Real-Time Coir Production Monitoring System - CoiioT

JEEVAN N

Register No: 7176 22 31 023

DISSERTATION SUBMITTED IN PARTIAL FULFILMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

M.Sc. (Software Systems)

OF ANNA UNIVERSITY

November 2025

DEPARTMENT OF COMPUTING

COIMBATORE INSTITUTE OF TECHNOLOGY

(Autonomous Institution affiliated to Anna University)

COIMBATORE – 641014

COIMBATORE INSTITUTE OF TECHNOLOGY

(Autonomous Institution affiliated to Anna University)

COIMBATORE 641014

(Bonafide Certificate)

Project Work - I
Seventh Semester

Real-Time Coir Production Monitoring System - CoiioT

Bonafide record of work done by
JEEVAN N
(Register No: 7176 22 31 023)

Submitted in partial fulfilment of the requirements for the degree of
M.Sc. (Software Systems) of Anna University

November 2025

Faculty Guide	Head of the Department

Submitted for the viva-voce held on ________________

Internal Examiner	External Examiner

CONTENTS

ACKNOWLEDGEMENT

The Success is achieved through mentorship, guidance, and the support of many people. I take this opportunity to express my sincere gratitude to all those who have provided invaluable encouragement and direction throughout this project. Their contributions have been instrumental in its successful completion.

I respect and thank **Dr. A. RAJESWARI**, **Principal,** Coimbatore Institute of Technology, Coimbatore, for providing me with the opportunity to undertake this full-time industry project.

I thank **Dr. K. SAKTHI MALA**, **Dean, Department of Computing,** Coimbatore Institute of Technology, Coimbatore, for her motivation, constant support, and guidance throughout the project work, inspiring me to achieve higher.

I thank **Dr. S. MANJULA GANDHI, Head and Associate Professor, Department of Computing (Software Systems),** Coimbatore Institute of Technology, Coimbatore, for her motivation, constant support, and guidance throughout the project work, inspiring me to achieve higher.

I am indebted to my internal guide, **Dr. T.N. SUGUMAR**, **Assistant Professor, Department of Computing (Software Systems)**, Coimbatore Institute of Technology, Coimbatore, for his constant support and guidance throughout the project.

Additionally, I am sincerely thankful to my external guide, **Mr. B. JAGADEESAN**, **Product Development Lead, IMIK Technologies**, Coimbatore, for his invaluable assistance and insights across various aspects of this project.

Finally, I would like to acknowledge the faculty and staff of Department of Computing, Coimbatore Institute of Technology for their encouragement and support throughout the project.

SYNOPSIS

The CoiioT system aims to digitally transform operations in coir factories by automating production monitoring, resource tracking, and reporting using IoT-enabled machines. The system enhances visibility into factory activities, improves decision-making through real-time insights, and minimizes manual intervention by collecting and analyzing production data automatically.

Traditional coir factories depend on manual methods to record production and resource usage, which often cause delays, errors, and inefficiencies. Without real-time information, factory owners find it difficult to identify machine downtime, energy wastage, or low productivity, leading to poor operational control.

CoiioT addresses these challenges by deploying IoT devices on key factory machines, including defibre machines, baling machines, pith block machines, and water meters. These devices automatically collect data every few seconds and transmit it to the cloud using the MQTT communication protocol. The raw data is then processed and securely stored through automated scripts and scheduled tasks.

The system provides interactive dashboards and analytics for visualizing production performance, resource consumption, inventory tracking, and historical trends. It also supports administrative functions such as attendance tracking, expense management, and report generation. Additionally, CoiioT integrates with tomorrow.io for weather forecasting to support production planning, uses Twilio for WhatsApp-based OTP authentication and employs Firebase Cloud Messaging for push notifications.

CoiioT employs Flutter for the frontend, Laravel for the backend and MySQL for data management. The system is hosted on a cloud platform to ensure high performance and scalability, and the mobile app is available on both the Google Play Store (Android) and App Store (iOS) for easy access by users.

PREFACE

Chapter I, provides an overview of the organization where the project was completed. It introduces the CoiioT project, describing the system that was developed along with the environment in which it was built and deployed.

Chapter II, explores the problem in detail through system analysis, defining requirements, specifications, and the proposed system's features. It also describes the functional and non-functional requirements of the system.

Chapter III, explains the design of the system that developed with related diagrams which outlines the different phases of the system development and elucidates the user interface flow.

Chapter IV, details the testing phase, showcasing various test cases along with their respective outcomes.

Chapter V, describes the implementation phase, listing the necessary pre-installed software and providing a walkthrough of key validations performed to ensure system functionality.

Chapter VI, summarizes the unique features of the CoiioT project and offers suggestions for future enhancements.

CHAPTER I

INTRODUCTION

The Introduction provides a comprehensive overview of the organization for which the application has been developed. It discusses the existing system, the challenges associated with it and outlines the goals and scope of the proposed application. Additionally, it specifies the system environment used for the development of the application and briefly introduces the various technologies integrated into the solution.

1.1 ORGANIZATION PROFILE

IMIK Technologies, established in 2007, is a technology-driven company specializing in embedded systems, real-time software, process control, and industrial automation. Formed by professionals with strong technical backgrounds, the company focuses on developing reliable and cost-effective customized embedded products for industries such as automation, robotics, agriculture, automotive, textile, consumer electronics, and defense.

With a focus on innovation, IMIK has also expanded into 3D printing solutions for rapid prototyping. The company has made notable contributions to agricultural automation through products like dry run preventers and automatic mobile controllers. Its flagship product, iMOS Starter, combines IoT and embedded systems to help farmers remotely control irrigation motors via mobile devices, making their work more efficient and convenient.

1.2 PROBLEM DEFINITION

Traditional coir factories face significant challenges in monitoring production activities, tracking resource usage, and maintaining accurate operational records. The existing manual methods of data collection and reporting often result in delays, inaccuracies, and a lack of real-time visibility into factory operations. Without an automated system, factory owners experience difficulties in identifying production inefficiencies, detecting resource wastage, and making timely decisions. The CoiioT system aims to overcome these challenges by automating data collection through IoT-enabled machines, ensuring accuracy, transparency and improved decision making.

1.2.1 PROBLEM OBJECTIVE

The objective of CoiioT is to digitally transform coir factory operations by automating production monitoring, data recording, and reporting through IoT integration. The system utilizes data collected from IoT devices to provide real-time insights into production performance and resource usage. It eliminates errors from manual data entry, and supports data-driven decision-making via interactive dashboards and analytics. It also delivers timely production summary notifications to help factory owners enhance productivity and reduce downtime.

1.2.2 SCOPE

The CoiioT system enables users to monitor production and resource usage in real time, make data-driven decisions, and optimize operational efficiency. It allows tracking of machine performance, identification of downtime or inefficiencies, and effective production planning using historical insights and analytics. The system also supports seamless management of attendance, expenses, and inventory, while delivering timely notifications to specific users according to their roles. Overall, CoiioT enhances informed decision-making, boosts productivity, and provides greater control over factory operations.

1.2.3 USERS

1. **Admin / Factory Owner:** Responsible for monitoring production activities, managing settings, and overseeing overall factory operations.
2. **Staff:** Responsible for recording actual production details, managing daily operations, and maintaining attendance and expense records.
3. **SuperAdmin:** Responsible for maintaining the CoIIoT system, managing factories and IoT devices, and ensuring smooth system operation, maintenance, and updates.

1.2.4 LIMITATIONS

CoiioT operates within certain constraints. Its production and resource monitoring rely on IoT sensors and predefined settings, which may not account for unexpected machine behavior or unusual operational patterns. While the system handles real-time data collection and reporting efficiently, sudden spikes in production or device malfunctions may require manual verification.

1.3 SYSTEM ENVIRONMENT

The system environment for the CoiioT project includes both hardware and software specifications essential for its development and deployment. This section outlines the key components, tools, and technologies utilized throughout the project lifecycle.

1.3.1 HARDWARE SPECIFICATION

Table 1.1 shows the specifications of the hardware, including the hardware additives used when constructing the application.

Table 1.1 Hardware Specifications

Processor	Intel Core i7
System Type	64-bit Operating System
RAM	16 GB
Hard Disk Drive	512 GB

1.3.2 SOFTWARE SPECIFICATION

Table 1.2 presents the specifications of the software tools and frameworks used to develop the application.

Table 1.2 Software Specifications

Operating System	Windows 11 Version 24H2
Frontend	Flutter 3.32.1
Backend	Laravel 11.45.1
Database	MySQL 8.0.33
IDE	Visual Studio Code (v16.11.46)

1.3.3 DEVELOPMENT ENVIRONMENT

- Development is conducted using Visual Studio Code for code editing, debugging, and version control.
- The frontend is built using Flutter (Dart) for cross-platform mobile app development, while the backend is developed using Laravel (PHP), ensuring a robust and scalable application structure.

1.3.4 DEPLOYMENT ENVIRONMENT

- CoiioT is deployed on a cloud platform to support scalability, enabling it to handle large volumes of real-time data efficiently.
- The mobile application is distributed through the Google Play Store (Android) and App Store (iOS) to provide seamless accessibility for users across platforms.
- Continuous Integration and Deployment (CI/CD) pipelines are utilized to automate testing, ensure code quality, and streamline deployment processes.

CHAPTER II

SYSTEM ANALYSIS

System analysis is a process of studying the system in order to identify the goals, purpose of the system and better understanding of the system's requirements. This chapter gives a brief discussion about the detailed study of the proposed system and the different functionalities involved in the system.

2.1 SYSTEM DESCRIPTION

This section provides an analysis of the existing system's challenges and introduces the proposed CoiioT system, outlining its features and benefits over the current setup.

2.1.1 EXISTING SYSTEM

In traditional coir factories, production monitoring, resource tracking, and reporting are primarily performed manually. Staff record production data, track resource usage, and maintain attendance and expense records by hand, often resulting in delays, inaccuracies, and inconsistent documentation. Manual reporting also makes it challenging for factory owners to identify production inefficiencies, machine downtime, or resource wastage in real time. Furthermore, the absence of a centralized system for visualizing historical trends or analyzing performance limits the ability to make informed operational decisions.

2.1.2 PROPOSED SYSTEM

The proposed CoiioT system automates production monitoring and resource tracking by collecting real-time data from IoT-enabled machines, eliminating the need for manual data collection and providing accurate insights into factory operations. The system processes and analyzes this data automatically, offering real-time dashboards and analytics to track key metrics, identify inefficiencies such as machine downtime and energy wastage, and visualize historical trends. CoiioT also supports administrative functions like attendance tracking, expense management, and report generation. Designed on a scalable architecture, the system ensures continuous data flow and provides secure user access through authentication mechanisms.

2.1.3 FEATURES OF THE PROPOSED SYSTEM

- **Automated Data Collection:** CoiioT automates data collection using IoT devices installed on factory machines. These devices continuously transmit data, ensuring accurate, real-time monitoring of production and resource usage while eliminating manual entry.
- **Real-Time Monitoring and Insights:** The system provides live dashboards and visual analytics that display machine states, production metrics, and resource usage, helping factory owners make quick and informed decisions.
- **Attendance & Expense Management:** The system tracks employee attendance and salary details. It also records expenses, including amounts, payers, and purposes.
- **Inventory Management:** The system allows efficient management of pith inventory, including tracking pith locations, pith units, loads offloaded, loads taken, and stock availability.
- **Notifications & Role-Based Access:** CoiioT sends push notifications with daily, weekly, monthly, and yearly production summaries. It also ensures secure access through role-based permissions assigned to different user roles.
- **External Integrations:** CoiioT integrates with third-party services, including Tomorrow.io for weather forecasting, Twilio for WhatsApp-based OTP authentication, and Firebase Cloud Messaging for push notifications.

2.2 USE CASE MODEL

A use case diagram is a visual tool used in system analysis to depict the interactions between users and a system. It provides an overview of the main functionalities, or "use cases," that a system offers and shows how various user roles interact with these functionalities. By clearly mapping out who can perform which actions, use case diagrams help to clarify system requirements, ensuring that the development team and stakeholders have a shared understanding of user needs and system capabilities.

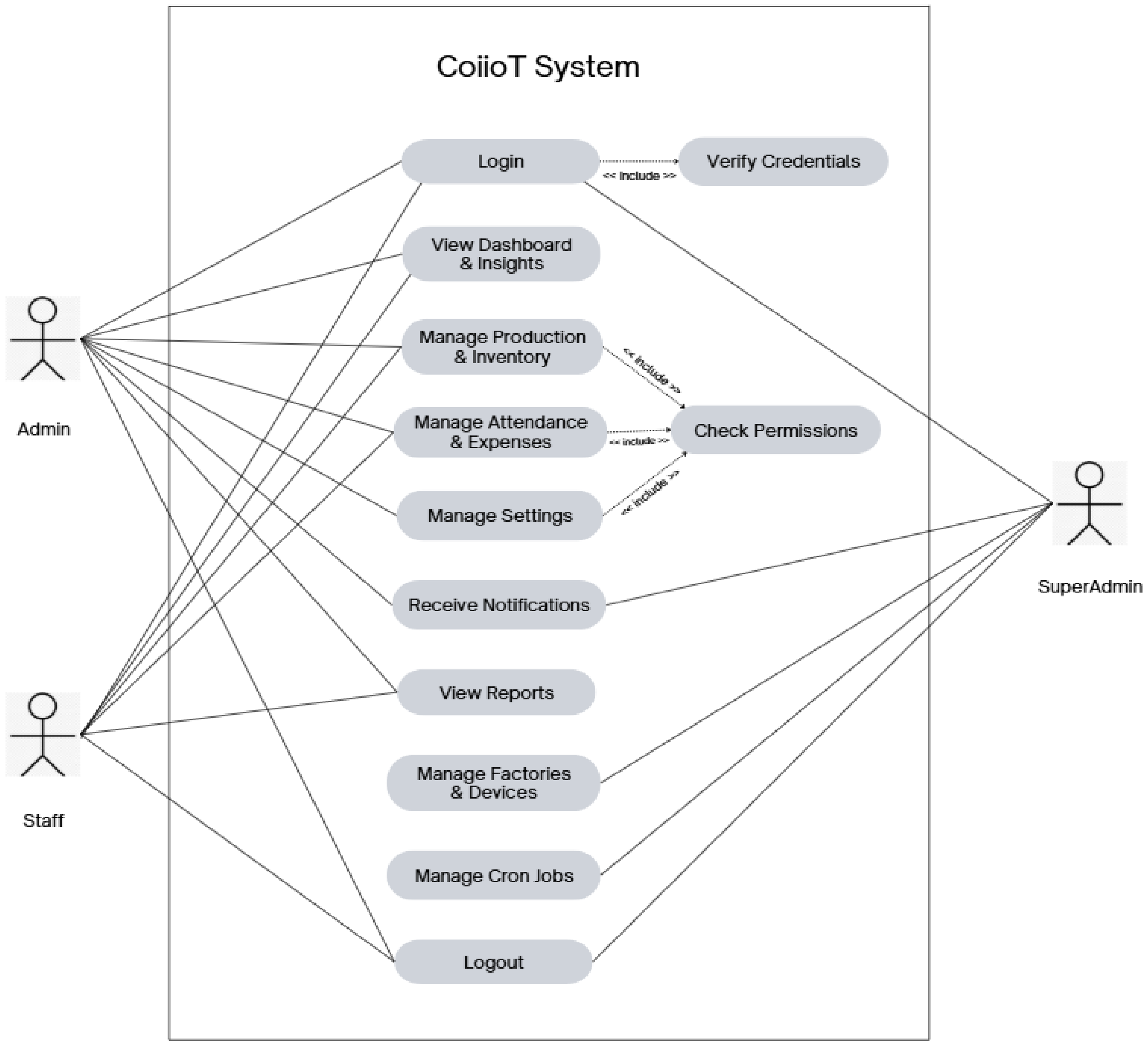


Figure 2.1 Use Case Model for CoiioT System

2.2.1 USE CASE DESCRIPTION

Table 2.1 - LOGIN

Goal	To allow users to securely access the CoiioT System based on their roles.
Actor(s)	SuperAdmin, Admin, Staff.
Description	Users authenticate themselves using their credentials to access the CoiioT System and perform actions allowed for their role.
Precondition	The user must have valid login credentials.
Postcondition	User is logged in and directed to their respective dashboard based on their role.
Main flow of events	1. User enters login credentials. 2. System validates credentials. 3. User is granted access and redirected to the dashboard.
Exceptions	If invalid credentials are provided or the user account is inactive or locked, access is denied.

Table 2.2 - MANAGE PRODUCTION & INVENTORY

Goal	To allow Admin and Staff to manage production and inventory records efficiently, ensuring accurate tracking of pith, fibre, and stock details.
Actor(s)	Admin, Staff
Description	Admin and Staff manage production (pith, fibre, etc.) and inventory records to keep data on loads offloaded, taken, and in-stock updated, with Admins having full control to create, edit, and delete records, while Staff can only create and edit them.
Precondition	User is logged into the system with valid permissions.
Postcondition	Production or inventory data is updated in the system.
Main flow of events	1. User accesses the Production or Inventory section. 2. User performs add, edit, or delete operations (based on their role). 3. System validates the input, updates the database, and reflects the changes to all authorized users.
Exceptions	Operation fails if the user lacks permissions or enters invalid data.

Table 2.3 - MANAGE ATTENDANCE & EXPENSES

Goal	To allow Admin and Staff to efficiently manage employee attendance and expense records, ensuring accurate tracking of working hours, salaries, and expenditures.
Actor(s)	Admin, Staff
Description	Admin and Staff manage employee attendance, salaries, and expenses with Admins having full control to create, edit, and delete records, while Staff can only create and edit them.
Precondition	User is logged into the system with valid permissions.
Postcondition	Attendance and expense data is updated and available for reporting.
Main flow of events	1. User accesses the Attendance or Expenses section. 2. User performs add, edit, or delete operations (based on their role). 3. System validates the input, updates the database, and reflects the changes to all authorized users.
Exceptions	Operation fails if the user lacks permissions or enters invalid data.

Table 2.4 - MANAGE SETTINGS

Goal	To allow Admins to configure and manage factory-related settings for the accurate operation of the system.
Actor(s)	Admin
Description	Admins manage factory settings such as shift timings, staff details, pith locations, pith units, and custom settings like EB units per pith load and charges per EB unit to ensure the system functions according to factory requirements.
Precondition	Admin is logged into the system.
Postcondition	The updated settings are saved and applied across the system.
Main flow of events	1. Admin opens the Settings section. 2. Admin adds or updates shift, staff, and custom settings. 3. System validates and saves the changes.
Exceptions	Operation fails if invalid data is entered or a system error occurs.

Table 2.5 - MANAGE FACTORIES & DEVICES

Goal	To allow the SuperAdmin to manage factories and IoT device configurations.
Actor(s)	SuperAdmin
Description	The SuperAdmin manages factories and device settings, including configuration of underload and efficiency thresholds, message intervals, and missing check duration. The SuperAdmin also validates factory license expiry and invites users to join respective factories.
Precondition	SuperAdmin is logged into the system.
Postcondition	Factory and device configurations are updated and reflected across the system.
Main flow of events	1. SuperAdmin adds or updates factory and device configurations. 2. SuperAdmin validates factory license status and invites users to teams. 3. The system validates inputs, updates the database, and applies the changes.
Exceptions	Operation fails if invalid data is entered, or a system error occurs during the update.

2.3 SOFTWARE REQUIREMENTS SPECIFICATION

A software requirements specification (SRS) is a comprehensive document that outlines the necessary requirements for a software product. It includes both functional and non-functional requirements and often involves the use of use cases to describe user interactions with the software.

2.3.1 FUNCTIONAL REQUIREMENTS

- **User Authentication**

  CoiioT provides secure login for SuperAdmin, Admin, and Staff roles, each with specific access permissions.SuperAdmin manages factories and devices, Admin oversees operations, and Staff records daily data. Role-based access ensures data security and prevents unauthorized access.

- **Factory and Device Management**

  The SuperAdmin manages factories and IoT device configurations, including setting thresholds, message intervals, and device status. They also validate factory license expiry and invite users to join teams. The system updates configurations across all modules, ensuring only active devices are monitored.

- **Automated Data Collection**

  CoiioT automates data collection through IoT devices installed on key machines like defibre units, baling machines, pith block machines, and water meters. These devices automatically send operational data to the cloud at regular intervals using the MQTT protocol, enabling accurate and real-time monitoring of production and resource usage.

- **Inventory Management**

  CoiioT allows efficient management of pith inventory, including tracking pith locations, pith units, loads offloaded, loads taken, and stock availability.

- **Attendance and Expense Management**

  CoiioT helps manage employee attendance and daily factory expenses. It records employee working hours, salaries, and expense details like amount, payer, and purpose. Admins can generate reports for easy analysis and payroll processing.

- **Dashboard and Visualization**

  CoiioT includes an interactive dashboard that displays real-time and historical data, showing pith and fibre production, machine states, water usage, and EB consumption using charts. Users can view data by day, month, or a custom date range for easy analysis.

- **Notifications and Alerts**

  CoiioT sends real-time notifications for production summaries, alerts, and system updates using Firebase Cloud Messaging. Admins receive daily, weekly, monthly, and yearly production summaries, while SuperAdmins are alerted about missed or failed cron jobs.

2.3.2 NON-FUNCTIONAL REQUIREMENTS

- **Performance**

  CoiioT is designed to efficiently handle high volumes of data from numerous devices while providing fast response times to user actions, ensuring smooth, reliable, and efficient operation.

- **Scalability**

  The CoiioT architecture supports scaling to accommodate future increases in factories, device numbers, users, and data volume. This ensures that the system remains performant as demands grow over time.

- **Reliability and Availability**

  CoiioT aims for 99.9% uptime, ensuring continuous monitoring and data availability. Alerts and notifications remain operational even during system maintenance, supporting uninterrupted decision-making.

- **Security**

  Data security is paramount, with HTTPS encryption for data transmission. Strict role-based access control and permissions ensure that sensitive information is protected from unauthorized access.

- **Usability**

  CoiioT's interface is designed for ease of use, with an intuitive dashboard and user-friendly navigation. Clear visuals and organized displays enable users to quickly interpret data and take timely action.

- **Maintainability**
  CoiioT is built with a modular architecture, making updates, troubleshooting, and future feature additions straightforward. Comprehensive documentation ensures maintainers can efficiently manage the system.
- **Data Accuracy**
  High data accuracy is essential for CoiioT, ensuring precise device readings with minimal discrepancies.

2.3.3 ASSUMPTIONS AND DEPENDENCIES

ASSUMPTIONS

The CoiioT system operates under several key assumptions to ensure effective performance and user satisfaction. It is assumed that users have a basic understanding of factory monitoring and production management systems and are adequately trained to use the CoiioT interface. Additionally, it is presumed that all IoT devices are fully operational and properly configured to communicate data to the CoiioT application. The system also assumes reliable internet connectivity to facilitate seamless data transfer between devices and the cloud platform.

DEPENDECIES

The successful implementation of the CoiioT system relies on several critical dependencies. These include a stable cloud infrastructure capable of handling real-time data processing and storage. The application also depends on the integration with third-party services, such as tomorrow.io for weather forecasts, Twilio for WhatsApp OTP login, and FCM for push notifications, all of which require appropriate API access and permissions. Furthermore, the system's performance is contingent on the correct configuration and maintenance of the underlying hardware and software components, including network and IoT devices. Lastly, ongoing user training and support are essential for maximizing system utilization and effectiveness.

2.4 TEST PLAN

The test plan for the CoiioT system provides a structured approach to validate that the software meets both functional and non-functional requirements essential for efficient coir factory operations. CoiioT integrates real-time production and resource monitoring, automated data collection from IoT devices, inventory management, attendance and expense tracking, and notification systems. The test plan includes multiple testing methodologies, such as unit testing, integration testing, system testing, and performance testing, to ensure accurate, reliable, and scalable operation. Usability testing is also conducted to verify that the interface is intuitive for SuperAdmins, Admins, and Staff, allowing them to efficiently manage factories, monitor production, and respond promptly to alerts and notifications.

2.4.1 TEST SCOPE

The test scope for the CoiioT system covers all essential functionalities critical to production monitoring, resource tracking, and administrative management in coir factories. Testing will include core modules such as real-time data collection from IoT-enabled machines, inventory management, attendance and expense tracking, and notifications. The scope also includes verifying the integration of IoT devices, Firebase Cloud Messaging, Twilio OTP authentication, and weather forecasting from Tomorrow.io. Performance and responsiveness of the interactive dashboards under varying data loads will be evaluated. Additionally, user authentication, role-based access control, factory and device management, and report generation workflows will be tested to ensure smooth and secure operations for SuperAdmins, Admins, and Staff.

2.4.2 TESTING FUNCTIONS

1. **User Authentication:** Verification of secure login processes for SuperAdmin, Admin, and Staff roles, ensuring that only authorized users can access their respective functionalities.
2. **Factory and Device Management:** Testing the ability of SuperAdmin to add, update, activate or deactivate factories, and configure IoT devices, including thresholds, message intervals, and device status, ensuring accurate updates across the system.
3. **Automated Data Collection:** Validation of real-time data collection from IoT-enabled machines, confirming that production and resource usage data are correctly

transmitted to the cloud and reflected in the system.

4. **Inventory Management:** Ensuring accurate tracking of pith inventory, including locations, units, loads offloaded, loads taken, and stock availability.
5. **Attendance and Expense Management:** Testing the recording and reporting of employee attendance, salary details, and daily factory expenses, ensuring data is accurate and reports are generated correctly.
6. **Dashboard and Visualization:** Ensuring that the interactive dashboards display real-time and historical data for production, machine states, water usage, and EB consumption. Tests will validate responsiveness and accuracy under varying data loads.
7. **Notifications and Alerts:** Testing the push notification system (Firebase Cloud Messaging) to ensure Admins receive production summaries and SuperAdmins get cron job alerts accurately and on time.

2.4.3 TESTING TECHNIQUES

1. **Functional Testing:** This technique ensures that each of CoiioT's core functionalities, such as data collection, factory and device management, and notifications, works as specified, providing the intended results without errors.
2. **Performance Testing:** Given the high volume of data CoiioT processes in real time, performance testing is critical. It measures system responsiveness, stability, and scalability under typical and peak data loads to ensure CoiioT performs reliably and efficiently.
3. **Regression Testing:** Regression testing in CoiioT is performed during version updates or feature enhancements to ensure that new changes do not disrupt existing functionalities. It ensures that core modules continue to operate accurately, maintaining overall system stability, reliability, and performance after each release.
4. **UI Testing:** UI testing in CoiioT focuses on verifying the usability, design consistency, and responsiveness of the mobile interface. It ensures that dashboards, charts, and data visualizations are displayed accurately and intuitively across various devices.

2.4.4 TESTING TOOLS

1. **Android Emulators/Simulators:** These are used to test the functionality, usability, and performance of the CoiioT mobile application across various device types, screen sizes, and OS versions. They ensure that all UI components and workflows operate smoothly and consistently before deployment.
2. **Postman:** This is essential for API testing, where it is used to validate endpoint functionality and accuracy. With Postman, testers can verify the system's ability to handle data requests and responses accurately by performing CRUD operations, checking response headers, validating status codes, and automating test collections. This helps to confirm the APIs' correct functionality and seamless integration with other components.

CHAPTER III

SYSTEM DESIGN

System design is a set of procedures performed to convert the needs and requirements of a business or organization into a design that can be implemented on the organization’s computer system. It identifies the software components, specifies relationships among the components, properties of both components and relations, defines program structure and provides a blueprint for implementation. The following chapter deals with the various design issues that guide the interface development of the application.

3.1 ARCHITECTURAL DESIGN

Architectural design is the process of defining a collection of structure of data, hardware and software components and their interfaces to establish the framework for the development of a computer-based system. It focuses on the decomposition of a system into different components and their interactions to satisfy functionality and performance requirements as well as non-functional requirements of the system. The software that is built for computer-based systems can exhibit one of these many architectural styles or patterns. An architectural pattern is a general, reusable solution to a commonly occurring problem in software architecture within a given context.

The CoiioT System is designed using a layered architecture that integrates IoT devices, cloud infrastructure, and a web-based interface to ensure modularity, scalability, and real-time responsiveness. At the device layer, IoT sensors collect operational data and transmit it using the MQTT protocol to an MQTT Broker hosted on an AWS EC2 server. This broker acts as the central hub for receiving and routing device data. Scheduled Cron Scripts on the server process incoming data and store it in a MySQL database, ensuring consistent logging and availability for analysis. This setup allows for efficient data handling and supports continuous monitoring of field equipment.

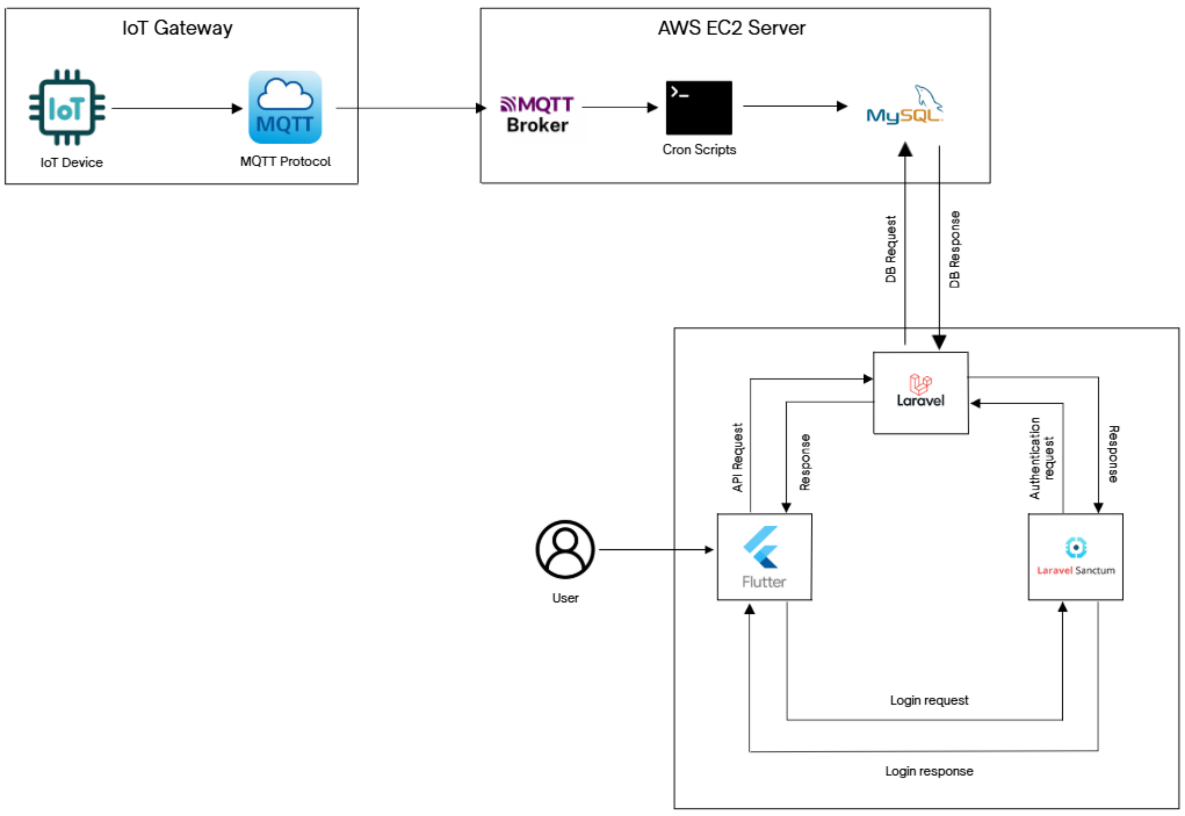


Figure 3.1 Architectural Design for CoiioT

On the application side, the system uses the Laravel framework to manage API requests, user authentication, and business logic. Laravel services interact with the MySQL database to retrieve production metrics, user profiles, and configuration settings. The frontend is built with Flutter, offering users a responsive dashboard for viewing production insights, reports, and system controls. By separating responsibilities across distinct layers—device communication, server-side processing, and user interaction—the architecture supports independent scaling and maintenance of each component.

3.2 STRUCTURAL DESIGN

Structured design is a diagrammatic notation that is designed to help people understand the system. The basic goal of structured design is to improve quality and reduce the risk of system failure. It establishes concrete management specifications and documentation. It focuses on the solidity, pliability and maintainability of the system.

3.2.1 MODULE DIAGRAM

A module diagram is a design approach that subdivides a system into smaller parts called modules that can be independently created and then used in different systems. A modular system can be characterized by functional partitioning into discrete scalable, reusable modules; rigorous use of well-defined modular interfaces; and making use of industry standards for interfaces.

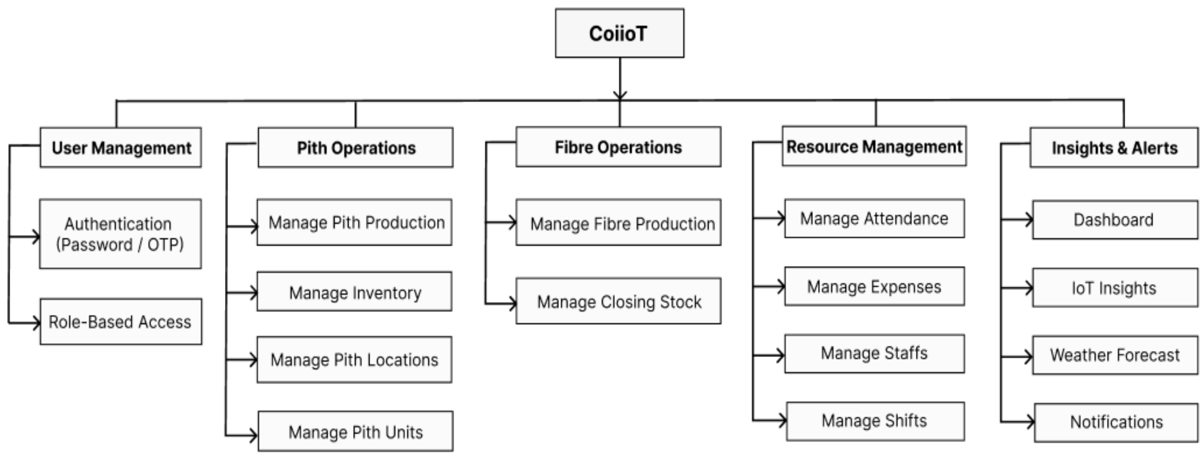


Figure 3.2 Module Design of CoiioT

User Management:

- ***Authentication (Password / OTP):*** Provides secure login functionality for SuperAdmin, Admin, and Staff using password or WhatsApp OTP. Ensures only authorized users can access the system.
- ***Role-Based Access:*** Assigns specific permissions based on roles. SuperAdmin manages factories and devices, Admin oversees production, and Staff handles daily operations. Prevents unauthorized access to critical functions.

Pith Operations:

- ***Manage Pith Production:*** Allows recording and monitoring of pith production in real-time, including shift-wise tracking, ensuring accurate and up-to-date data.
- ***Manage Inventory:*** Tracks pith stock, including loads offloaded, loads taken, and overall stock availability.
- ***Manage Pith Locations:*** Maintains information about pith storage locations across the factory for easy access and tracking.
- ***Manage Pith Units:*** Monitors and manages units of pith produced, linking production quantities (e.g., tractor loads, bag counts) with inventory records.

Fibre Operations:

- ***Manage Fibre Production:*** Allows recording and monitoring of fibre production in real-time, including shift-wise tracking, ensuring accurate and up-to-date data.
- ***Manage Closing Stock:*** Keeps track of fibre stock at the end of each day or shift for inventory management.

Resource Management:

- ***Manage Attendance:*** Records staff working hours, calculates salaries, and maintains attendance logs.
- ***Manage Expenses:*** Tracks daily factory expenses, including amounts, payers, and purposes.
- ***Manage Staffs:*** Maintains staff profiles, including status (Active/Inactive) and hourly salary, ensuring proper record-keeping and payroll management.

- ***Manage Shifts:*** Configures shift timings, assigns staff, and monitors shift-specific production and resource usage.

Insights & Alerts:

- ***Dashboard:*** Provides an interactive dashboard showing pith and fibre production, machine states, water and EB consumption. Users can view data by day, month, or custom ranges.
- ***IoT Insights:*** Displays real-time and historical data from IoT devices installed on factory machines for detailed monitoring and analysis.
- ***Weather Forecast:*** Integrates with tomorrow.io to provide weather updates, assisting in production planning.
- ***Notifications:*** Sends push notifications via Firebase Cloud Messaging for production summaries, alerts, and system updates. Admins receive daily, weekly, monthly, and yearly summaries, while SuperAdmins are notified of failed or missed cron jobs.

3.3 BEHAVIORAL DESIGN

Behavioral Design is a systematic approach that focuses on how a system responds to various inputs and events, aiming to enhance user experience and functionality. It emphasizes clear use cases and workflows, promoting a better understanding of user interactions. By concentrating on a system's dynamic aspects, behavioral design creates adaptable systems that evolve with user needs, supporting robust documentation and effective communication among stakeholders to minimize errors during implementation.

3.3.1 ACTIVITY DIAGRAM

Activity Diagram is a visual representation used to model the dynamic aspects of a system by illustrating the flow of control or data among various activities. It provides a clear view of the sequence of operations, decision points, and parallel processes within a specific workflow. By detailing the steps involved in a process, activity diagrams facilitate understanding, analysis, and communication among stakeholders, making them essential for identifying inefficiencies and optimizing system behavior.

Figure 3.3 presents the activity diagram for the CoiioT system, illustrating the role-based workflow and dynamic interactions between SuperAdmin and Admin/Staff users. The diagram begins with a login request, followed by credential validation and appropriate error handling for failed authentication attempts. Upon successful login, the system checks the user's role and directs them to corresponding functionalities. SuperAdmin users are granted access to advanced administrative controls such as managing device settings, factories, and cron jobs, while Admin/Staff users are directed to dashboards displaying production metrics and operational summaries.

The diagram further outlines the conditional logic for managing production records, where Admin users are authorized to perform full CRUD operations – including delete, while Staff users are restricted to create and update actions only. These permissions are enforced through role checks and authorization gates within the system. All users eventually proceed to the logout flow, which includes session token invalidation and confirmation of logout success. This activity diagram highlights the structured flow of operations, emphasizing secure access control, modular functionality, and differentiated user capabilities within the CoiioT system.

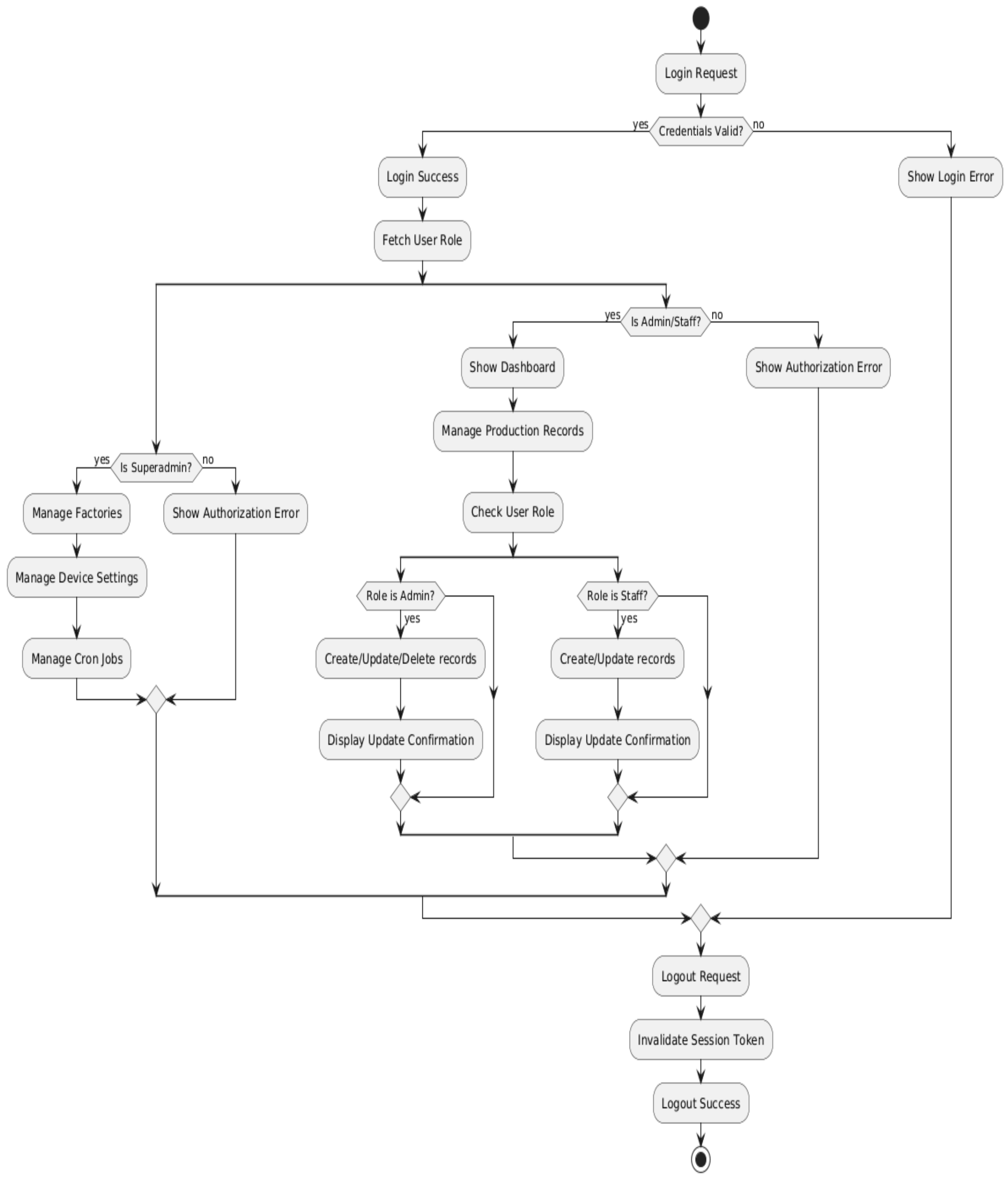


Figure 3.3 Activity Diagram of CoiioT

3.3.2 SEQUENCE DIAGRAM

A sequence diagram is a type of interaction diagram that models the sequence of messages exchanged between objects or components in a system over time. It illustrates how processes operate and the order in which interactions occur, highlighting the timing and flow of messages in a visually structured format. Sequence diagrams are essential for understanding dynamic behaviour within a system, as they detail the relationships and dependencies between different components, facilitating clear communication of system functionalities and requirements.

Figure 3.4 illustrates the sequence of interactions between core actors and system components within the CoiioT system. It captures the operational flow involving SuperAdmin, Admin/Staff, the System, and the Database, highlighting key processes such as user authentication, dashboard access, device management, production updates, and session termination. This diagram provides a structured view of how different user roles engage with the system and how data flows between the application and its backend services.

The sequence begins with SuperAdmin and Admin/Staff initiating login requests. The system validates credentials by querying the database and, upon successful authentication, generates and stores session tokens. The SuperAdmin then accesses device settings, while the Admin/Staff view the production dashboard, triggering data retrieval from the database. When performing CRUD operations, the system verifies user roles – allowing Admins to execute full Create, Update, and Delete actions, while Staff are restricted to Create and Update only. All users conclude their session with a logout request, prompting the system to invalidate tokens and remove session data. This flow ensures secure access control, role-based permissions, and efficient data handling across the CoiioT system.

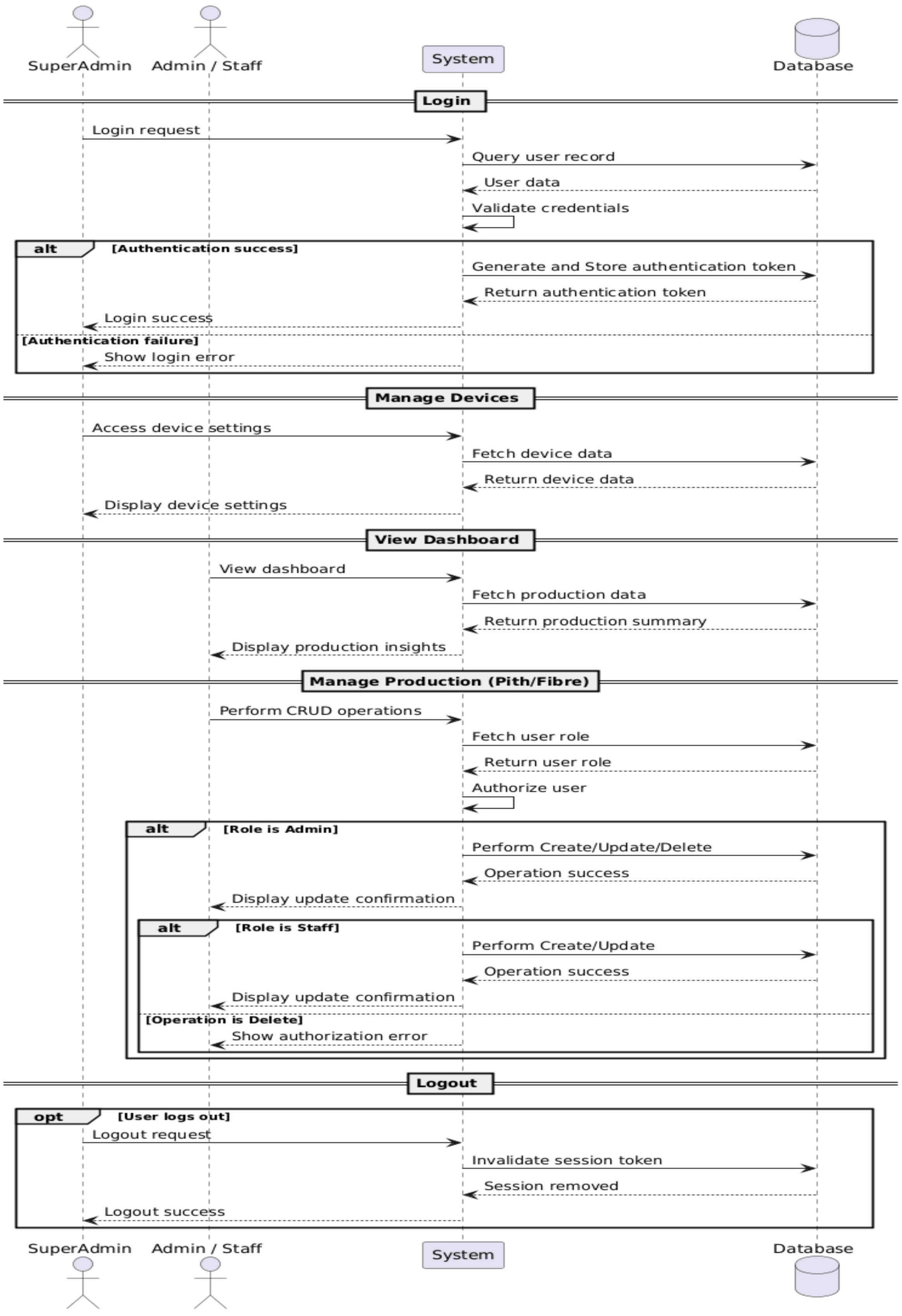


Figure 3.4 Sequence Diagram of CoiioT

3.4 TABLE DESIGN

Database Design is a collection of processes that facilitate the designing, development, implementation and maintenance of enterprise data management systems. Sample table design is as follows.

3.4.1 ENTITY RELATIONSHIP DIAGRAM

An Entity-Relationship Diagram (ERD) is a visual representation of the entities within a system and their interrelationships. It serves as a foundational blueprint for database design and helps in understanding the data structure and organization. ERDs are composed of several key components:

1. **Entities:** These are objects or things within the system that have a distinct existence. They can represent physical items, concepts, or events. In an ERD, entities are usually depicted as rectangles.
2. **Attributes:** Attributes are the properties or characteristics of an entity.
3. **Relationships:** Relationships illustrate how entities are connected to one another. They define the associations between entities, such as one-to-one, one-to-many, or many-to many. Relationships are typically represented by diamonds connecting related entities.
4. **Cardinality:** This term refers to the number of instances of one entity that can be associated with instances of another entity. It provides crucial information about the nature of the relationship between entities.
5. **Primary Key (PK):** A primary key is a unique identifier for an entity, ensuring that each instance can be distinguished from others. It is often underlined in the diagram.
6. **Foreign Key (FK):** A foreign key is an attribute that creates a link between two entities, referencing the primary key of another entity. This helps establish relationships and maintain referential integrity.

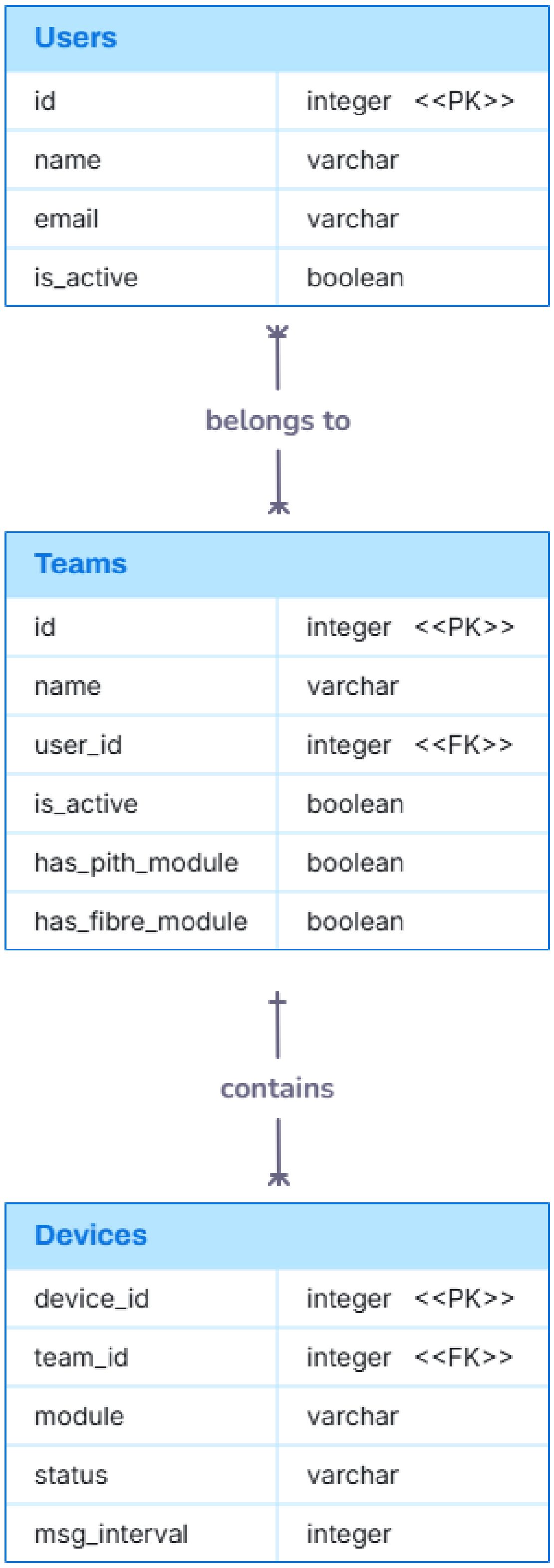


Figure 3.5 ER Diagram of CoiioT – Part 1

The ER diagram depicted in **Figure 3.5** illustrates the relationships among three core entities: **Users**, **Teams**, and **Devices**. The Users entity includes attributes such as id, name, email, and is_active, representing individual users who interact with the system. Each user can be associated with one or more teams, as indicated by the foreign key user_id in the Teams table. The Teams entity contains fields like id, name, is_active, and module flags such as has_pith_module and has_fibre_module, which define the operational scope of each team.

The Devices entity captures attributes such as device_id, module, status, and msg_interval, along with a foreign key team_id that links each device to a specific team. This structure establishes a many-to-many relationship between Users and Teams, and one-to-many relationship between Teams and Devices. Together, these relationships reflect a hierarchical model where users oversee teams, and teams manage multiple devices. This design supports modular deployment, efficient device tracking, and scalable team-based management within the coir production system.

The ER diagram depicted in **Figure 3.6** illustrates the relationship between three interconnected entities: **PithProductions**, **PithLoads**, and **PithLocations**. The PithProductions entity includes attributes such as id, team_id, production_date, and total_production, representing the overall daily output of pith material by a specific team. The PithLoads entity captures details of individual loading activities, including id, pith_production_id, pith_location_id, load_date, and quantity, linking each load to its corresponding production and physical location. The PithLocations entity contains attributes like id and location_name, identifying the various sites where pith loads are stored or dispatched.

The relationships in this schema reflect a structured flow of data where each production can be associated with multiple pith loads, and each load is tied to a specific location. This design establishes a one-to-many relationship between PithProductions and PithLoads, and another many-to-one relationship between PithLoads and PithLocations. By organizing production, loading, and location data into distinct but connected entities, the system ensures accurate tracking of material movement and supports efficient reporting and operational oversight within the coir production workflow.

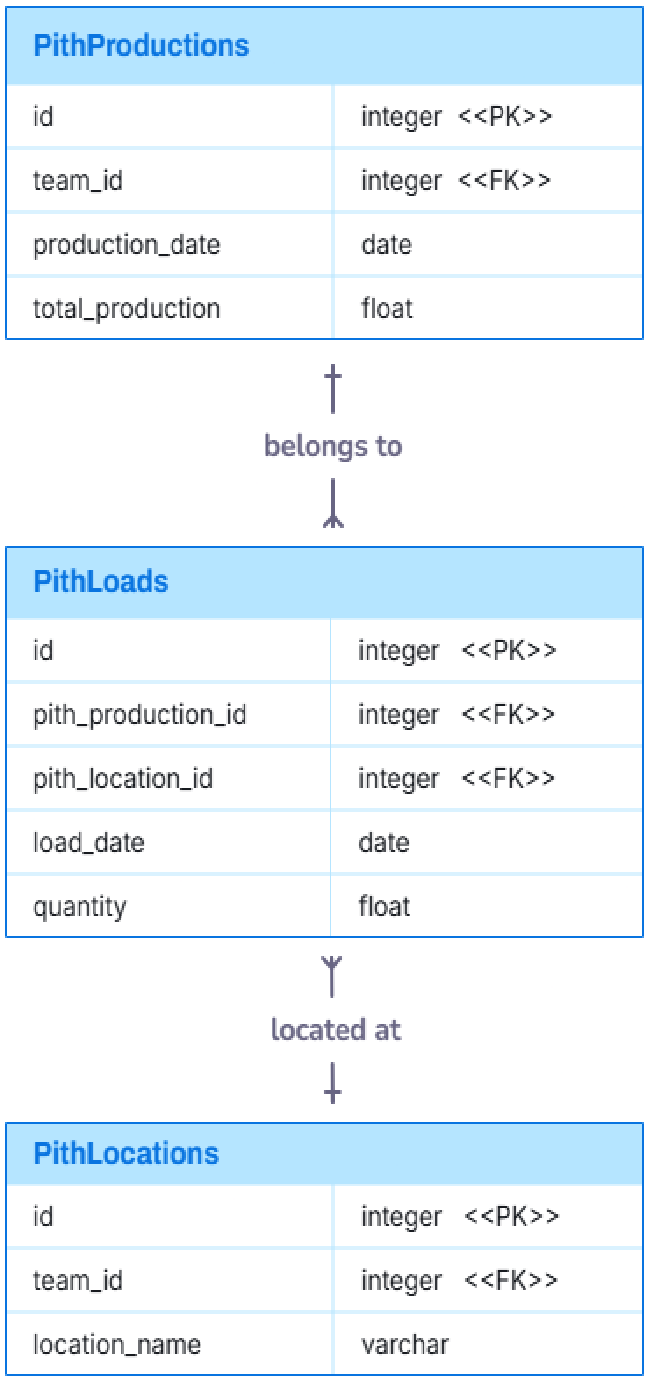
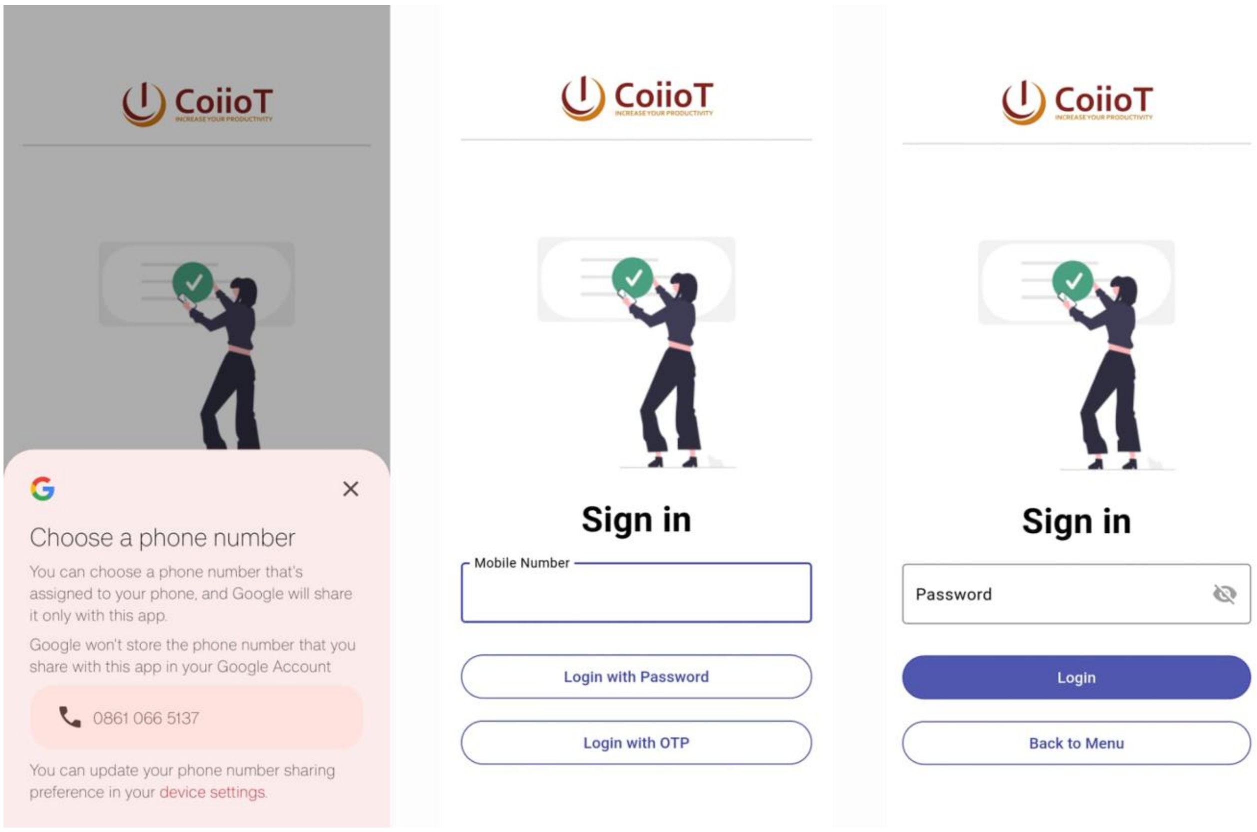


Figure 3.6 ER Diagram of CoiioT – Part 2

3.5 USER INTERFACE DESIGN

The User Interface Diagram for the CoiioT system outlines the application's screen layouts and form interactions through detailed wireframes. This diagram provides insight into the system's visual structure, with flexibility in the level of detail based on design requirements. It can depict a basic layout, highlight specific interface sections, or present a comprehensive view of various UI components.

The user interface (UI) design of the CoiioT system plays a crucial role in ensuring intuitive and efficient monitoring of production processes, machine states, and resource usage. By emphasizing clarity and ease of navigation, the UI enables administrators and staff to interact with real-time production data effortlessly, providing quick access to key features such as dashboards, reports, notifications, and settings. With its streamlined and well-organized layout, CoiioT's UI simplifies complex industrial data, helping users make informed decisions swiftly while minimizing potential errors. This design approach enhances usability, operational transparency, and productivity across factory workflows.


Choose a phone number

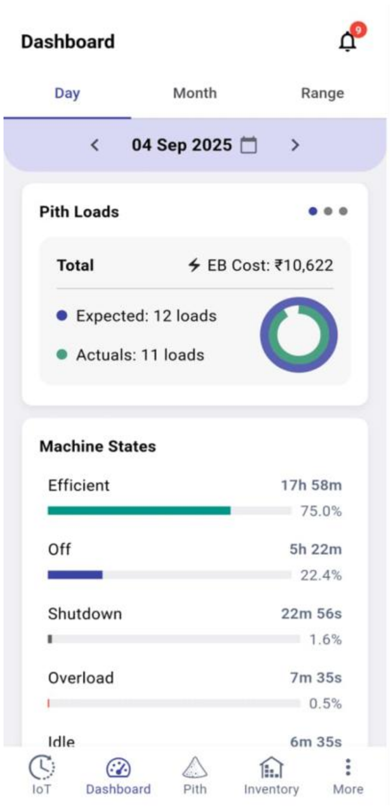
You can choose a phone number that's assigned to your phone, and Google will share it only with this app.

Google won't store the phone number that you share with this app in your Google Account

0861 066 5137

You can update your phone number sharing preference in your device settings.

Screen 3.1 - Login



Dashboard

Day | Month | Range

04 Sep 2025

Pith Loads

Total — EB Cost: ₹10,622

Expected: 12 loads

Actuals: 11 loads

Machine States

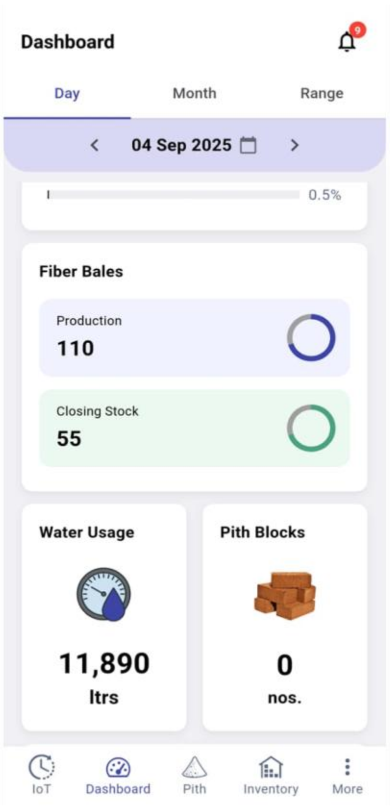
Efficient 17h 58m 75.0%

Off 5h 22m 22.4%

Shutdown 22m 56s 1.6%

Overload 7m 35s 0.5%

Idle 6m 35s



Dashboard

Day | Month | Range

04 Sep 2025

0.5%

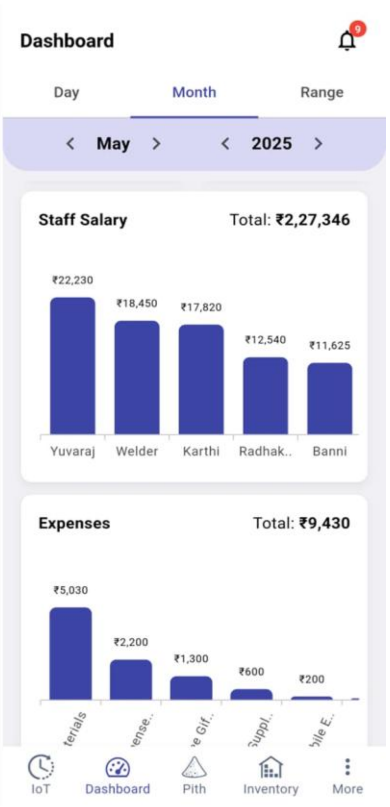
Fiber Bales

Production 110

Closing Stock 55

Water Usage 11,890 ltrs

Pith Blocks 0 nos.



Dashboard

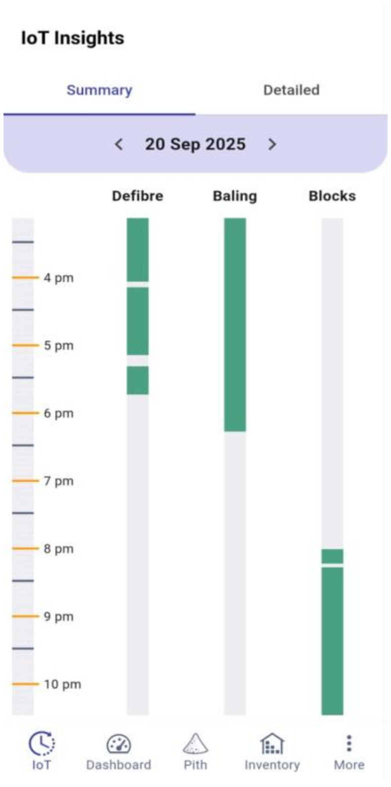
Day | Month | Range

May 2025

Staff Salary Total: ₹2,27,346

Expenses Total: ₹9,430

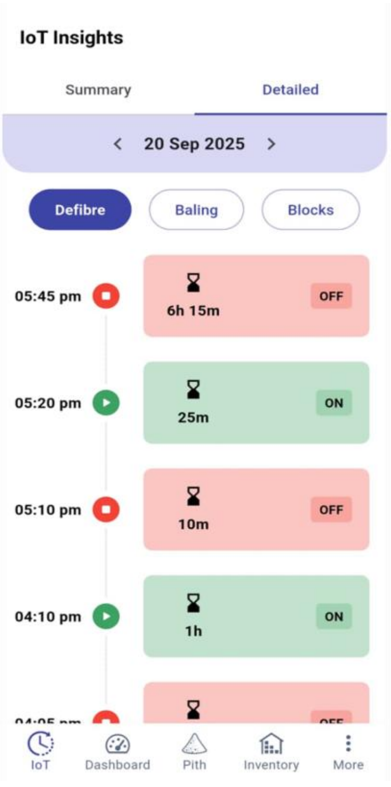
Screen 3.2 - Dashboard



IoT Insights

Summary | Detailed

20 Sep 2025



IoT Insights

Summary | Detailed

20 Sep 2025

Defibre | Baling | Blocks

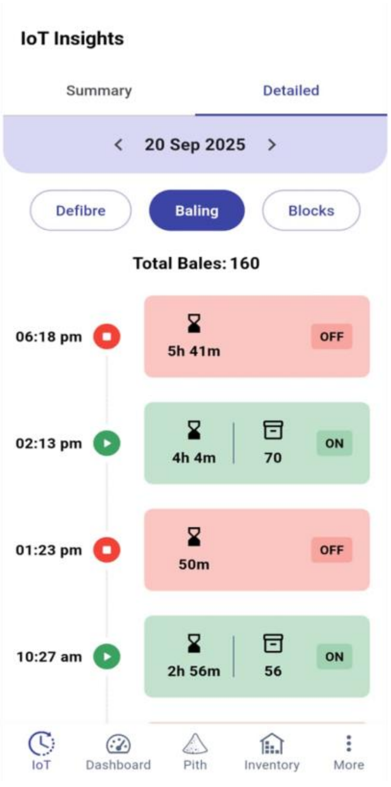
05:45 pm — 6h 15m — OFF

05:20 pm — 25m — ON

05:10 pm — 10m — OFF

04:10 pm — 1h — ON

04:05 pm — OFF



IoT Insights

Summary | Detailed

20 Sep 2025

Defibre | Baling | Blocks

Total Bales: 160

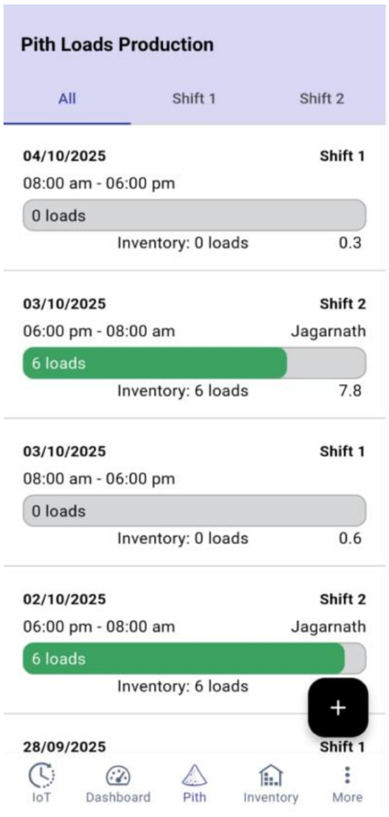
06:18 pm — 5h 41m — OFF

02:13 pm — 4h 4m — 70 — ON

01:23 pm — 50m — OFF

10:27 am — 2h 56m — 56 — ON

Screen 3.3 - IoT Insights


Pith Loads Production

All | Shift 1 | Shift 2

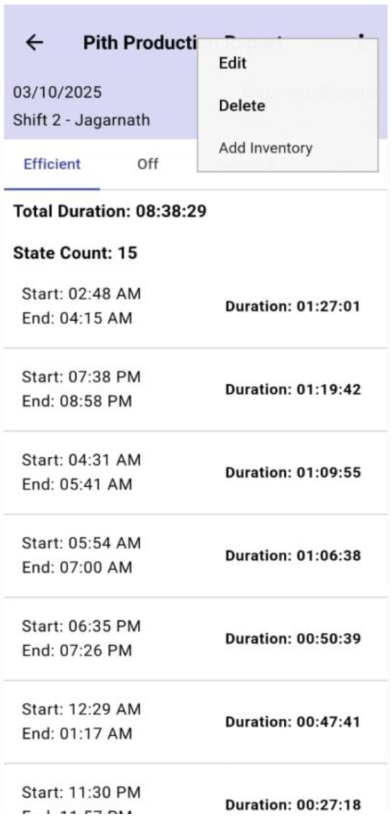
04/10/2025 Shift 1
08:00 am - 06:00 pm
0 loads
Inventory: 0 loads 0.3

03/10/2025 Shift 2
06:00 pm - 08:00 am Jagarnath
6 loads
Inventory: 6 loads 7.8

03/10/2025 Shift 1
08:00 am - 06:00 pm
0 loads
Inventory: 0 loads 0.6

02/10/2025 Shift 2
06:00 pm - 08:00 am Jagarnath
6 loads
Inventory: 6 loads

28/09/2025 Shift 1

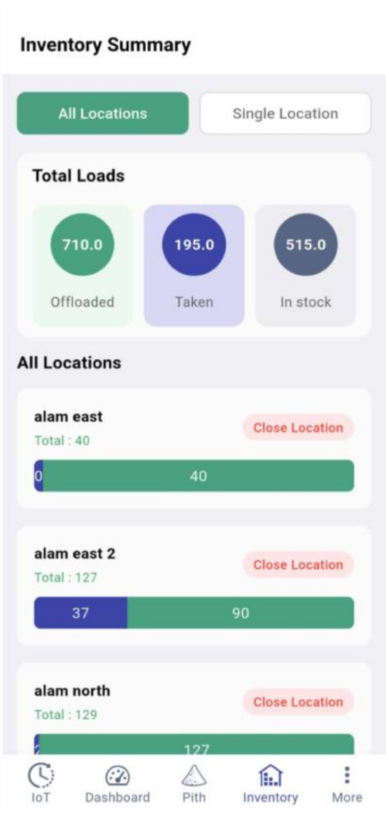

Pith Producti

03/10/2025
Shift 2 - Jagarnath

Total Duration: 08:38:29

State Count: 15

Start: 02:48 AM End: 04:15 AM	Duration: 01:27:01
Start: 07:38 PM End: 08:58 PM	Duration: 01:19:42
Start: 04:31 AM End: 05:41 AM	Duration: 01:09:55
Start: 05:54 AM End: 07:00 AM	Duration: 01:06:38
Start: 06:35 PM End: 07:26 PM	Duration: 00:50:39
Start: 12:29 AM End: 01:17 AM	Duration: 00:47:41
Start: 11:30 PM	Duration: 00:27:18


Inventory Summary

Total Loads

Offloaded	Taken	In stock
710.0	195.0	515.0

All Locations

alam east
Total : 40

0 | 40

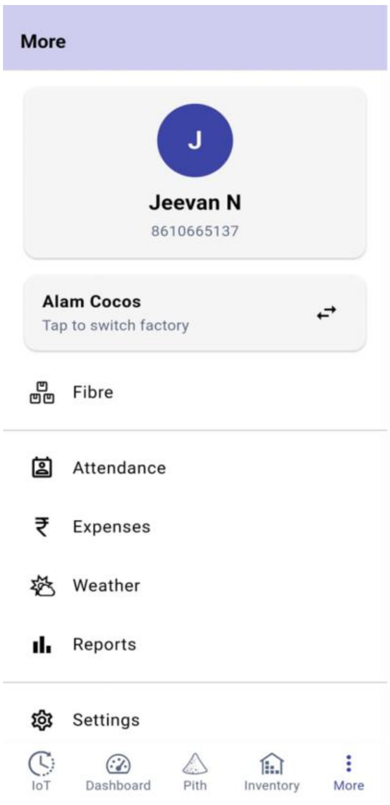
alam east 2
Total : 127

37 | 90

alam north
Total : 129

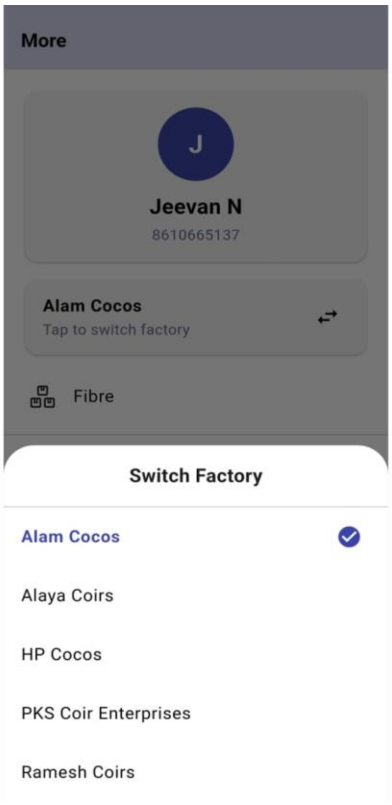
127

Screen 3.4 - Pith Production and Pith Inventory


More

J
Jeevan N
8610665137

Alam Cocos
Tap to switch factory

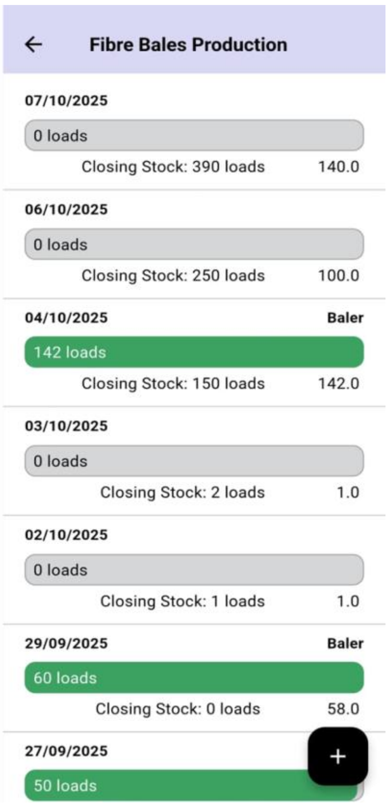

More

J
Jeevan N
8610665137

Alam Cocos
Tap to switch factory

Switch Factory

Alam Cocos ✓
Alaya Coirs
HP Cocos
PKS Coir Enterprises
Ramesh Coirs


Fibre Bales Production

07/10/2025
0 loads
Closing Stock: 390 loads 140.0

06/10/2025
0 loads
Closing Stock: 250 loads 100.0

04/10/2025 Baler
142 loads
Closing Stock: 150 loads 142.0

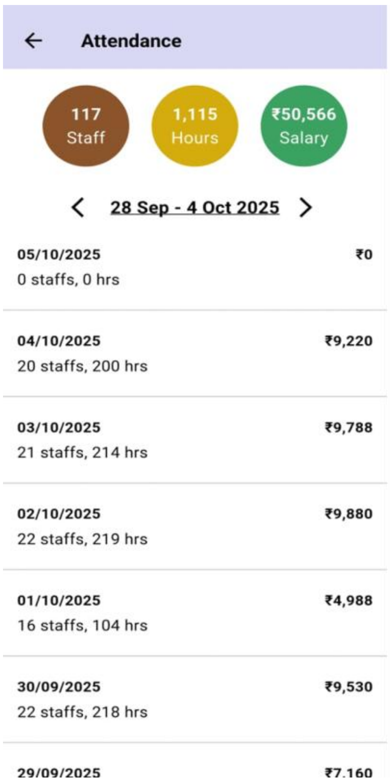
03/10/2025
0 loads
Closing Stock: 2 loads 1.0

02/10/2025
0 loads
Closing Stock: 1 loads 1.0

29/09/2025 Baler
60 loads
Closing Stock: 0 loads 58.0

27/09/2025
50 loads

Screen 3.5 - More Options and Fibre Production

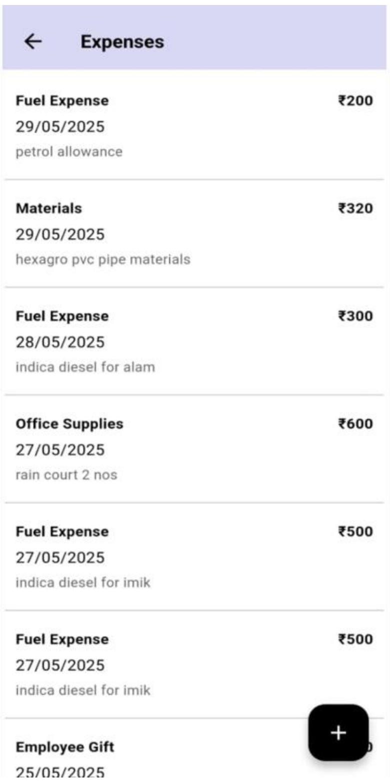


Attendance

117 Staff | 1,115 Hours | ₹50,566 Salary

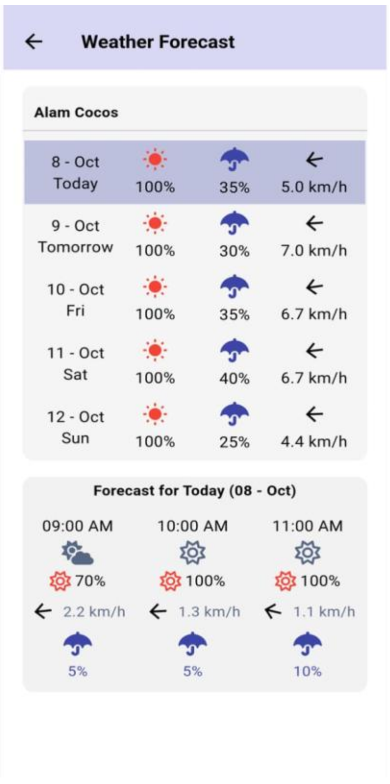
28 Sep - 4 Oct 2025

05/10/2025 0 staffs, 0 hrs	₹0
04/10/2025 20 staffs, 200 hrs	₹9,220
03/10/2025 21 staffs, 214 hrs	₹9,788
02/10/2025 22 staffs, 219 hrs	₹9,880
01/10/2025 16 staffs, 104 hrs	₹4,988
30/09/2025 22 staffs, 218 hrs	₹9,530
29/09/2025	₹7,160



Expenses

Fuel Expense 29/05/2025 petrol allowance	₹200
Materials 29/05/2025 hexagro pvc pipe materials	₹320
Fuel Expense 28/05/2025 indica diesel for alam	₹300
Office Supplies 27/05/2025 rain court 2 nos	₹600
Fuel Expense 27/05/2025 indica diesel for imik	₹500
Fuel Expense 27/05/2025 indica diesel for imik	₹500
Employee Gift 25/05/2025	



Weather Forecast

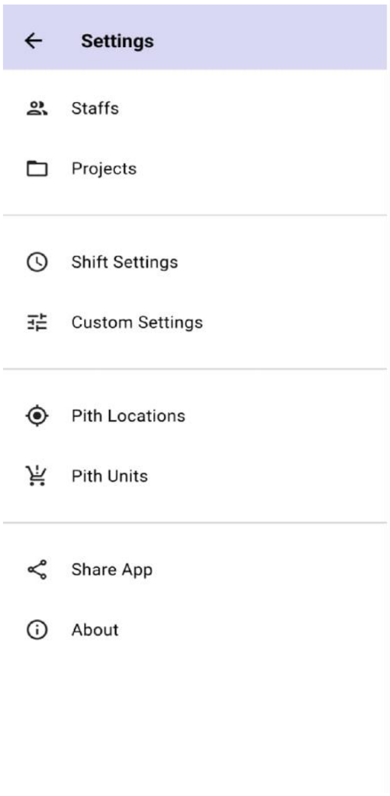
Alam Cocos

8 - Oct Today	100%	35%	5.0 km/h
9 - Oct Tomorrow	100%	30%	7.0 km/h
10 - Oct Fri	100%	35%	6.7 km/h
11 - Oct Sat	100%	40%	6.7 km/h
12 - Oct Sun	100%	25%	4.4 km/h

Forecast for Today (08 - Oct)

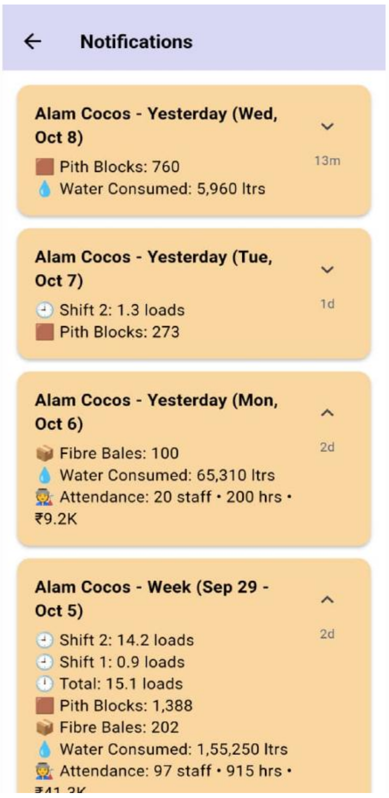
09:00 AM	10:00 AM	11:00 AM
70%	100%	100%
2.2 km/h	1.3 km/h	1.1 km/h
5%	5%	10%

Screen 3.6 - Attendance, Expenses and Weather



Settings

- Staffs
- Projects
- Shift Settings
- Custom Settings
- Pith Locations
- Pith Units
- Share App
- About



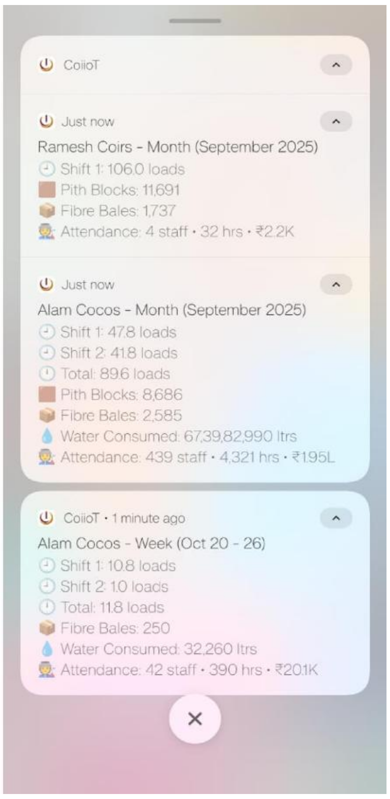
Notifications

Alam Cocos - Yesterday (Wed, Oct 8) — 13m
🟫 Pith Blocks: 760
💧 Water Consumed: 5,960 ltrs

Alam Cocos - Yesterday (Tue, Oct 7) — 1d
🕘 Shift 2: 1.3 loads
🟫 Pith Blocks: 273

Alam Cocos - Yesterday (Mon, Oct 6) — 2d
📦 Fibre Bales: 100
💧 Water Consumed: 65,310 ltrs
👷 Attendance: 20 staff • 200 hrs • ₹9.2K

Alam Cocos - Week (Sep 29 - Oct 5) — 2d
🕘 Shift 2: 14.2 loads
🕘 Shift 1: 0.9 loads
🕛 Total: 15.1 loads
🟫 Pith Blocks: 1,388
📦 Fibre Bales: 202
💧 Water Consumed: 1,55,250 ltrs
👷 Attendance: 97 staff • 915 hrs • ₹41.3K



CoiioT

Just now
Ramesh Coirs - Month (September 2025)
🕘 Shift 1: 106.0 loads
🟫 Pith Blocks: 11,691
📦 Fibre Bales: 1,737
👷 Attendance: 4 staff • 32 hrs • ₹2.2K

Just now
Alam Cocos - Month (September 2025)
🕘 Shift 1: 47.8 loads
🕘 Shift 2: 41.8 loads
🕛 Total: 89.6 loads
🟫 Pith Blocks: 8,686
📦 Fibre Bales: 2,585
💧 Water Consumed: 67,39,82,990 ltrs
👷 Attendance: 439 staff • 4,321 hrs • ₹1.95L

CoiioT • 1 minute ago
Alam Cocos - Week (Oct 20 - 26)
🕘 Shift 1: 10.8 loads
🕘 Shift 2: 1.0 loads
🕛 Total: 11.8 loads
📦 Fibre Bales: 250
💧 Water Consumed: 32,260 ltrs
👷 Attendance: 42 staff • 390 hrs • ₹20.1K

Screen 3.7 - Settings and Notifications

3.6 DEPLOYMENT DESIGN

Deployment Design refers to the process of planning and structuring how a software system's components will be distributed across hardware environments. It focuses on ensuring efficient allocation of resources, seamless communication between components, and optimal performance. This design phase outlines the configuration, placement, and interaction of software modules within the network infrastructure.

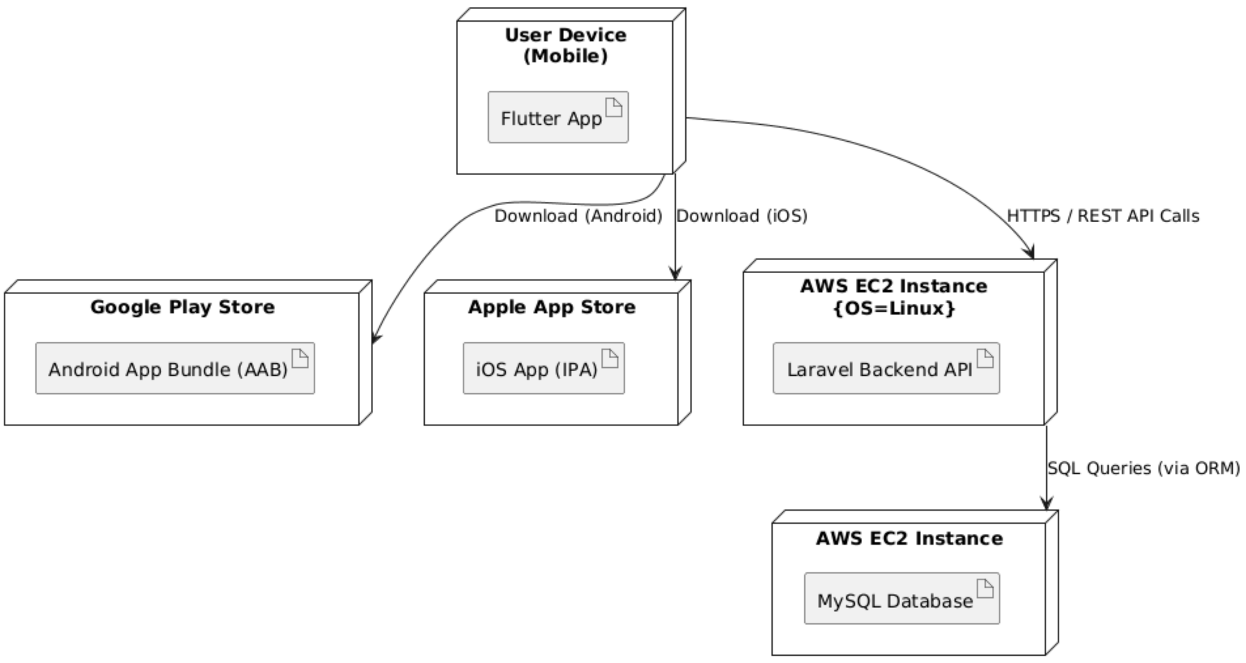


Figure 3.7 Deployment Diagram of CoiioT

The deployment diagram presented in **Figure 3.7** illustrates the deployment architecture of the CoiioT system, showcasing how the application's mobile, backend, and database components interact across different environments. The User Device (Mobile) node represents the Flutter-based mobile application, which serves as the primary interface for users to access production insights, reports, and control features. The application is distributed through both the Google Play Store for Android (as an Android App Bundle) and the Apple App Store for iOS (as an IPA file), ensuring cross-platform availability.

At the backend, the AWS EC2 Instance (Linux) hosts the Laravel Backend API, which manages business logic, authentication, and communication with the database. The mobile application interacts with this API over secure HTTPS/REST API calls, enabling real-time data exchange and operational monitoring.

The MySQL Database, also hosted on an AWS EC2 Instance, handles persistent data storage for production data, user records, and analytics. Laravel communicates with MySQL through its built-in Eloquent ORM, ensuring efficient query execution and seamless data management.

Overall, this deployment configuration provides a scalable, cloud-based architecture that integrates mobile accessibility, backend processing, and secure database management, ensuring high performance, reliability, and maintainability of the CoiioT system.

CHAPTER IV

SYSTEM TESTING

System testing is a critical phase in the software development lifecycle that focuses on validating the complete and integrated software system to ensure it meets the specified requirements. This testing level evaluates the end-to-end functionality of the system as a whole, examining aspects such as performance, usability, and security. It includes a range of testing techniques such as functional testing, performance testing, and regression testing to verify that all modules and features work as expected. By running comprehensive system tests, developers can identify and address defects before deployment, minimizing potential risks and improving overall system stability.

In the context of the CoiioT project, system testing plays a crucial role due to its integration of IoT devices, cloud-based data storage, and real-time analytics dashboards for monitoring pith and fibre production. Through rigorous system testing, CoiioT ensures that all interconnected modules – such as IoT data acquisition, production tracking, attendance and expense management, and role-based user access – work seamlessly together. This testing process validates the end-to-end workflow, from sensor data collection and data processing to dashboard visualization and automated notifications.

By verifying these integrated components, system testing confirms that production insights, employees attendance records, and automated daily summaries are accurately displayed and reliably updated in real time. Ultimately, this ensures that CoiioT provides dependable and data-driven decision support for factory administrators and operators, improving operational efficiency and system reliability across all modules.

4.1 TEST CASES AND REPORTS

Test cases and reports are critical for validating CoiioT system's functionality and performance. Test cases define inputs, conditions, and expected outcomes for key features, like data collection and anomaly detection, ensuring each part works as intended.

Table 4.1 List of Test case Scenarios

TEST CASE ID	TEST CASE SCENARIO	TEST STEPS	EXPECTED OUTCOME	RESULT
TC 01	Login with Valid credentials	1. Enter valid phone number. 2. Enter correct password. 3. Click “Login”.	User is successfully logged in and redirected to the dashboard.	PASS
TC 02	Login with Invalid credentials	1. Enter invalid phone number or password 2. Click “Login”.	Login is rejected, and an error message is displayed.	PASS
TC 03	View Dashboard Data	1. Login 2.Navigate to dashboard	Dashboard displays overall metrics, production charts, and system status correctly.	PASS
TC 04	Display IoT Insights	1.Login 2.Navigate to IoT Insights	IoT device data metrics for pith and fibre are displayed correctly.	PASS
TC 05	Scheduler Job Execution	1. Wait for scheduled data sync job that runs daily. 2. Observe updates in dashboard.	Scheduler runs automatically at the defined time and updates data accurately.	PASS
TC 06	Delete Duplicate Records	1. Run duplicate removal job. 2. Check database records.	Duplicate records are detected and deleted successfully.	PASS

TC 07	Manage Expenses records	1. Open Expense module. 2. Add, edit, or delete expense records.	Expense records are correctly added, modified, or deleted, reflecting in totals.	PASS
TC 08	Manage Employee Attendance	1. Mark the no of hours worked and the wages for employees. 2. Save and verify record.	Attendance entries save successfully and reflect in the calendar view.	PASS
TC 09	Inventory Management – Pith Production Stock	1. Open Inventory module. 2. Add and update stock details.	Inventory displays accurate totals for produced, offloaded, and taken quantities.	PASS
TC 10	Inventory Location-Wise Details	1. Select a location in inventory view. 2. Review records.	Location-based inventory data is correctly displayed.	PASS
TC 11	Role-Based Access Control	1. Log in as SuperAdmin, Admin and staff. 2. Attempt to access restricted pages.	Each user only accesses modules permitted by their role.	PASS
TC 12	Automated Daily Summary Notification	1. Wait for scheduler or trigger summary manually. 2. Check notification inbox.	Daily summary notification with production metrics is sent successfully.	PASS

CHAPTER V

SYSTEM IMPLEMENTATION

Implementation is the process of converting a new or a revised system design into an operational one. It is the most crucial stage in achieving a new successful system and in giving confidence to the new system for the teams that it will work efficiently and effectively. In this phase, one can build the components either from scratch or by composition. Given the architecture document from the design phase and requirement document from the analysis phase, one can build exactly what has been requested.

5.1 IMPLEMENTATION PROCEDURE

1. **IoT Device Integration**:
   - Configured IoT devices to transmit operational data periodically using the MQTT protocol, ensuring lightweight and reliable communication.
   - Devices connect to an MQTT Broker hosted on an AWS EC2 server, enabling real-time data ingestion into the backend system.
   - The gateway ensures secure and continuous data flow from field-level sensors to the cloud infrastructure.
2. **Backend Services with Laravel:**
   - Established the core backend using the Laravel framework, which handles user authentication, API requests, and business logic.
   - Implemented Laravel Services to process incoming device data and manage system operations.
   - Used MySQL as the primary data store, with Laravel's Eloquent ORM facilitating smooth interaction between application logic and database records.
   - Scheduled Cron Scripts on the EC2 server to automate data processing, cleanup routines, and periodic tasks.

3. **Device Management and Notification System:**
   - Integrated device settings and control modules accessible only to SuperAdmin, ensuring secure configuration and oversight.
   - Configured Cron Jobs to monitor device status, trigger alerts, and maintain system health.
   - Upon detecting irregularities or missing data, the system notifies relevant users through push or email notifications.
4. **Frontend Development with Flutter**:
   - Developed a responsive frontend using Flutter, providing Admin and Staff users with dashboards to view production summaries, resource usages, and system metrics.
   - Integrated real-time data visualization with secure, role-based CRUD operations for managing production data.
   - The interface supports intuitive navigation, enabling users to manage inventory, attendance, expense and system settings based on their permissions.
5. **System Testing and Deployment**:
   - Conducted thorough testing using Postman for API validation.
   - Deployed the system on AWS EC2, ensuring scalability and maintainability across services.
   - Role-based access control was validated to ensure that SuperAdmin, Admin, and Staff users could only access permitted functionalities, maintaining system integrity and operational clarity.

CHAPTER VI

CONCLUSION

The CoiioT project has successfully implemented a comprehensive IoT-based monitoring system designed for real-time production tracking and operational management in the coir industry. It integrates IoT sensors, cloud-based storage, and intelligent dashboards to efficiently monitor pith and fibre production, machine states, water consumption, and expenses. Built using PHP Laravel for backend management, Flutter for a responsive user interface, and Laravel Queues and Scheduler Jobs for automated operations, the system ensures seamless data flow from sensors to dashboards. By providing real-time visibility, automated summary generation, and role-based access control (RBAC) for various user levels, CoiioT enhances productivity, transparency, and decision-making efficiency. Its modular architecture delivers a unified, reliable, and user-friendly platform that empowers administrators, managers, and operators to make informed, data-driven decisions and optimize resource utilization across factory operations.

6.1 FUTURE ENHANCEMENTS

1. **GPS-based Vehicle Tracking System:** Implement real-time GPS tracking for factory and transport vehicles using IoT telemetry data and display live vehicle positions on an interactive map for logistics monitoring.
2. **Integration of AI-based Production Forecasting:** Use machine learning models to analyze historical production data and predict future pith and fibre outputs. This will help management in planning raw material requirements, optimizing shifts, and achieving consistent productivity.
3. **Data Export:** Allow users to export production and expense records in Excel or PDF formats, filtered by date or team. Include options to export reports directly to stakeholders or download them from the reports for offline use

BIBLIOGRAPHY

[1] Laravel: Up & Running by Matt Stauffer. O'Reilly Media, 2019.

[2] Understanding Android Push Notifications: A Developer's Complete Guide by Abhinay. Medium, 2025. https://medium.com/codetodeploy/understanding-android-push-notifications-a-developers-complete-guide-9a1ddbfbba7c

WEB REFERENCE

[1] https://flutter.dev/ - Frontend (Flutter)

[2] https://laravel.com/ - Backend (Laravel)

[3] https://www.mysql.com/ - Database (MySQL)

[4] https://www.twilio.com/docs/ - Twilio (Whatsapp OTP Integration)

[5] https://docs.tomorrow.io/reference/weather-forecast/ - Tomorrow.io (Weather Integration)