**School of Mathematical and Computer Sciences**

**COURSEWORK GROUP REPORT**

**ON**

**DEVS PHASE**

|  |  |
| --- | --- |
| Group 6 Members - PG | |
| Name | ID Number |
| Sugashini Kaliappan | H00472677 |
| Fadhil Ashraf | H00462367 |
| Rishab Sajith Kumar | H00450730 |
| Tressa Poulose Arikkadan | H00448683 |

Github Link: <https://github.com/SugashiniKaliappan/bee_hive>

Confluence:<https://sugashinikaliappan.atlassian.net/wiki/spaces/PP/pages/98543/Bee+Hive+-+Project+plan>

JIRA: <https://sugashinikaliappan.atlassian.net/jira/software/projects/BHW/boards/3/backlog>

A POSTMAN export file with the username/password and tests for all route  
<https://app.getpostman.com/join-team?invite_code=3884e759166ba132838238225c36d4ff39974554f6e718ff7cc6af1f4c91f9ba&target_code=2f0e80af388b652faac64c9f8aee5e9a>

**Date:** 28/02/2025

**Course Title:** Applied Development and Operations

**Course Code:** F21AO

**Advisor:** Prof. Abrar Ullah

**Year of Study:** 2025-2026

Table of Contents

[1. Introduction: 3](#_Toc191668043)

[2. System Requirement and functional modules: 3](#_Toc191668044)

[Modules of the System: 4](#_Toc191668045)

[3. System Architecture & Design 5](#_Toc191668046)

[High-Level System Design 5](#_Toc191668047)

[Database Schema Overview 5](#_Toc191668048)

[4. Development Methodology 6](#_Toc191668049)

[Agile Scrum Methodology : 6](#_Toc191668050)

[5. Change Management Strategy and Implementation 7](#_Toc191668051)

[6. Steps to Execute the modules: 8](#_Toc191668052)

[Post man screenshots: 9](#_Toc191668053)

[7. Conclusion 14](#_Toc191668054)

[8. Appendix: 14](#_Toc191668055)

[Jira screenshots: 14](#_Toc191668056)

[10. Reference 15](#_Toc191668057)

# **Introduction:**

The problem domain provided for the coursework was to develop a Hospital Management System for a tertiary care hospital. The system is designed to streamline patient registration, ward admissions, diagnostic services, treatment plans, and referrals while integrating with regulatory bodies and medical research institutions. The hospital operates through A&E, OPD, and direct ward admissions, managing 36 wards across nine departments, including ICU and CCU. It also offers key diagnostic and treatment services like Radiology, Pathology, and Surgery. The HMS will ensure secure authentication, data integration, and efficient patient tracking, following the Agile methodology with DevOps automation.

The project follows the Agile development methodology, designed to streamline patient registration, admissions, treatment, and discharge processes through a modular and API-driven approach. This system will ensure secure and efficient data handling using Node.js for the backend and MongoDB for data storage. The authentication module is mandatory, ensuring that staff members such as doctors, nurses, paramedics, and registration clerks access the system based on defined roles and permissions. This report outlines the system architecture, design considerations, change management strategy, deployment steps, and continuous development practices.

# **2. System Requirement and functional modules:**

The architecture follows a microservices-based design with a RESTful API structure to facilitate modularity, scalability, and maintainability. It consists of a backend-centric, modular architecture that enables smooth interaction between authentication, patient management, and treatment tracking. The technology stack used for this project includes Node.js (Express.js) for the backend, with MongoDB as the database, managed efficiently using Prisma ORM. Prisma simplifies database interactions, provides type safety, and streamlines schema management. Authentication for secure role-based access is implemented using JSON Web Token (JWT), while API testing and development are conducted using Postman:

1. Application Layer (Backend - Node.js & Express.js)

* Handles business logic, authentication, and API requests.

1. Data Layer (MongoDB with Prisma ORM)

* Stores patient details, admissions, and treatment records.

1. API Interface (Postman for Testing & Development)

* RESTful APIs facilitate communication between different modules.

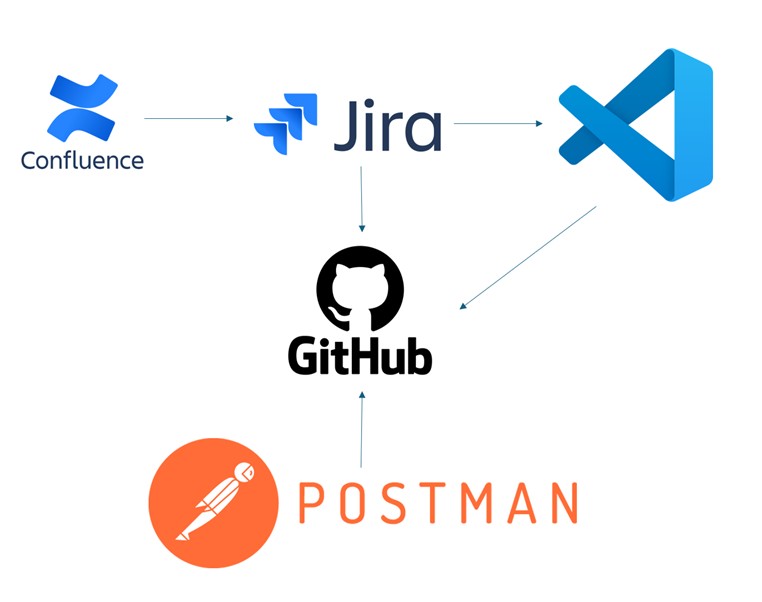


Figure 1 System architecture

## Modules of the System:

1. **User Authentication Module**: Manages user authentication and role-based access control. Ensures secure access using **JWT-based authentication**. Secure password storage using **bcrypt**.
2. **Initial Patient Registration Module**: Handles patient registration at different service points. Registers and stores patient details in **MongoDB**. Assigns a **unique Patient ID** to each record. Provides API endpoints for patient data retrieval and updates.
3. **Ward Admissions Module**: Manages inpatient admissions, including bed allocation and patient monitoring. Assigns patients to wards based on availability. Tracks admission and discharge status.
4. **Patient Treatment Module**: Records and manages patient treatment, prescriptions, and progress updates. Doctors and nurses update patient records via API endpoints. Stores medical history and generates discharge summaries.

# **3. System Architecture & Design**

## High-Level System Design

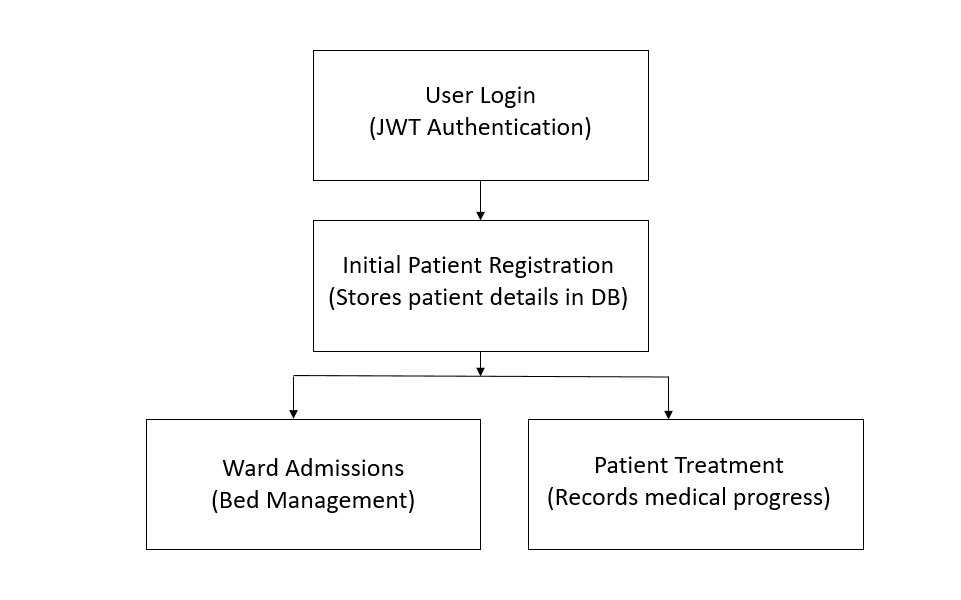


Figure 2 High level System Design

## Database Schema Overview

List of database:

In the development phase of our DevOps project, we are using Postman to send data to an API built using NodeJS and ExpressJS. This data is further stored into MongoDB across different tables depending on the kind of API used:

1. "d" table : for testing purposes
2. "Patients" : for storing patient information
3. "staffDetails" : for managing staff records
4. "wards" : for tracking ward details where patients are referred to

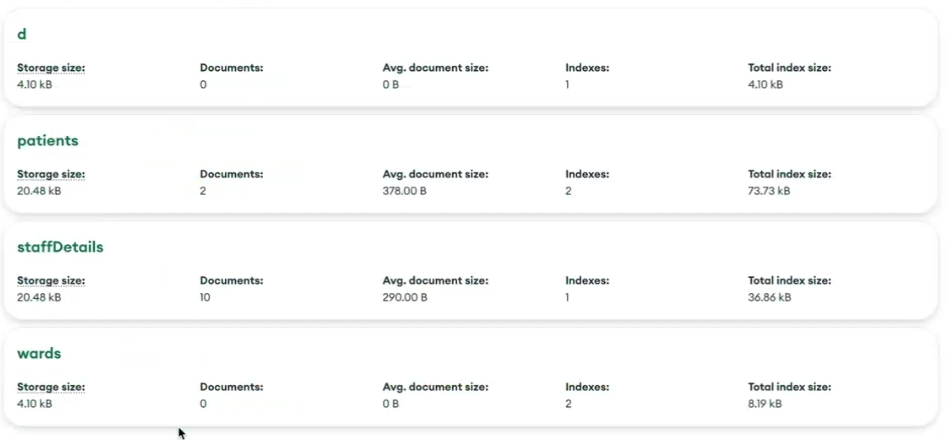


Figure 3 MongoDB schemas

# 4. Development Methodology

## Agile Scrum Methodology :

#### For the Hospital Management System (HMS), we adopted the Agile scrum methodology due to its adaptability, iterative development approach, and focus on continuous delivery. Given the complexity of managing hospital workflows—ranging from patient registration and ward admissions to treatment tracking—Agile ensures that development remains responsive to evolving requirements and stakeholder feedback.

#### The project was divided into weekly sprints, each focusing on specific modules (e.g., authentication, admissions, treatment tracking). Each sprint included feature development, testing, deployment, and stakeholder feedback collection. The sprints ended with sprint review, necessary improvements or feature enhancements were added to the backlog for future sprints. We used JIRA for tracking feature requests, bugs, and sprint progress. Tasks were assigned to developers based on priority and complexity. GitHub was used for Version control and code reviews. GitHub was used for branch-based development, ensuring isolated changes before merging. This ensured codes were reviewed and pull requests prevented bugs from reaching the production. We used the following testing strategy: Unit testing to validate individual functions, Integration Tests to ensure different system modules work correctly and End-to-End Testing to simulate real-world user interactions using Postman.

#### 

Figure 4 Unit testing example

# **5. Change Management Strategy and Implementation**

The Change Management strategy manages the people side of change. It is helpful for an organization to plan for change, control the change process, and effectively implement change. For minimizing resistance to change and achieving the benefits of new strategies, processes, and technologies, this approach tends to be crucial.

The objective of change management is includes:

1. Streamlining Change Requests
2. Aligning Changes with Business Goals
3. Effective Communication
4. Efficiency in Minor Changes

The key components of the change management process are:

1. Change identification
2. Change request management
3. Change control board
4. Change log
5. Change implementation

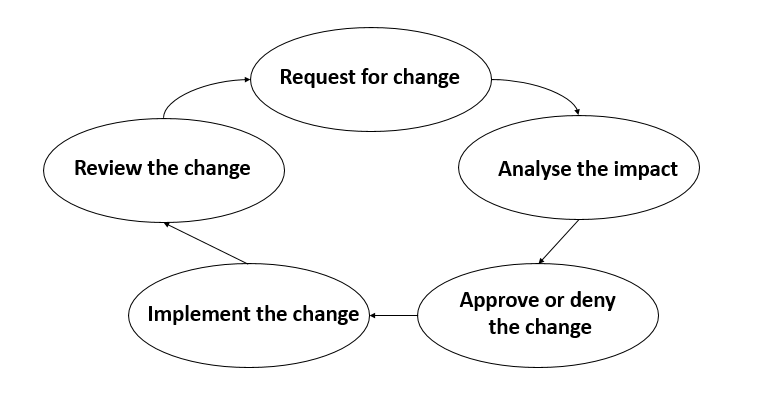


Figure 5 Steps for effective change management

# 6. Steps to Execute the modules:

#### **1: Extract code zip.file**

#### **2: User Authentication Module :**

#### **Change the directory to user\_authentication**

#### **Use command npm install followed by npm start**

#### **Connection to MongoDB will be established using port 5000**

#### **Following link can be used for user authentication** <http://localhost:5000/api/v1/auth/register>

#### 3: Lab referrals Micro-Service:

#### C**hange the directory to modules/lab\_results**

#### **Use command npm install followed by npm start**

#### **Connection to MongoDB will be established using port 5001**

#### **Following link can be used for user authentication** <http://localhost:5003/api/v1/labtreatment/entry/treatment>

#### 4: Patient Registration Micro-Service:

#### **Change the directory to modules/patient\_enrolment**

#### **Use command npm install followed by npm start**

#### **Connection to MongoDB will be established using port 5002**

#### **Following link can be used for user authentication** <http://localhost:5002/api/v1/patient/register>

#### 5: Ward Admission Micro-Service:

#### **Change the directory to modules/ward\_report**

#### **Use command npm install followed by npm start**

#### **Connection to MongoDB will be established using port 5003**

#### **Following link can be used for user authentication** <http://localhost:5001/api/v1/wardadmissions/wards>

## **Post man screenshots:**

#### ****Login****

#### 

Figure 6 User Login

#### ****Create new user and assigning role****

#### 

Figure 7 creating a new user and assigning role

#### ****Error on duplicate entry****

#### 

Figure 8 Error on duplicate entry

#### ****Creating a patient****

#### 

Figure 9 Creating a patient record

#### ****Retrieving patient details****

#### 

Figure 10 Retrieving a patient record

#### ****Referring patient to lab****

#### 

Figure 11 Referring a patient to a lab

#### ****Ward availability****

#### 

Figure 12 Ward **Availability**

#### ****Assigning patient to ward****

#### 

Figure 13 Assigning patient to ward

#### ****Retrieving ward details by id****

#### 

Figure 14 Retrieving ward details by id

# 7. Conclusion

The goal of developing the Hospital Management System is to improve efficiency of patient registration, ward admissions, tracking of treatments and general tasks of a hospital. We have utilized microservices based design and technologies like Node.js and MongoDB to ensure safe management of data and to process data efficiently. By following the Agile Scrum methodology, we were able to develop iteratively with continuous testing as well. Version control through GitHub and API testing with Postman helped us to guarantee code dependability and system integrity. All things considered, the project has provided useful information about API-driven system architecture and software development methodologies in its implementation so far.

# 8. Appendix:

## **Jira screenshot**s**:**

#### 

Figure 15 Jira Dashboard

#### 

Figure 16 Jira Backlog

## **Confluence Screenshot**s**:**

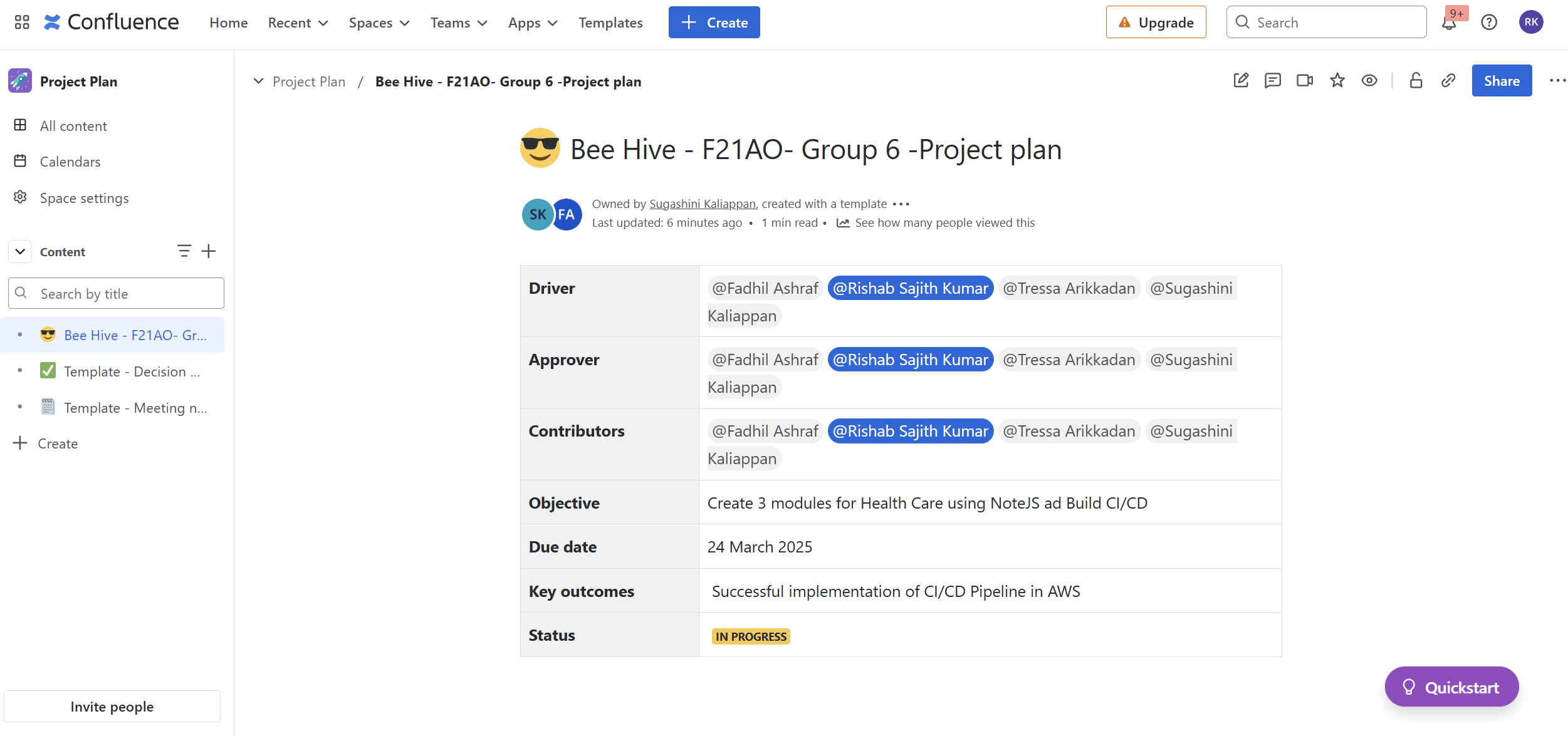


Figure 17 : Confluence Page

# 10. Reference

#### <https://www.4pmti.com/learn/change-management-process-guide/>

#### **Dr. Abrar’s notes and course materials from Herriot Watt University, Dubai**

#### <https://medium.com/@anandam00/build-a-secure-authentication-system-with-nodejs-and-mongodb-58accdeb5144>

#### **https://www.youtube.com/watch?v=A1gm6z1d4Tg&t=9s**