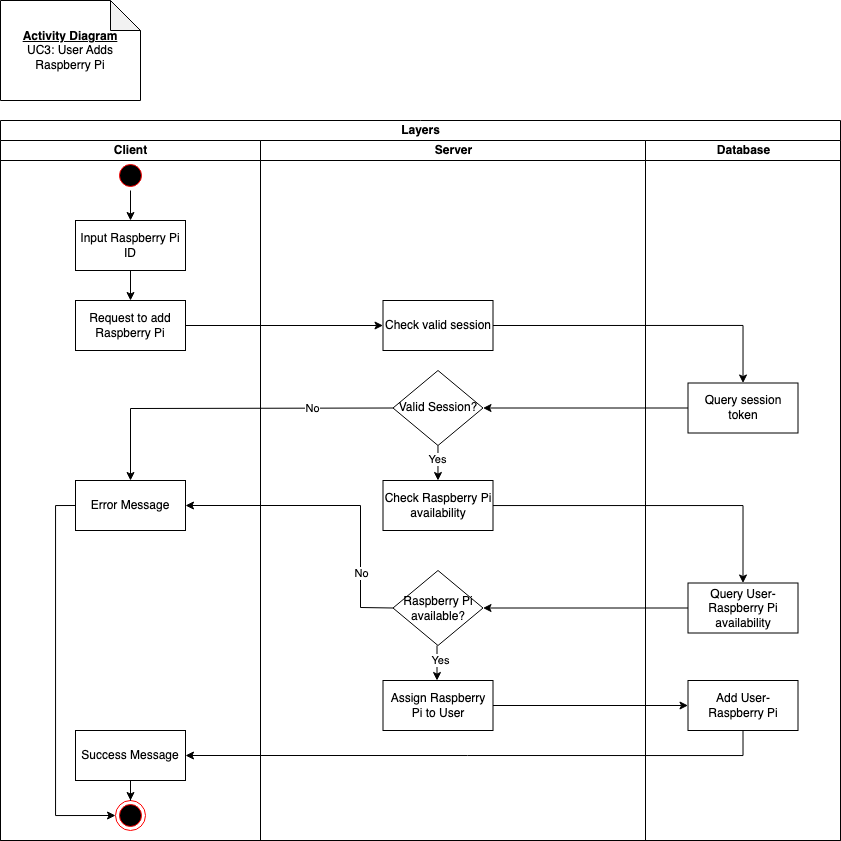
## 3.4.1 AD1 – Registration via Email and Registration



## 3.4.2 AD2 – Registration via Google Auth



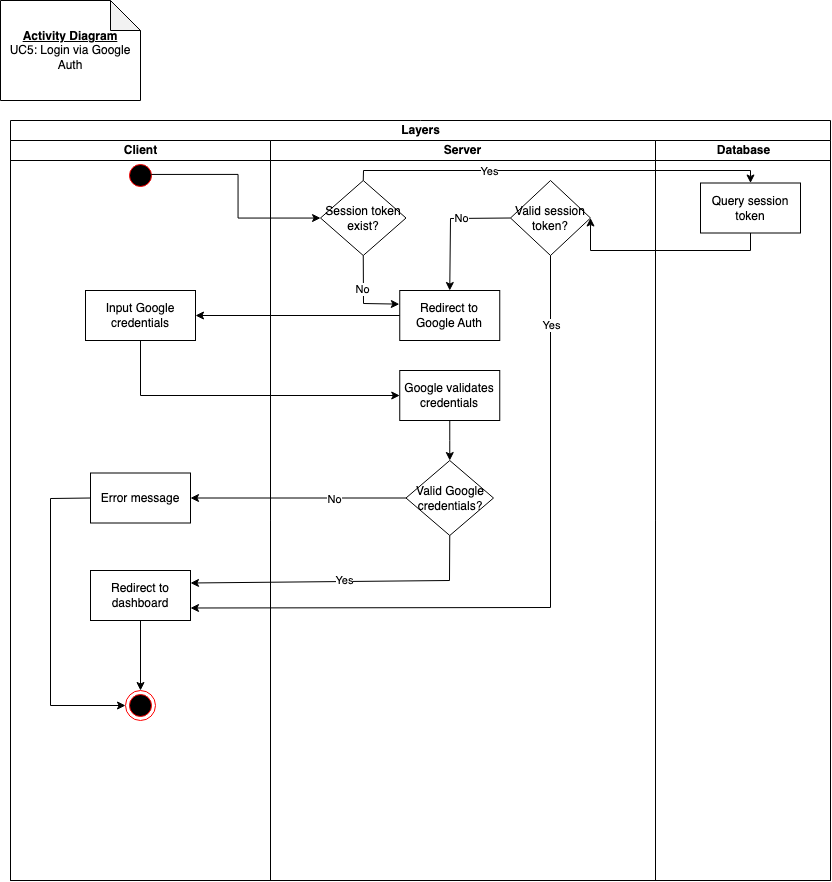
## 3.4.3 AD3 – User Adds Raspberry Pi



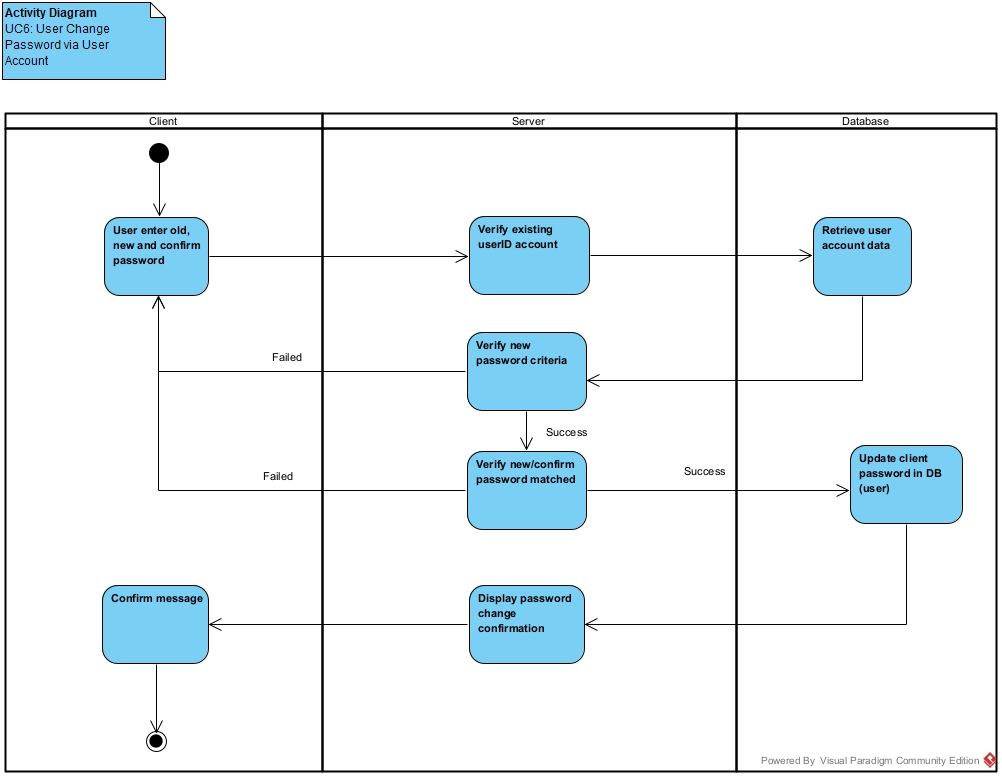
## 3.4.4 AD4 – User Login via Email and Password



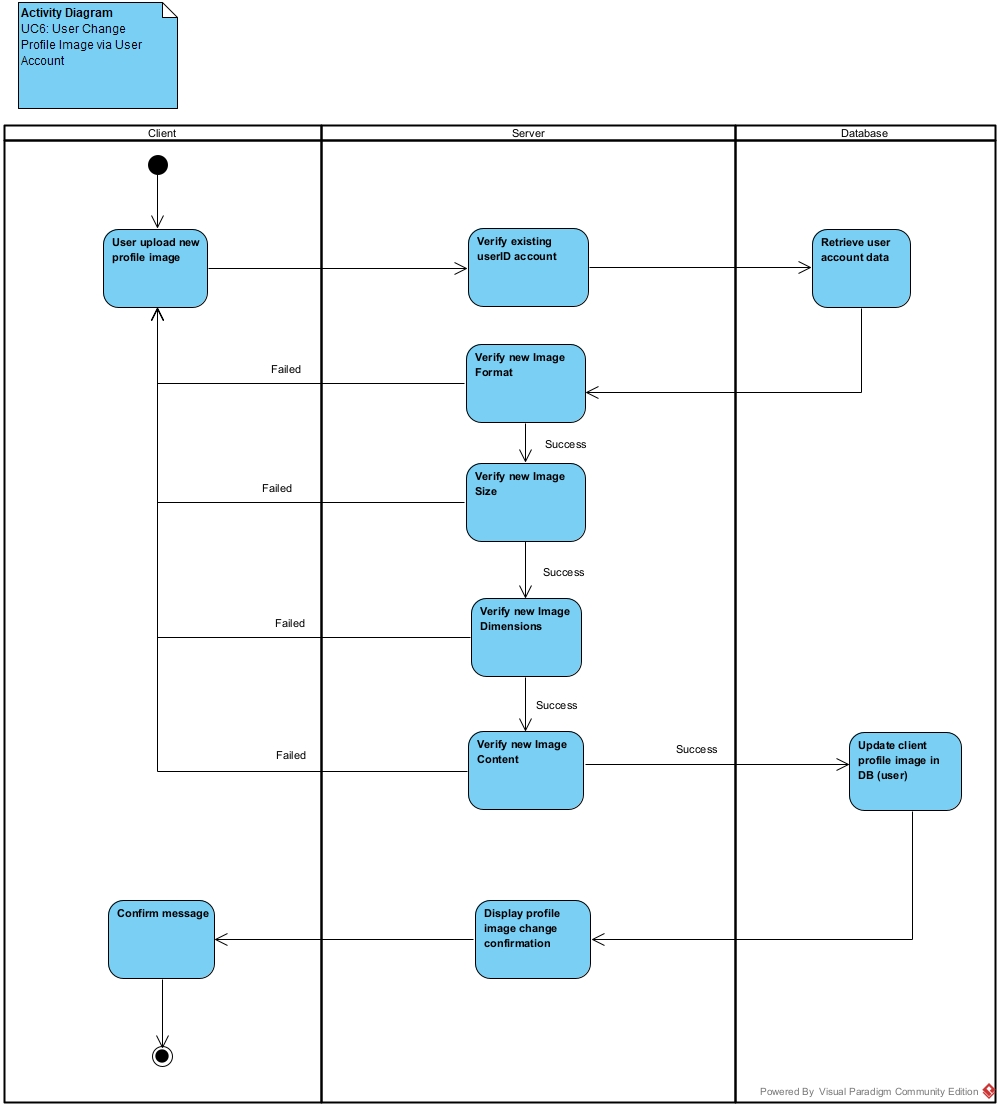
## 3.4.5 AD5 – User Login via Google Auth



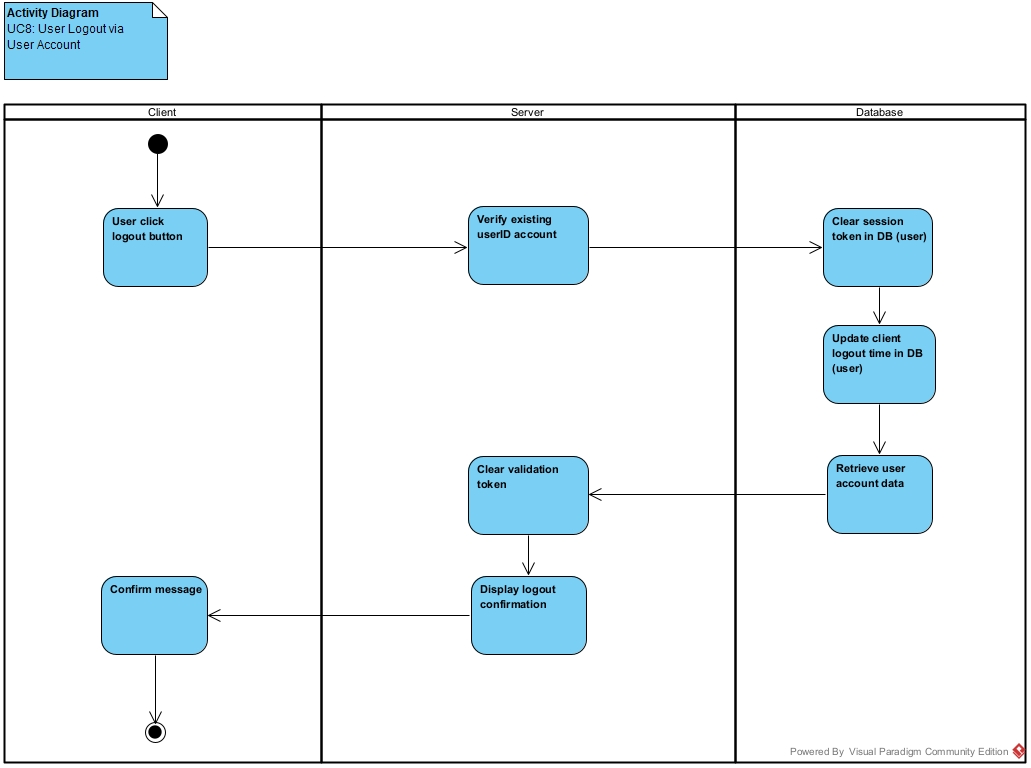
## 3.4.6 AD6 – User Change Password via User Account



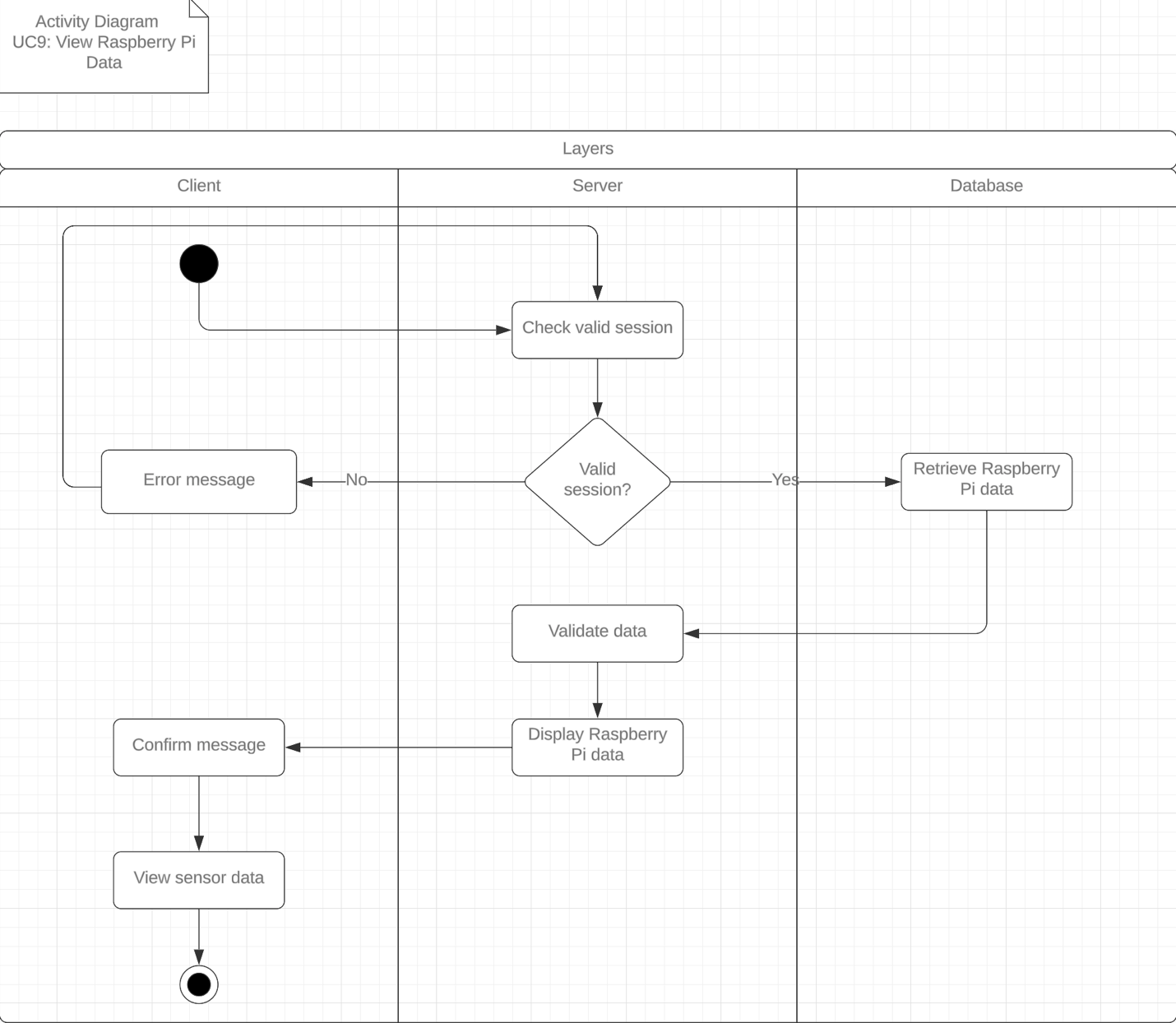
## 3.4.7 AD7 – User Change Profile Image via User Account



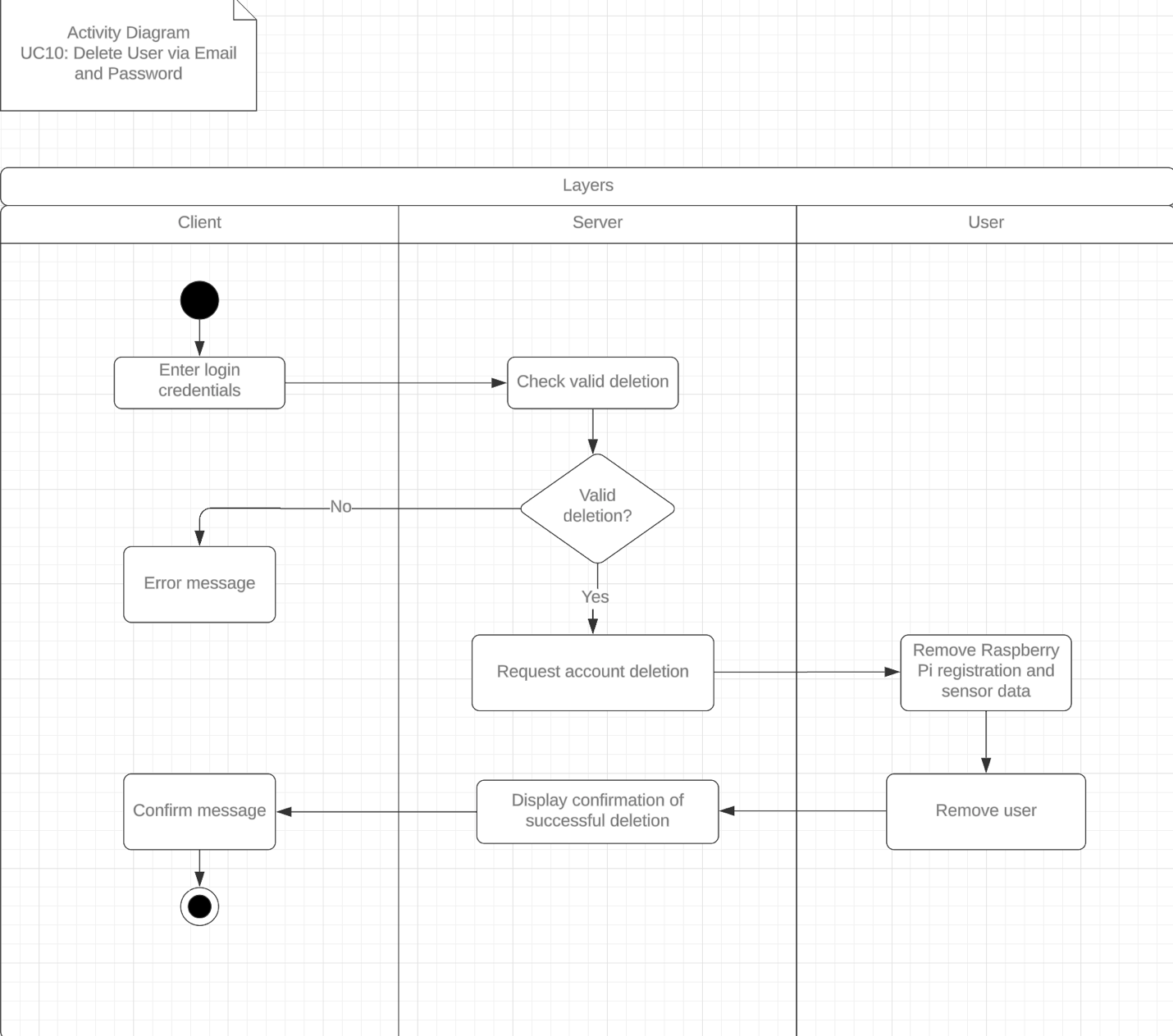
## 3.4.8 AD8 – User Logout via User Account



## 3.4.9 AD9 – View Raspberry Pi Data



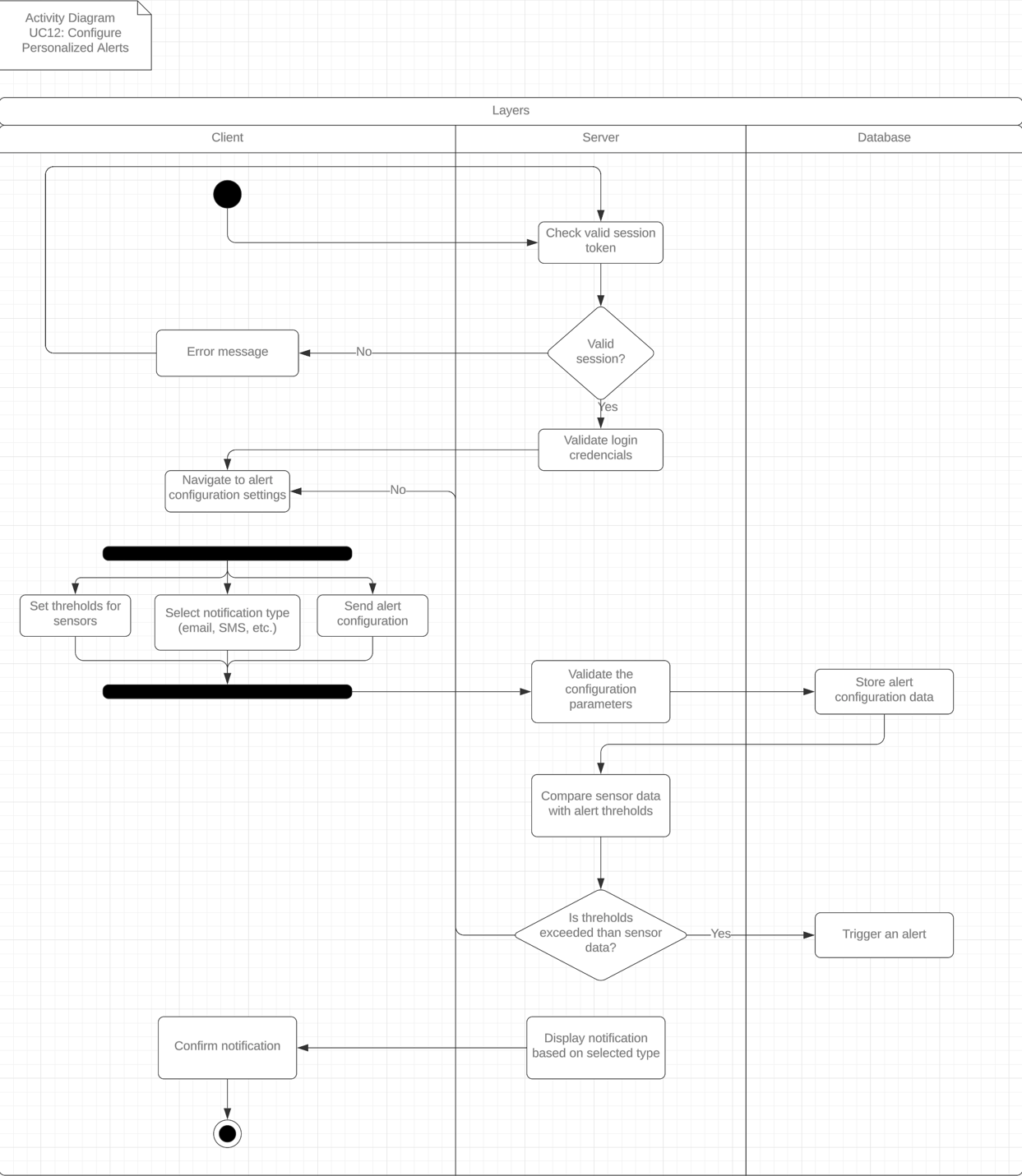
## 3.4.10 AD10 – Delete User via Email and Password



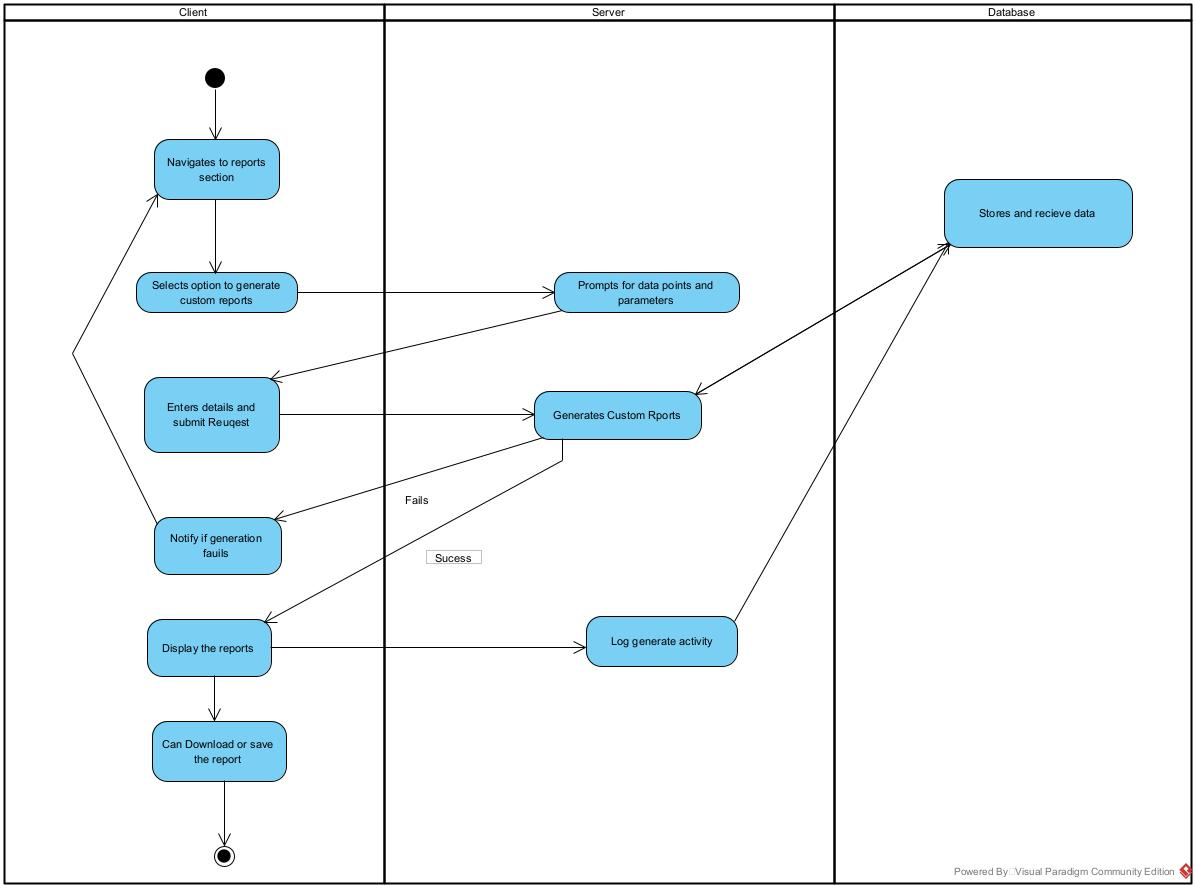
## 3.4.11 AD11 – Update User Profile Information



## 3.4.12 AD12 – Configure Personalized Alerts



## 3.4.13 AD13 – User Generates Custom Reports



# 3.5 Business Rules

The following is a list of Business Rules that must be met through the design of the <name> application. Each rule is described below and associated with the corresponding Activity Diagrams, Use Case Scenarios, and User-Interface Mock-up.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Business Rule #** | **Description** | **Activity Diagram** | **Related UCS** | **UI Mockup** |
| BR001 | Users must have a valid email address upon registration | AD1,AD2 | UC1,UC3 | Login/Registration Page |
| BR002 | Users must Register a Raspberry Pi to access their sensor data | AD3 | UC1,UC3 | User Dashboard Page |
| BR003 | User deletes their account | AD10 | UC10 | User Account Page |
| BR004 | User passwords must meet security standards to ensure data protection | AD6 | UC6 | Registration Page, Password Reset Page |
| BR005 | Users must agree to the terms and conditions before completing the registration | AD1, AD2 | UC1, UC2 | Registration Page |
| BR006 | Users must verify their identity via email when requesting a password reset | AD6 | UC6 | Password Reset Page |
| BR007 | User data must be encrypted during transmission and storage to protect privacy | AD6,  AD7,  AD8,  AD9 | UC6,  UC7,  UC8,  UC9 | Data Security Page |
| BR008 | Users must be able to add multiple sensors to their profile and view data from each sensor | AD3 | UC3 | Sensor Management Page, Dashboard Page |
| BR009 | Users must be able to set personalized alerts | AD12 | UC12 | Alerts Setting/Configuration Page |
| BR010 | Users must be able to export sensor data in multiple formats | AD13 | UC13 | Data Export Page |
| BR011 | Users must have the ability to reset their passwords through a secure process | AD6 | UC6 | Password Reset Page |
| BR012 | Administrators must be able to manage system backups and restore data if needed | AD14 | UC14 | System Backup Management Page |
| BR013 | Administrators must audit user activity regularly | AD15 | UC15 | Admin Dashboard Page, Activity Log Page |
| BR014 | Users must be able to generate custom reports from sensor data | AD13 | UC13 | Report Generation Page |
| BR015 | The system must handle and log all error messages | AD15 | UC15 | Error Handling Page |
| BR016 | Users must be able to change their profile pictures | AD7 | UC7 | Profile Picture Update Page |
| BR017 | Users must receive a confirmation email upon successful registration | AD1, AD2 | UC1, UC2 | Registration/Email Confirmation Page |
| BR018 | Administrators must be able to disable user accounts | AD14 | UC14 | Admin User Management Page |
| BR019 | The system must log all administrative actions | AD15 | UC15 | Admin Activity Log Page |
| BR020 | The system must support user roles and permissions | AD15 | UC15 | Role Management Page |
| BR021 | The system must support multilingual user interfaces | AD15 | UC15 | Language Selection/Settings Page |
| BR022 | The system must enforce data retention policies | AD15 | UC15 | Data Retention Policy Page |
| BR023 | Users must be notified of any changes to the terms and conditions | AD15 | UC15 | Notifications Page, Terms and Conditions Page |
| BR024 | Users must be able to recover their accounts via security questions | AD6 | UC6 | Account Recovery Page |
| BR025 | Confirmed password must match the new password exactly | AD6 | UC6 | User Account page |
| BR026 | User must be able to change profile image, ensuring it meets format, size, dimensions, and content criteria | AD7 | UC7 | User Account page |
| BR027 | User must be able to log out, invalidating the session token | AD8 | UC8 | Logout Button on Navigation Bar |

## BR001: Users must have a valid email address upon registration

### Rule ID: BR001

### Description:

Users must have a valid email address upon registration to access pages and features that require proper user authentication.

### Activity Diagram(s):

* 3.4.1 AD1
* 3.4.2 AD2

### Related UCS:

* 3.3.1 UC1
* 3.3.2 UC3

### UI Mockup:

* Login/Registration Page

### Purpose:

* Various pages, data, and features (e.g. personalized notifications) are specific to a user.
* Data belonging to a user should be only accessible and shared by said user.
* Notifications set by the user should be only pushed to said user.

### Conditions:

* User provided email should follow the standard email format:
  + user@example.com
* The email provided is not registered.

### Actions:

* Email is validated upon submission.
* Account registration must be verified via link sent to provided email.
* Activate user’s account only after verification link has been clicked on.

### Exceptions:

* If the user’s account is not verified within 48 hours, the registration process will expire - the user must re-register.

## BR002: Users must Register a Raspberry Pi to access their sensor data

### Rule ID: BR002

### Description:

Registered users must register a Raspberry Pi to access their sensor data, and to set and receive alerts/updates when data meets a set threshold.

### Activity Diagram(s):

* 3.4.3 AD3

### Related UCS:

* 3.3.1 UC1
* 3.3.3 UC3

### UI Mockup:

* User Dashboard Page

### Purpose:

Registering a Raspberry Pi making the data only accessible to its’ registered owner. The Raspberry Pi cannot have multiple owners, therefore cannot be registered to another user unless the previous owner has deregistered it from their account.

### Conditions:

* The Raspberry Pi must be available (i.e. not registered to another account).

### Actions:

* Check Raspberry Pi availability in the database.
* Register Raspberry Pi to user.
* Give user access to Raspberry Pi data.

### Exceptions:

* If the Raspberry Pi is not available (i.e. registered to another user), deny the registration and send an error message.

## BR003: User deletes their account

### Rule ID: BR003

### Description:

Registered users delete their account from the system.

### Activity Diagram(s):

* 3.4.3 AD10

### Related UCS:

* 3.3.10 UC10

### UI Mockup:

* User Account Page

### Purpose:

Registered user removes their personal information and data from the system. Deregistering their Raspberry Pi(s).

### Conditions:

* User must be logged into their account in order to delete their account.
* The user must confirm account deletion by entering “delete”.
* The system informs user of the consequences of deleting their account.
* The system gives option for user to download their data before deletion.

### Actions:

* User is informed of consequences of deleting their account.
* User is given an option to download their data before account deletion.
* Export user data for download.
* Validate deletion confirmation (e.g. by inputting “delete”).
* Deregister Raspberry Pi(s) from user.
* Delete all user data from system.
* Redirect user to “Home” page.

### Exceptions:

* If user does not confirm deletion, re-prompt to confirm deletion.

## BR004: User passwords must meet security standards to ensure data protection

### Rule ID: BR004

### Description:

User passwords must meet security standards to ensure data protection.

### Activity Diagram(s):

* 3.4.6 AD6

### Related UCS:

* 3.3.6 UC6

### UI Mockup:

* Registration Page, Password Reset Page

### Purpose:

To ensure that users' data is protected by strong authentication mechanisms.

### Conditions:

* Password must be at least 8 characters long.
* Must include at least one uppercase letter,
* one lowercase letter, one number,
* and one special character.

### Actions:

* Password strength is validated upon submission.

### Exceptions:

* If the password does not meet the criteria, the user is prompted to enter a new password.

## BR005: Users must agree to the terms and conditions before completing the registration

### Rule ID: BR005

### Description:

Users must agree to the terms and conditions before completing the registration.

### Activity Diagram(s):

* 3.4.1 AD1
* 3.4.2 AD2

### Related UCS:

* 3.3.1 UC1
* 3.3.2 UC2

### UI Mockup:

* Registration Page

### Purpose:

Ensure that users are aware of and agree to the legal terms and conditions of using the service.

### Conditions:

* Users must check a box indicating agreement with terms and conditions.

### Actions:

* Registration can only be completed if the terms and conditions box is checked.

### Exceptions:

* If the box is not checked, the registration cannot be completed.

## BR006: Users must verify their identity via email when requesting a password reset

### Rule ID: BR006

### Description:

Users must verify their identity via email when requesting a password reset.

### Activity Diagram(s):

* 3.4.6 AD6

### Related UCS:

* 3.3.6 UC6

### UI Mockup:

* Password Reset Page

### Purpose:

To ensure the security of password reset processes.

### Conditions:

* User must enter a valid email address.

### Actions:

* A verification link is sent to the provided email address. Password reset is enabled only after the user clicks the verification link.

### Exceptions:

* If the email address is not valid or not registered, an error message is displayed.

## BR007: User data must be encrypted during transmission and storage to protect privacy

### Rule ID: BR007

### Description:

User data must be encrypted during transmission and storage to protect privacy.

### Activity Diagram(s):

* 3.4.6 AD6
* 3.4.7 AD7
* 3.4.8 AD8
* 3.4.9 AD9

### Related UCS:

* 3.3.6 UC6
* 3.3.7 UC7
* 3.3.8 UC8
* 3.3.9 UC9

### UI Mockup:

* Data Security Page

### Purpose:

To ensure that user data is securely transmitted and stored.

### Conditions:

* Data encryption must follow industry standards.

### Actions:

* All user data is encrypted before transmission. Encrypted data is stored securely in the database.

### Exceptions:

* None specified.

## BR008: Users must be able to add multiple sensors to their profile and view data from each sensor

### Rule ID: BR008

### Description:

Users must be able to add multiple sensors to their profile and view data from each sensor.

### Activity Diagram(s):

* 3.4.3 AD3

### Related UCS:

* 3.3.3 UC3

### UI Mockup:

* Sensor Management Page, Dashboard Page

### Purpose:

To allow users to manage and monitor multiple sensors for comprehensive data analysis.

### Conditions:

* Users can add, remove, and configure sensors.
* Sensor data should be displayed in an organized manner.

### Actions:

* Provide an interface to manage sensors. Display real-time data from each sensor.

### Exceptions:

* If the sensor data is not available, display an appropriate error message.

## BR009: Users must be able to set personalized alerts

### Rule ID: BR009

### Description:

Users must be able to set personalized alerts.

### Activity Diagram(s):

* 3.4.12 AD12

### Related UCS:

* 3.3.12 UC12

### UI Mockup:

* Alerts Settings/Configuration Page

### Purpose:

To notify users when certain conditions are met, or thresholds are exceeded.

### Conditions:

* Users can set personalized alerts for conditions he need alert for.

### Actions:

* Trigger alerts when sensor data exceeds predefined thresholds. Notify users via chosen communication methods (e.g., email, SMS).

### Exceptions:

* If alert conditions are not met, ensure no false notifications are sent.

## BR010: Users must be able to export sensor data in multiple formats

### Rule ID: BR010

### Description:

Users must be able to export sensor data in multiple formats.

### Activity Diagram(s):

* 3.4.13 AD13

### Related UCS:

* 3.3.13 UC13

### UI Mockup:

* Data Export Page

### Purpose:

To provide flexibility in how users handle and analyze their data.

### Conditions:

* Support export in CSV and JSON formats.

### Actions:

* Provide options for data export. Ensure exported data is accurate and complete.

### Exceptions:

* If export fails, display an error message and log the incident for troubleshooting.

## BR011: Users must have the ability to reset their passwords through a secure process

### Rule ID: BR011

### Description:

Users must have the ability to reset their passwords through a secure process.

### Activity Diagram(s):

* 3.4.6 AD6

### Related UCS:

* 3.3.6 UC6

### UI Mockup:

* Password Reset Page

### Purpose:

To ensure users can regain access to their accounts securely if they forget their passwords.

### Conditions:

* User must provide a valid email address.
* Password reset link should expire after a certain period.

### Actions:

* Send a password reset link to the user’s email. Allow the user to set a new password upon clicking the link.

### Exceptions:

* If the email is not registered, display an error message.

## BR012: Administrators must be able to manage system backups and restore data if needed

### Rule ID: BR012

### Description:

Administrators must be able to manage system backups and restore data if needed.

### Activity Diagram(s):

* 3.4.14 AD14

### Related UCS:

* 3.3.14 UC14

### UI Mockup:

* System Backup Management Page

### Purpose:

To ensure data integrity and availability through regular backups.

### Conditions:

* Regular automatic backups should be scheduled.

### Actions:

* Provide options to backup and restore data. Notify administrators of backup status.

### Exceptions:

* If a backup fails, an error message should be logged, and an alert sent to administrators.

## BR014: Users must be able to generate custom reports from sensor data

### Rule ID: BR014

### Description:

Users must be able to generate custom reports from sensor data.

### Activity Diagram(s):

* 3.4.13 AD13

### Related UCS:

* 3.3.13 UC13

### UI Mockup:

* Report Generation Page

### Purpose:

To provide users with the ability to analyze and present sensor data in a format that suits their needs.

### Conditions:

* Support multiple formats for reports, including PDF and Excel.
* Users can select specific data ranges and sensors for the report.

### Actions:

* Provide options for generating and downloading reports in selected formats.

### Exceptions:

* If report generation fails, display an error message and log the incident.

## BR015: The system must handle and log all error messages

### Rule ID: BR015

### Description:

The system must handle and log all error messages.

### Activity Diagram(s):

* 3.4.15 AD15

### Related UCS:

* 3.3.15 UC15

### UI Mockup:

* Error Handling Page

### Purpose:

To ensure that any issues encountered are logged and can be addressed by the support team.

### Conditions:

* Error messages must be clear and provide guidance on how to proceed.
* All errors must be logged with details including timestamp, error type, and affected user.

### Actions:

* Display error messages to users and log errors for further analysis.

### Exceptions:

* None specified.

## BR016: Users must be able to change their profile pictures

### Rule ID: BR016

### Description:

Users must be able to change their profile pictures.

### Activity Diagram(s):

* 3.4.7 AD7

### Related UCS:

* 3.3.7 UC7

### UI Mockup:

* Profile Picture Update Page

### Purpose:

To allow users to personalize their profiles.

### Conditions:

* Users can upload and change their profile pictures.
* Profile pictures must meet size and format requirements.

### Actions:

* Provide an interface for uploading and updating profile pictures.

### Exceptions:

* If the uploaded picture does not meet the requirements, display an error message.

## BR017: Users must receive a confirmation email upon successful registration

### Rule ID: BR017

### Description:

Users must receive a confirmation email upon successful registration.

### Activity Diagram(s):

* 3.4.1 AD1
* 3.4.2 AD2

### Related UCS:

* 3.3.1 UC1,
* 3.4.2 UC2

### UI Mockup:

* Registration/Email Confirmation Page

### Purpose:

To confirm that the user has successfully registered and to provide important account information.

### Conditions:

* A confirmation email must be sent to the user’s registered email address.
* The email should include a confirmation link and account details.

### Actions:

* Send a confirmation email upon successful registration.

### Exceptions:

* If the email fails to send, log the incident and notify the user to try again.

## BR020: The system must support user roles and permissions

### Rule ID: BR020

### Description:

The system must support user roles and permissions.

### Activity Diagram(s):

* 3.4.15 AD15

### Related UCS:

* 3.3.15 UC15

### UI Mockup:

* Role Management Page

### Purpose:

To manage access control and ensure that users have the appropriate permissions for their roles.

### Conditions:

* Users can be assigned roles with specific permissions.
* Only administrators can manage roles and permissions.

### Actions:

* Provide an interface for assigning roles and managing permissions.

### Exceptions:

* If role assignment fails, display an error message and log the incident.

## BR021: The system must support multilingual user interfaces

### Rule ID: BR021

### Description:

The system must support multilingual user interfaces.

### Activity Diagram(s):

* 3.4.15 AD15

### Related UCS:

* 3.3.15 UC15

### UI Mockup:

* Language Selection/Settings Page

### Purpose:

To provide accessibility and usability for users who speak different languages.

### Conditions:

* Users can select their preferred language for the interface.
* The system must support at least English, Spanish, and French.

### Actions:

* Provide an interface for language selection and ensure all UI elements are translated.

### Exceptions:

* If translation is not available, default to English and notify the user.

## BR022: The system must enforce data retention policies

### Rule ID: BR022

### Description:

The system must enforce data retention policies.

### Activity Diagram(s):

* 3.4.15 AD15

### Related UCS:

* 3.3.15 UC15

### UI Mockup:

* Data Retention Policy Page

### Purpose:

To ensure compliance with legal and regulatory requirements for data storage and retention.

### Conditions:

* Data must be retained for a minimum of 5 years.
* Data older than 5 years must be archived or deleted according to the policy.

### Actions:

* Implement automated processes to archive or delete data based on the retention policy.

### Exceptions:

* None specified.

## BR023: Users must be notified of any changes to the terms and conditions

### Rule ID: BR023

### Description:

Users must be notified of any changes to the terms and conditions.

### Activity Diagram(s):

* 3.4.15 AD15

### Related UCS:

* 3.3.15 UC15

### UI Mockup:

* Notifications Page, Terms and Conditions Page

### Purpose:

To ensure users are aware of and agree to any updates to the terms and conditions.

### Conditions:

* Users must be notified via email of any changes to the terms and conditions.
* Users must accept the new terms and conditions upon their next login.

### Actions:

* Send notification emails and require users to accept the updated terms and conditions.

### Exceptions:

* If the user does not accept the new terms and conditions, restrict access to the account until acceptance.

## BR024: Users must be able to recover their accounts via security questions

### Rule ID: BR024

### Description:

Users must be able to recover their accounts via security questions.

### Activity Diagram(s):

* 3.4.6 AD6

### Related UCS:

* 3.3.6 UC6

### UI Mockup:

* Account Recovery Page

### Purpose:

To provide an additional method for users to recover their accounts if they forget their passwords.

### Conditions:

* Users must set up security questions during account creation.
* Users can recover their accounts by answering their security questions correctly.

### Actions:

* Provide an interface for setting up and answering security questions for account recovery.

### Exceptions:

* If the user cannot answer the security questions, prompt them to use an alternative recovery method.

## BR025: Confirmed password must match the new password exactly

### Rule ID: BR025

### Description:

The confirmed password must match the new password exactly to ensure that the user has correctly entered their desired password and prevent any potential errors during the password change or registration process.

### Activity Diagram(s):

* 3.3.6 UC6

### Related UCS:

* 3.3.6 UC6

### UI Mockup:

* User Account Page

### Purpose:

* To ensure accuracy and consistency in the password change or registration process by verifying that the user has entered the same password in both fields, reducing the likelihood of user errors and enhancing account security.
* User confirm the new password through User Account page

### Conditions:

* The confirmed password input field must exactly match the new password input field.

### Actions:

* Both the new password and confirmed password fields are compared upon submission.
* If the passwords do not match, the user is prompted to re-enter the passwords.

### Exceptions:

* If the confirmed password does not match the new password, the password change or registration process will be halted until the passwords match.

## BR026: User must be able to change profile image, ensuring it meets format, size, dimensions, and content criteria

### Rule ID: BR026

### Description:

Users must be able to change their profile image, ensuring it meets specific format, size, dimensions, and content criteria to maintain consistency and appropriateness of user profiles.

### Activity Diagram(s):

* 3.3.7 UC7

### Related UCS:

* 3.3.7 UC7

### UI Mockup:

* User Account Page

### Purpose:

* To allow users to personalize their profile while ensuring that the profile images are suitable and uniform in terms of format, size, dimensions, and content. This helps maintain the professional and appropriate appearance of user profiles.
* User upload the profile image through User Account Page

### Conditions:

* The profile image must be in an accepted format (JPEG, PNG).
* The image size must not exceed a specified limit (2MB).
* The image dimensions should meet the required specifications (min:100x100 pixel, max:1000x1000 pixel).
* The content of the image must be appropriate and comply with community guidelines.

### Actions:

* User uploads a new profile image.
* The system validates the image format, size, dimensions, and content.
* If the image meets all criteria, it is saved and updated as the user's profile image.
* If the image does not meet any criteria, the user is prompted to upload a different image that complies with the requirements.

### Exceptions:

* If the profile image does not meet the specified criteria, the upload process will be declined, and the user must provide an image that adheres to the guidelines.

## BR027: User must be able to log out, invalidating the session token

### Rule ID: BR027

### Description:

Users must be able to log out of the system, which will clear the session token and validation token to ensure that the user's session is securely terminated and cannot be reused by unauthorized parties.

### Activity Diagram(s):

* 3.3.8 UC8

### Related UCS:

* 3.3.8 UC8

### UI Mockup:

* Logout Button on Navigation Bar

### Purpose:

* To provide users with a secure way to end their session, ensuring that their session token and validation token are both clear to prevent any unauthorized access or reuse of the session.

### Conditions:

* The user must be logged in to initiate the logout process.
* The user clicks the “logout” button.

### Actions:

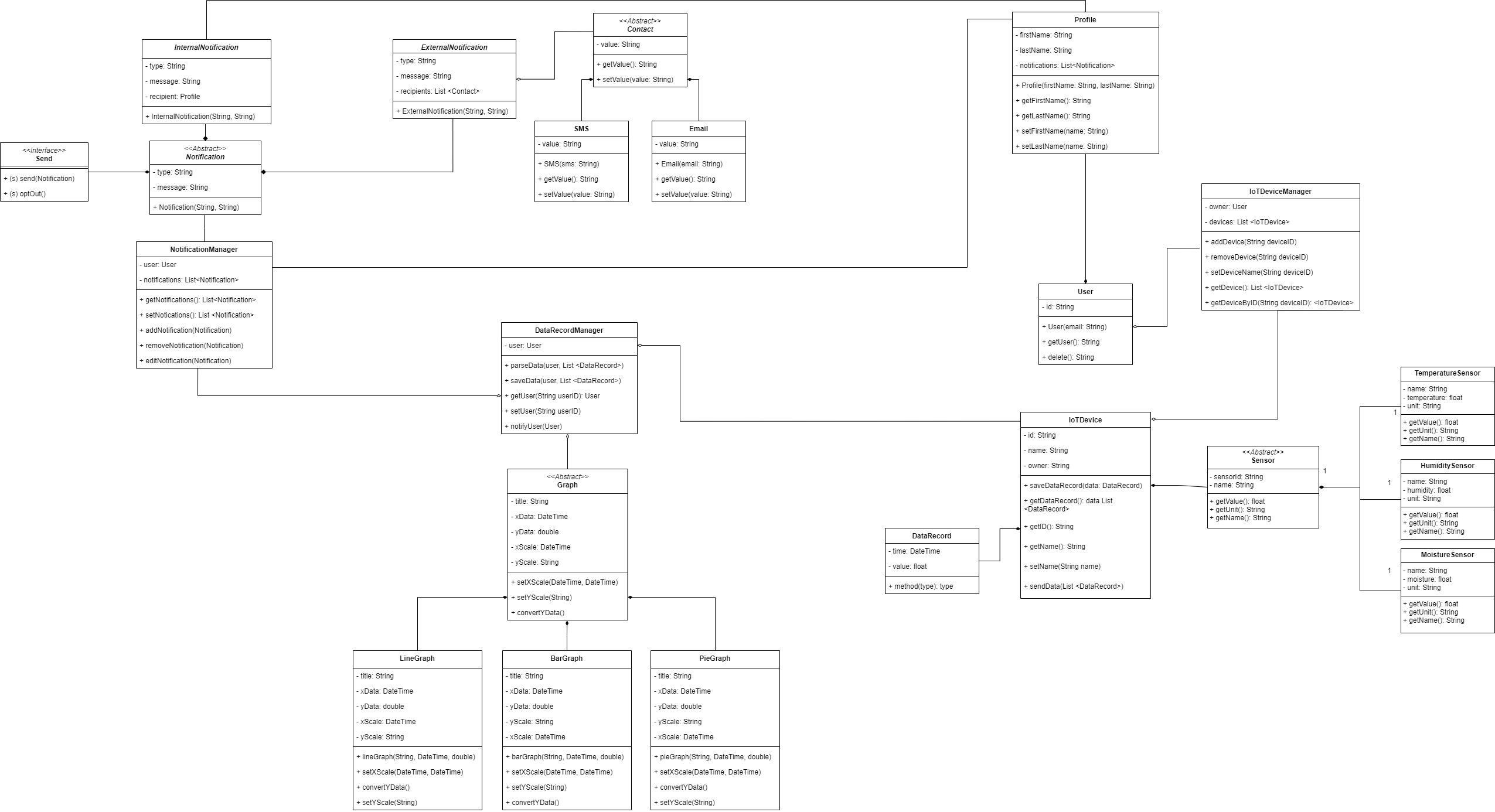
* User clicks the logout button.
* The system clears the session token associated with the user's session in Database.
* The system clears the validation token in website
* The user is redirected to the login page or homepage.
* A confirmation message is displayed to the user indicating successful logout.

### Exceptions:

* If the session token or validation token cannot be cleared due to a system error, the user is notified and prompted to try again.

Section 4 – Domain Class

Class Diagram Link: <https://app.diagrams.net/#G1hR7FevqOmEgeuQP0Rrf44yyuYv_HKvLu#%7B%22pageId%22%3A%22SoRPeFyQ3LD7muAqzU7t%22%7D>



Section 5 – Database

Database Schema:

# PRJ566 NoSQL DB Schema

{

**"collection":"Users",**

"document":{

"\_id":"ObjectID",

"id":"String",  *// id = email, email is unique*

"password":"String"

}

}

{

**"collection":"Profiles",**

"document":{

"\_id":"ObjectID",

"firstName":"String",

"lastName":"String",

“user”: “ObjectID”,

"Notifications":[

"ObjectID"

]

}

}

{

**"collection":"Notifications",**

"document":{

"\_id":"ObjectID",

"message":"String", *// e.g. Success, warning, and error message with colors*

"type":"String"

}

}

{

**"collection":"IoTDevices",**

"document":{

"\_id":"ObjectID",

"id":"String", *// unique id of an IOT device(e.g. 2342348483)*

"name":"String",

"ownerID":"ObjectID", *// device can belong to one owner*

“sensors”: [ “ObjectID” ]

}

}

{

**"collection":"DataRecords",**

"document":{

"\_id":"ObjectID",

“time”: “DateTime”,

“value”: “Float”,

“sensor”: "ObjectID",

“IoTDeviceID”: ObjectID *// data belongs to one IoT Device, with userID*

}

}

{

**"collection":"Sensor",**

"document":{

"\_id":"ObjectID",

"sensorId":"String",  *// unique id of a Sensor device*

"name":"String"

}

}

{

**"collection":"Contacts",**

"document":{

"\_id":"ObjectID",

“name”: String,

"value":"String",

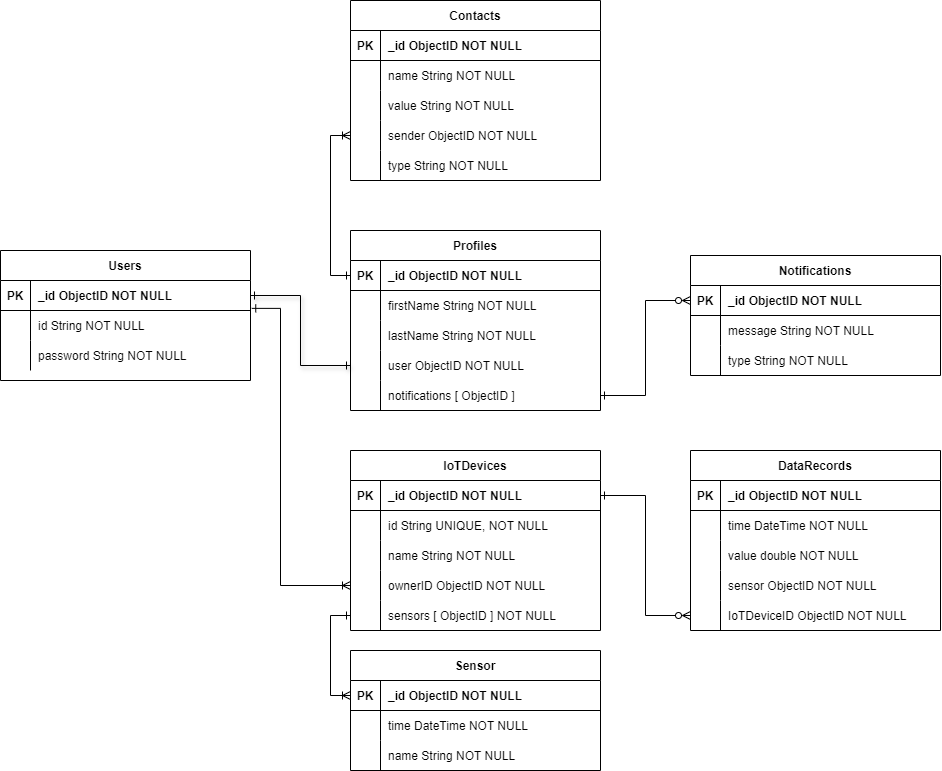
“sender”: "ObjectID", *// contact belongs to the sender*

“type”: "String"

}

}

Entity Relationship Diagram:



Section 6 – Project Management

# 6.1 Work Breakdown Structure

# 6.2 Milestones

# Milestone 1 (Week 1):

## Review SRS documentation:

### Acceptance Criteria:

* Setup local development environments
* Review team agreements
* Group sign-off on revised team agreement
* Review software architecture

**Hardware Set-up:**

Acceptance Criteria:

* Test each of the sensor's functionality (i.e. not damaged)
* Raspberry Pi can collect information from each sensor.
* Basic communication dataflow between Sensors and Raspberry Pi (userID, Raspberry Pi ID, list of data)

# Milestone 2 (Week 2 - Week 4):

**Raspberry Pi:**

Acceptance Criteria:

* Sensor data can be stored on SD card connected to Raspberry Pi
* Sensor data can be transferred from Raspberry Pi to computer locally.
* Implement error handling (logging and handling edge cases [Ex. Power off RP])
* Pass all unit tests

**MongoDB:**

Acceptance Criteria:

* Establish connection between API and local database
* Create database schemas based on project specifications

**API:**

Acceptance Criteria:

* Create logging system for API
* Setup environment types
* Create POST request to receive data from Raspberry Pi
* Received unparsed data from Raspberry Pi
* Parse data to be able to store into MongoDB Schema
* Handle user registration
  + Full implementation of NextAuth email strategy
  + Full implementation of NextAuth Google strategy
* Handle user login
* Manage User-Raspberry Pi connection(s)
* Pass all unit tests

# Milestone 3 (Week 5 - Week 8):

**MongoDB:**

Acceptance Criteria:

* Basic CRUD operations are tested and function correctly with MongoDB.

**API:**

Acceptance Criteria:

* Handle user contacts (i.e. CRUD operations)
* Send user in-app alerts
* Send user external app alerts (e.g. email/SMS notifications)
* Manage user alerts, both internal and external alerts (i.e. CRUD operations, Setup threshold condition, mobile setting)
* Retrieve user’s sensor data
* Export user’s data
* Pass all unit tests

**Client Setup:**

Acceptance Criteria:

* Setup Next.js (Vercel and Authentication)
* Setup Node.js and dependencies (etc. Express, Cors)
* Navigation setup (Ex. Home Page, Login Page, Dashboard)
* Basic API endpoints for retrieving data
* Pass all unit tests
* Sensor data can be stored on SD card connected to Raspberry Pi
* Sensor data can be transferred from Raspberry Pi to computer locally.
* Implement error handling (logging and handling edge cases [Ex. Power off RP])
* Pass all unit tests

# Milestone 4 (Week 9 - Week 11):

**End-to-End Testing:**

Acceptance Criteria:

* All components (Sensor, Raspberry Pi, MongoDB, web server, website) works together
* Reflect user use cases
* Pass all integration tests
* Pass all functionality tests
* Pass all end-to-end tests locally

**Deployment and Monitoring:**

Acceptance Criteria:

* Configure environment variables
* Montior system health and data accuracy
* Documentation and user guides are complete and available
* Deploy API on Vercel
* Deploy client to Vercel
* Deploy database to Mongo Atlas
* Configure email critical alerts on Vercel

# Milestone 5(Final Week)

Prepare Presentation Slides

Create Demo Script

Rehearse Presentation

Gather Feedback from Advisors/Peers

Revise Presentation Based on Feedback

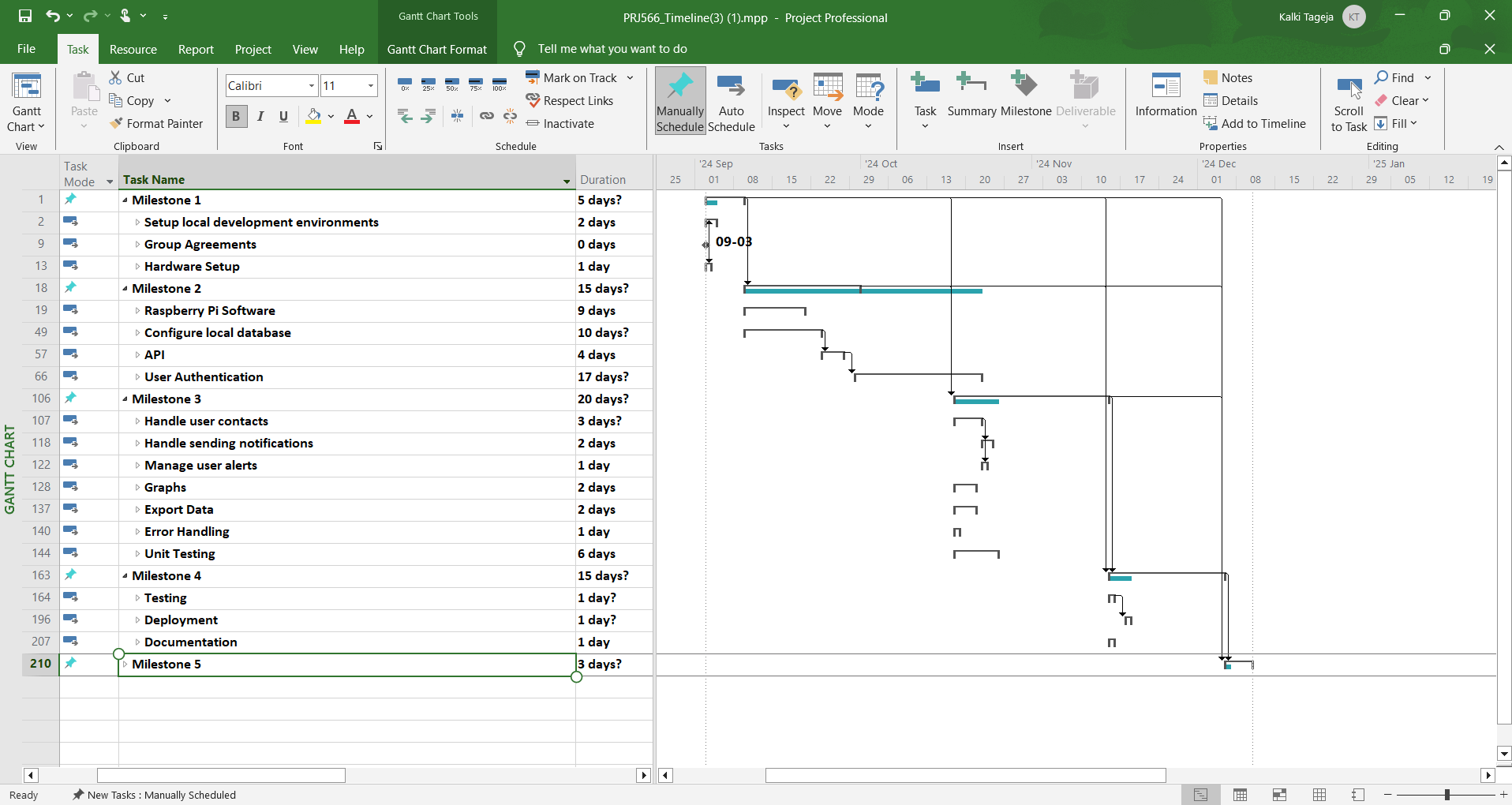
Finalize Presentation Slides and Demo

Present Project to Class/Advisors

# 6.3 Acceptance Criteria

* Setup local development environments
* Review team agreements
* Group sign-off on revised team agreement
* Review software architecture
* Test each of the sensor's functionality (i.e. not damaged)
* Raspberry Pi can collect information from each sensor.
* Basic communication dataflow between Sensors and Raspberry Pi (userID, Raspberry Pi ID, list of data)
* Establish connection between API and local database
* Create database schemas based on project specifications
* Create logging system for API
* Setup environment types
* Create POST request to receive data from Raspberry Pi
* Received unparsed data from Raspberry Pi
* Parse data to be able to store into MongoDB Schema
* Handle user registration
  + Full implementation of NextAuth email strategy
  + Full implementation of NextAuth Google strategy
* Handle user login
* Manage User-Raspberry Pi connection(s)
* Pass all unit tests
* All components (Sensor, Raspberry Pi, MongoDB, web server, website) works together
* Reflect user use cases
* Pass all integration tests
* Pass all functionality tests
* Pass all end-to-end tests locally
* Configure environment variables
* Montior system health and data accuracy
* Documentation and user guides are complete and available
* Deploy API on Vercel
* Deploy client to Vercel
* Deploy database to Mongo Atlas
* Configure email critical alerts on Vercel
* Slides cover all key aspects of the project, are clear, and visually engaging.
* Script is comprehensive, easy to follow, and covers all critical functionalities of the project.
* Team members are familiar with the presentation flow and can present confidently.
* Feedback is collected from at least three advisors or peers.
* Presentation is updated to address all major feedback points.
* Final slides and demo script are polished, error-free, and ready for presentation.
* Presentation is delivered smoothly, within the allocated time, and all questions are answered effectively.

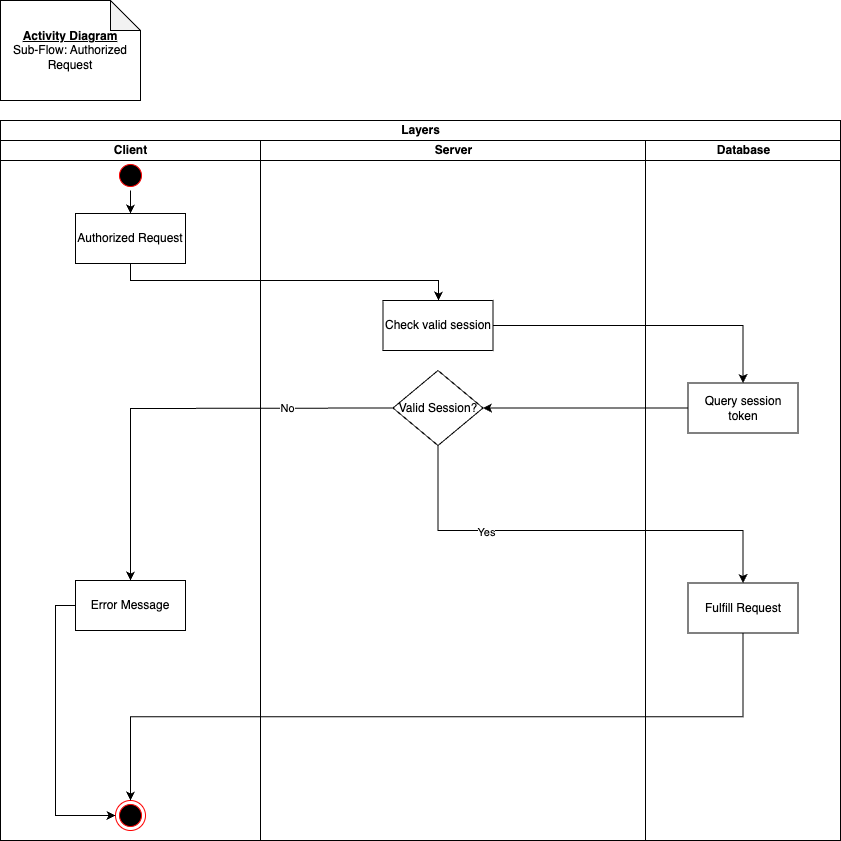
# 6.4 Implementation Schedule



Section 7 – Client/Faculty Sign-off

# Appendix

## Appendix A: Sub-Flow Diagram

Figure 1: Sub-Flow, Authorized Request