IOT-BASED HEALTH MONITORING SYSTEM

A PROJECT REPORT

Submitted by

Subramanya Trishank Reddy M[RA2211056010002]
Rishyunth Raju S R V[RA2211056010044]

Under the Guidance of

Dr. D.Prabakar

Assistant Professor
Department of Data Science and Business Systems

in partial fulfillment of the requirementsfor the degree of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE ENGINEERING with specialization in DATA SCIENCE

DEPARTMENT OF
DATA SCIENCE AND BUSINESS SYSTEMS
SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR- 603 203

MAY 2025

Department of Data Science And Bussiness

SRM Institute of Science & Technology

Own Work* Declaration Form

This sheet must be filled in (each box ticked to show that the condition has been met). It must be signed and dated along with your student registration number and included with all assignments you submit – work will not be marked unless this is done.

To be completed by the student for all assessments

Degree/ Course : B-TECH/ CSE(Data Science)

Student Name : Subramanya Trishank Reddy M, Rishyunth Raju S R V

Registration Number : RA2211056010002, RA2211056010044

Title of Work : IOT-BASED HEALTH MONITORING SYSTEM

We hereby certify that this assessment compiles with the University's Rules and Regulations relating to Academic misconduct and plagiarism**, as listed in the University Website, Regulations, and the Education Committee guidelines.

We confirm that all the work contained in this assessment is my / our own except where indicated, and that I / We have met the following conditions:

- Clearly referenced / listed all sources as appropriate
- Referenced and put in inverted commas all quoted text (from books, web, etc)
- Given the sources of all pictures, data etc. that are not my own
- Not made any use of the report(s) or essay(s) of any other student(s) either past or present
- Acknowledged in appropriate places any help that I have received from others (e.g. fellow students, technicians, statisticians, external sources)
- Compiled with any other plagiarism criteria specified in the Course handbook / University website

I understand that any false claim for this work will be penalized in accordance with the University policies and regulations.

DECLARATION:	
We are aware of and understand the University's policy on Academic misconduct and plagiarism and we certify that this assessment is our own work, except where indicated by referring, and that we have followed the good academic practices noted above.	
SUBRAMANAYA TRISHANK REDDY M RA2211056010002	RISHYUNTH RAJU S R V RA2211056010044

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR – 603 203

BONAFIDE CERTIFICATE

Certified that 21CSP302L-Minor Projet report titled"**IOT BASED HEALTH MONITORING SYSTEM**" is the bonafide work of **"SUBRAMANYA TRISHANK REDDY M [RA2211056010002], RISHYUNTH RAJU S R V [RA2211056010044]"** who carried out the project work[internship] under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE
Dr. D. Prabakar
SUPERVISOR
ASSISTANT PROFESSOR
DEPARTMENT OF DATA SCIENCE AND BUSINESS SYSTEMS

SIGNATURE
Dr. V. Kavitha
PROFESSOR &HEAD
DEPARTMENT OF DATA SCIENCE AND BUSINESS SYSTEMS

ACKNOWLEDGEMENTS

We express our humble gratitude to **Dr. C. Muthamizhchelvan**, Vice-Chancellor, SRM Institute of Science and Technology, for the facilities extended for the project work and his continued support.

We extend our sincere thanks to **Dr. Leenus Jesu Martin M,** Dean-CET, SRM Institute of Science and Technology, for his invaluable support.

We wish to thank **Dr. Revathi Venkataraman**, Professor and Chairperson, School of Computing, SRM Institute of Science and Technology, for her support throughout the project work.

We encompass our sincere thanks to, **Dr. M. Pushpalatha**, Professor and Associate Chairperson - CS, School of Computing and **Dr. Lakshmi,** Professor and Associate Chairperson -AI, School of Computing, SRM Institute of Science and Technology, for their invaluable support.

We are incredibly grateful to our Head of the Department**, Dr Kavitha V**, Professor and Head, Data Science And Business Systems, SRM Institute of Science and Technology, for her suggestions and encouragement at all the stages of the project work.

We want to convey our thanks to our Project Coordinators, Panel Head, and Panel Members Department of Data Science And Business Systems, SRM Institute of Science and Technology, for their inputs during the project reviews and support.

We register our immeasurable thanks to our Faculty Advisor**, Dr M Anand**, Assistant Professor Department of Data Science And Business Systems, SRM Institute of Science and Technology, for leading and helping us to complete our course.

Our inexpressible respect and thanks to our guide, **Dr D Prabakar**, Assistant Professor Department of Data Science And Business Systems, SRM Institute of Science and Technology, for providing us with an opportunity to pursue our project under his mentorship. He provided us with the freedom and support to explore the research topics of our interest. His passion for solving problems and making a difference in the world has always been inspiring.

We sincerely thank all the staff members of Department of Data Science And Business Systems, School of Computing, SRM Institute of Science and Technology, for their help during our project. Finally, we would like to thank our parents, family members, and friends for their unconditional love, constant support and encouragement

SUBRAMANYA TRISHANK REDDY M[RA221105601002]

RISHYUNTH RAJU S R V[RA2211056010044]

ABSTRACT

The **IoT-Based Health Monitoring System** project is designed to bridge the gap between patients and healthcare providers through a digital platform that simplifies appointment scheduling, health data monitoring, and profile management. This document details the project's development journey, structured across five organized sprints under Agile methodology, ensuring rapid delivery, adaptability, and quality assurance. Sprint 1 focused on building the core authentication system, enabling secure registration and login functionalities for both doctors and patients, with proper database integration. Sprint 2 developed user dashboards tailored to each role, providing intuitive navigation and access to key services. Sprint 3 introduced the management of health metrics, including the collection and storage of real-time biometric data from devices such as the ESP32 sensor, enhancing the platform's ability to support remote health monitoring. Sprint 4 implemented the appointment booking functionality, allowing patients to request consultations and doctors to manage their schedules effectively. Finally, Sprint 5 concentrated on improving the overall user interface design, ensuring a responsive and professional look, while comprehensive functional testing was performed to validate the system’s stability and performance. Each sprint documented sprint goals, user stories, system architecture, UI prototypes, functional specifications, testing strategies, daily call progress, and retrospective evaluations. The structured approach adopted throughout the project ensured not only the successful delivery of all committed features but also laid a strong foundation for future scalability, positioning **IoT-Based Health Monitoring System** as a reliable and user-centric healthcare solution.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

Abbreviation	Full Form
UI	**U**ser **I**nterface
API	**A**pplication **P**rogramming **I**nterface
DB	**D**ata **B**ase
ESP32	**E**spressif **S**ystems **M**icrocontroller (IoT Device)
PHP	**H**ypertext **P**reprocessor
HTML	**H**yperText **M**arkup **L**anguage
CSS	**C**ascading **S**tyle **S**heets
SQL	**S**tructured **Q**uery **L**anguage
CRUD	**C**reate, **R**ead, **U**pdate, **D**elete
MVC	**M**odel-**V**iew-**C**ontroller
HTTP	**H**ypertext **T**ransfer **P**rotocol
XAMPP	Cross-platform Apache, MariaDB, PHP, and Perl
QA	**Q**uality **A**ssurance
UX	**U**ser **E**xperience
SRS	**S**oftware **R**equirements **S**pecification

CHAPTER 1

INTRODUCTION

1.1 Introduction to IoT-Based Health Monitoring System

The manner in which we access healthcare is transforming at a rate of knots, driven by digital technologies. There is now increased emphasis on remote care, real-time feedback, and individualized support between patients and physicians. In this new paradigm, the IoT-Based Health Monitoring System emerges as an effective tool building a smart, networked platform where patients can be continuously monitored without having to visit a hospital each time.

At the center of this system are IoT technologies such as the ESP32 microcontroller and biomedical sensors such as the MAX30102. They work together to gather critical health information such as heart rate, oxygen saturation (SpO_2), and body temperature and transmit it securely to healthcare professionals in real time.

We've designed a web application with PHP handling backend computations and MySQL used to store the data. From there, the patient can sign up, sign in, schedule visits, review medical records, and receive input from physicians remotely, all within their homes. Meanwhile, doctors receive real-time data about their patients, are able to see medical trends, and intervene instantly whenever necessary. This way, patients are no longer restricted to time-bound check-ups — early intervention and detection become relatively easier, particularly when it comes to chronic diseases.

We made the system simple to use, scalable, and secure so that anyone of any age and technical skill can take advantage of it. And due to its modular design, it's poised for future enhancements such as adding new sensors or even AI-based health forecasts.

Ultimately, the IoT-Based Health Monitoring System is not merely about technology it's about creating a healthcare experience that's seamless, reliable, and accessible, in the process creating a healthier and more connected world.

1.2 Motivation

Today's healthcare environment is confronted with significant challenges — from the burden of aging populations and long-term diseases to the unpredictability of worldwide pandemics. Conventional models, which rely heavily on in-person consultations and manual monitoring of health, can't always keep pace. In rural communities, individuals often simply struggle to travel to a clinic, while urban hospitals grapple with swaths of people and lengthy wait times. And without ongoing monitoring, many health issues aren't detected until it's too late.

That is where the impetus for our IoT-Based Health Monitoring System derives from. Healthcare ought to be less difficult to attain, more swift to deliver, and even proactive — as opposed to reactionary once something does fail.

By merging IoT devices with a smart digital platform, our system enables one to monitor significant health markers such as heart rate, SpO_2 levels, and body temperature remotely and in real time. Patients can take charge of monitoring their own health at home, while physicians receive precise, current information to intervene early when necessary.

This not only enhances patient outcomes — it minimizes avoidable hospital stays and alleviates the burden on healthcare systems as a whole.

Most of all, it empowers individuals. Rather than waiting until something really feels wrong, individuals can be in control of their own health, detect warning signs early on, and create healthier lives. And since the system operates from a distance, it breaks down geographical barriers, making quality healthcare more accessible to all — regardless of where they reside.

At its heart, this project is propelled by a vision of a world in which technology bridges the gap between doctors and patients — building a smarter, healthier, and more connected world.

1.3 Sustainable Development Goal (SDG) of the Project

Our IoT-Based Health Monitoring System has a direct linkage with the United Nations' Sustainable Development Goal 3: Good Health and Well-Being.

By monitoring key vital signs such as heart rate, temperature, oxygen saturation, and blood pressure around the clock, our system allows for the detection of emerging health problems early on before they become severe. This immediate monitoring allows doctors to react sooner, emergencies to be avoided, and patients to become empowered by managing their own health trends.

For those patients who reside in remote or underserved areas, the system is particularly

dominant. Rather than go a long way for check-ups, they are able to remain in touch with medical professionals from their own homes, conserving time, money, and effort. This further decongests hospitals, making them available for more critical or emergency cases.

In addition, the system is customized. Physicians are able to view the information for each patient over time, providing individualized advice or treatment regimens that can adapt if necessary. This customization results in improved health outcomes and fewer disparities in access to healthcare.

Overall, our project works towards SDG 3 in the sense that it makes health care more preventive, efficient, and inclusive. It's bringing us closer to a world in which everyone no matter their location, earnings, or age can have full control of their health and enjoy healthier lives.

1.4 Product Vision Statement

1.4.1 Audience

Primary Audience:

Our system is primarily intended for patients requiring constant, real-time monitoring of their health particularly individuals with chronic conditions, the elderly, and those residing in remote locations where frequent hospital visits are not always feasible.

Secondary Audience:

It also benefits healthcare professionals such as physicians, nurses, and medical personnel who require immediate, accurate access to patient health information in order to make quicker, more informed decisions regarding care.

1.4.2 Needs

Primary Needs:

Real-Time Health Monitoring:

Patients and doctors need a way to continuously track key vital signs like heart rate, oxygen levels, body temperature, and blood pressure, around the clock.

Early Detection of Health Issues:

Immediate notification should inform patients and health professionals when something abnormal is found so that quick action may be taken before complications arise.**Remote Healthcare Management:**

Patients particularly those who are distant from hospitals ought to receive quality health care from their homes, reducing the frequency of hospital visits.

Secondary Needs:

Easy Access to Data:

Doctors and patients must be able to access prior medical history easily, aiding in the identification of patterns and improved long-term health choices.

Seamless Integration with Hospital Systems:

The platform must integrate well with healthcare systems such as hospital records and electronic health records (EHRs) without requiring significant tweaking.

Simple, User-Friendly Design:

The system should be simple for all to use whether a tech-savvy physician or an older patient so health tracking is easy and hassle-free.

1.4.3 Products

Core Product:

An Intelligent IoT-based health monitoring system that collects vital health information from wearable devices such as smartwatches or sensors and sends it immediately to a secure platform where doctors and patients can monitor it in real time.

Additional Features:

Health Data Insights:

The system 4nalyses data collected to provide useful trends and insights, aiding improved treatment decisions and lifestyle changes.

Emergency Alerts:

In case readings stray perilously off course, instant alerts will be sent to healthcare professionals and emergency contacts.

Remote Monitoring Dashboard:

Physicians can monitor numerous patients simultaneously through a special dashboard, enabling them to offer proactive care without requiring face-to-face consultations each time.

Patient Health History:

A concise, coherent timeline of the patient's health information facilitates easier observation of changes across time and early detection of problems.

1.4 Product Goal

The primary aim of the IoT-Based Health Monitoring System is to develop an intelligent, secure, and simple healthcare platform that leverages the capabilities of IoT technology with advanced healthcare management. Our aim is to develop a system that enhances the experience for patients and healthcare providers alike, rendering healthcare more convenient, proactive, and efficient.

One of the fundamental concepts behind the platform is ongoing health monitoring. Critical signs such as heart rate, oxygen saturation (SpO_2), body temperature, and other critical indicators are monitored in real-time. This allows physicians to detect health problems early and provide timely, customized treatments possibly even before patients are aware that anything's amiss.Security is also high priority. The system employs role-based access control, so that sensitive health data can only be seen by registered users (patients or physicians) to safeguard privacy and maintain conformance with healthcare regulations.

In addition to health tracking, the platform simplifies day-to-day healthcare activities. Patients can schedule, reschedule, or cancel appointments using an automated scheduling system without having to endure long wait times or complex processes. Physicians also gain, with less administrative burden and more time to devote to care.

One major way healthcare becomes simpler is by simple and intuitive data visualization. Doctors and patients can see over time how health trends are progressing using intuitive dashboards simpler to monitor progress, detect warning signs early on, and make more informed treatment decisions.

Another significant objective is ensuring that all patient information, such as electronic health records (EHRs), is kept safely and is available quickly when necessary. This enhances care continuity patients and physicians are able to view the history to see the complete scenario prior to progressing.

The platform's power comes in the form of real-time integration of IoT sensors. Health sensors at home and wearable devices all provide real-time data straight to the platform, providing doctors with current information on which to base their diagnosis and advice rather than working off out-of-date checkup reports.

1.5 Product Backlog (Key User Stories with Desired Outcomes)

Table 1.1 UserStories

S.No	User Stories of IoT-Based Health Monitoring System	Desired Outcome
#US 1	As a patient, I want to securely register and log in so that my medical information remains private.	Data privacy and secure authentication
#US 2	As a doctor, I want to view patient health metrics remotely so that I can monitor conditions proactively.	Remote monitoring
#US 3	As a patient, I want to record and upload my biometric readings using IoT devices.	IoT integration
#US 4	As a patient, I want to book appointments online at my convenience.	Appointment system
#US 5	As a doctor, I want to manage appointment slots and confirm bookings.	Doctor schedule management
#US 6	As an admin, I want to manage user accounts, appointments, and system logs.	Full administrative control
#US 7	As a patient, I want reminders for upcoming appointments.	Improved patient engagement

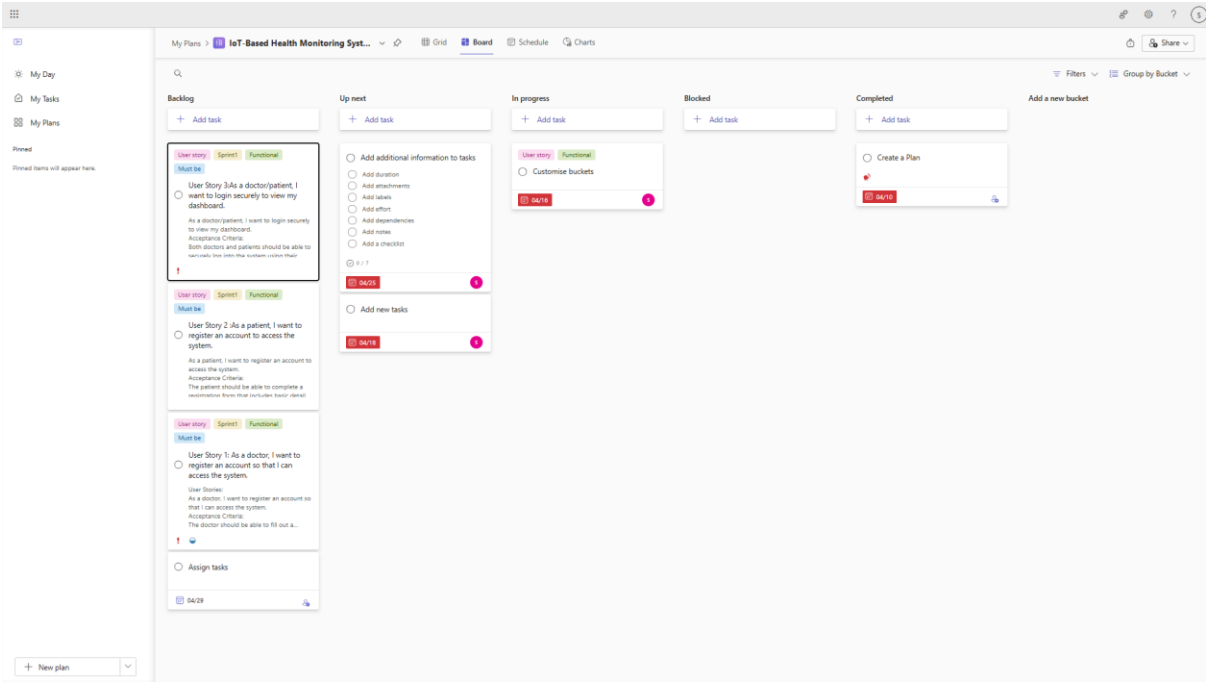

Figure 1.1 MS Planner Board of IoT-Based Health Monitoring System

1.7 Product Release Plan

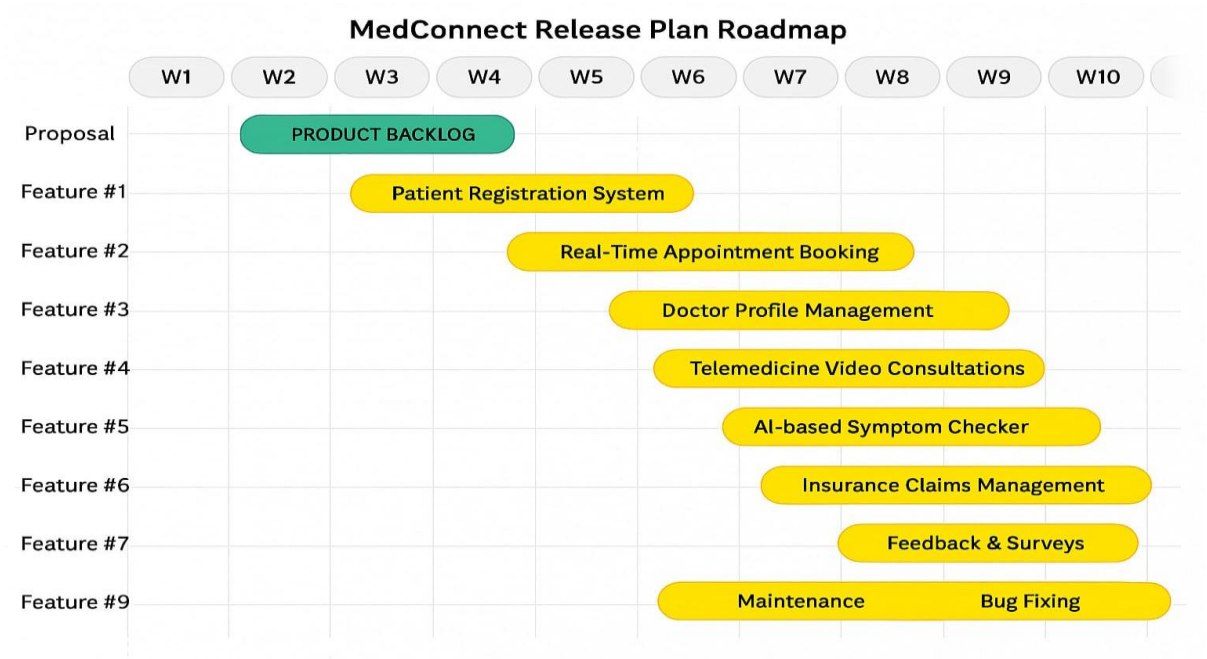

Figure 1.2 Release plan of IoT-Based Health Monitoring System

CHAPTER 2

SPRINT PLANNING AND EXECUTION

2.1 Sprint 1

2.1.1 Sprint Goal

Develop a secure Login, Registration, Update Profile, and Logout system for both doctors and patients in MedConnect.

Table 2.1 Detailed User Stories of sprint 1

ID	As a...	I want to...	So that...	Priority
US1	Patient	Register an account	I can access my dashboard securely	High
US2	Doctor	Register an account	I can monitor patient information	High
US3	Patient	Login to my account	Access my health data safely	High
US4	Doctor	Login to my account	View patient list safely	High
US5	User (Doctor/Patient)	Update my profile	Keep my information accurate	Medium
US6	User (Doctor/Patient)	Logout securely	Protect my account	High

Planner Board representation of user stories are mentioned below figures 2.1,2.2 and 2.3

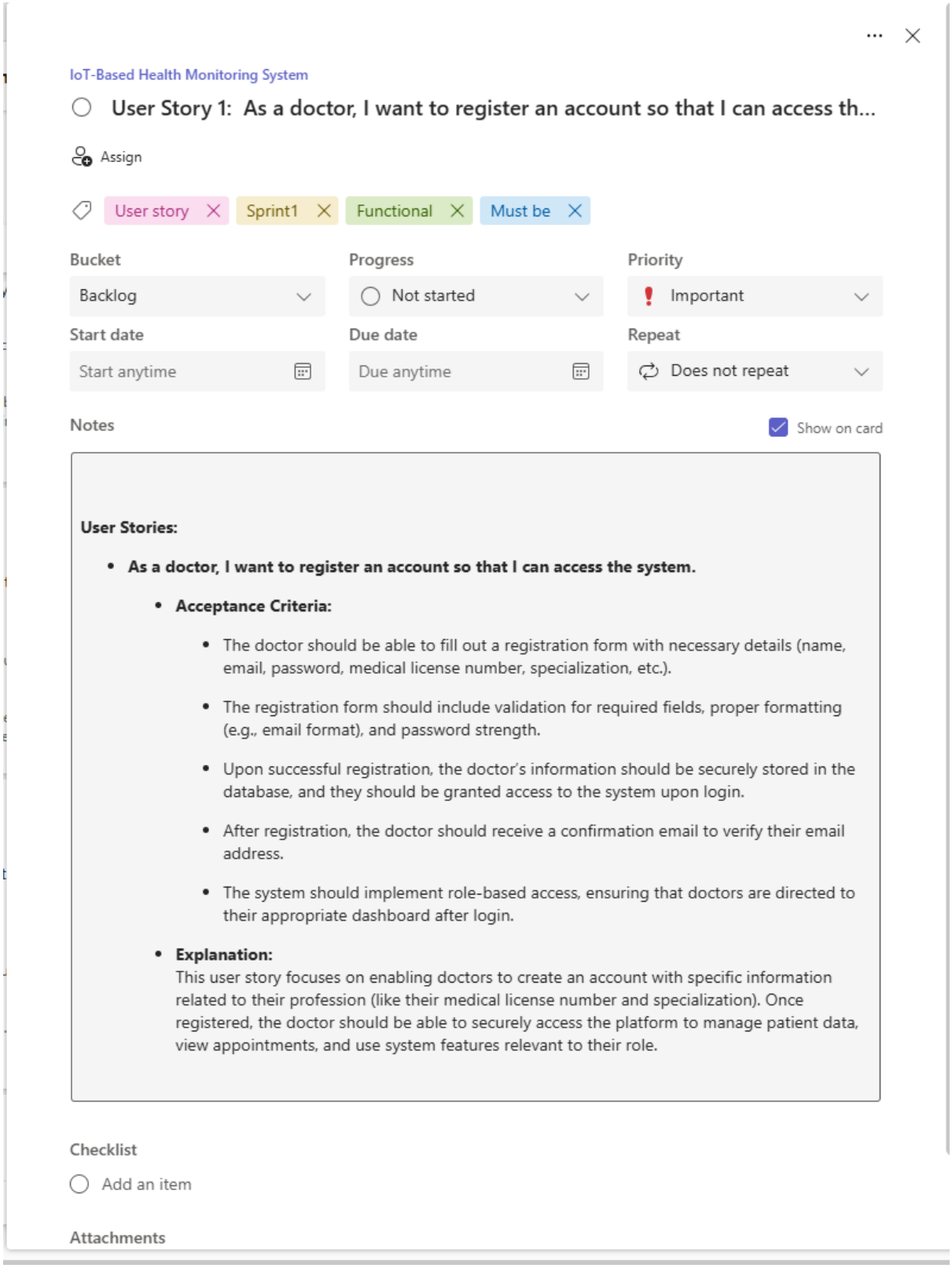

Figure 2.1 user story for user registration(Doctor)

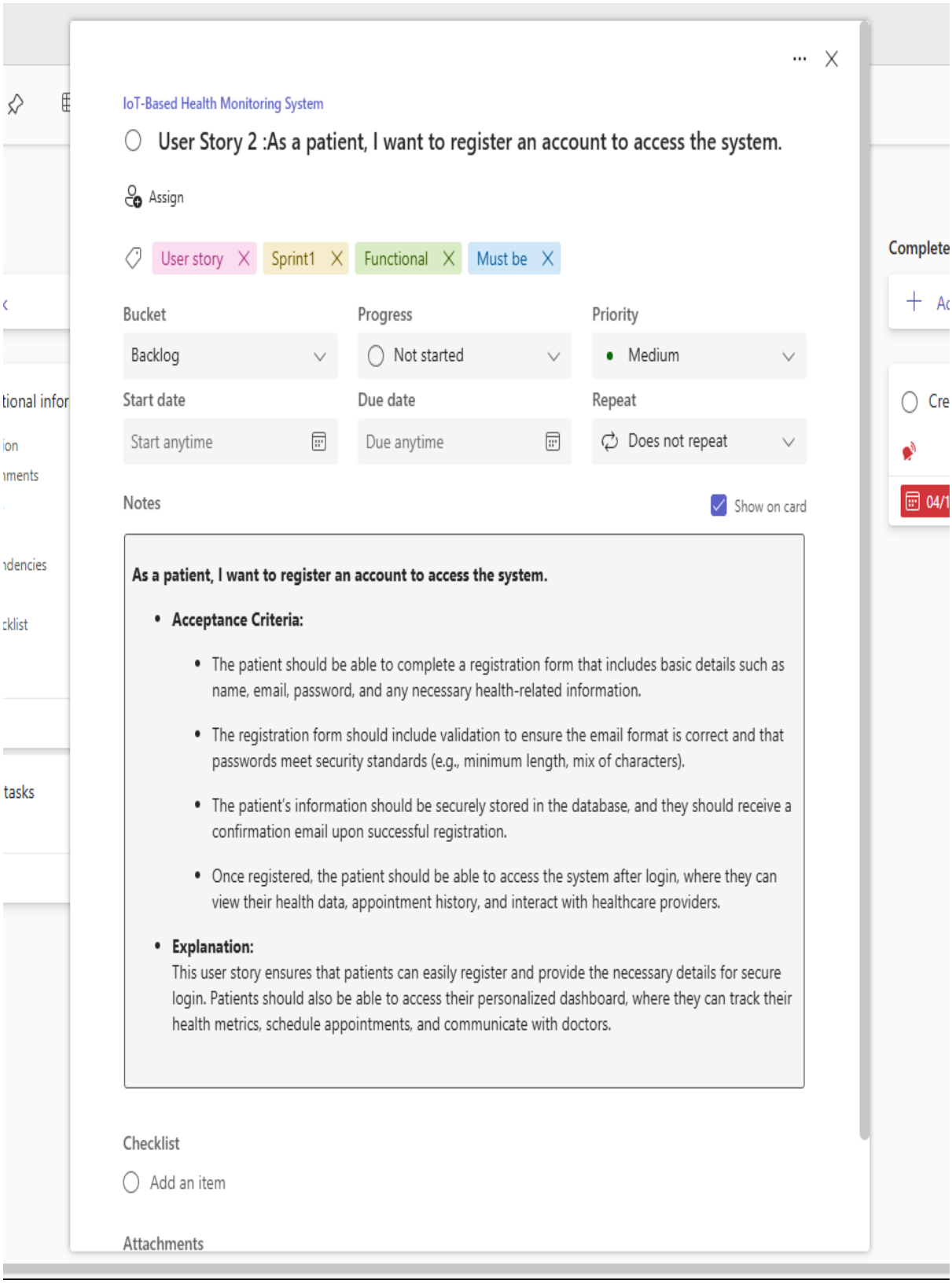

Figure 2.2 user story for user registration(patient)

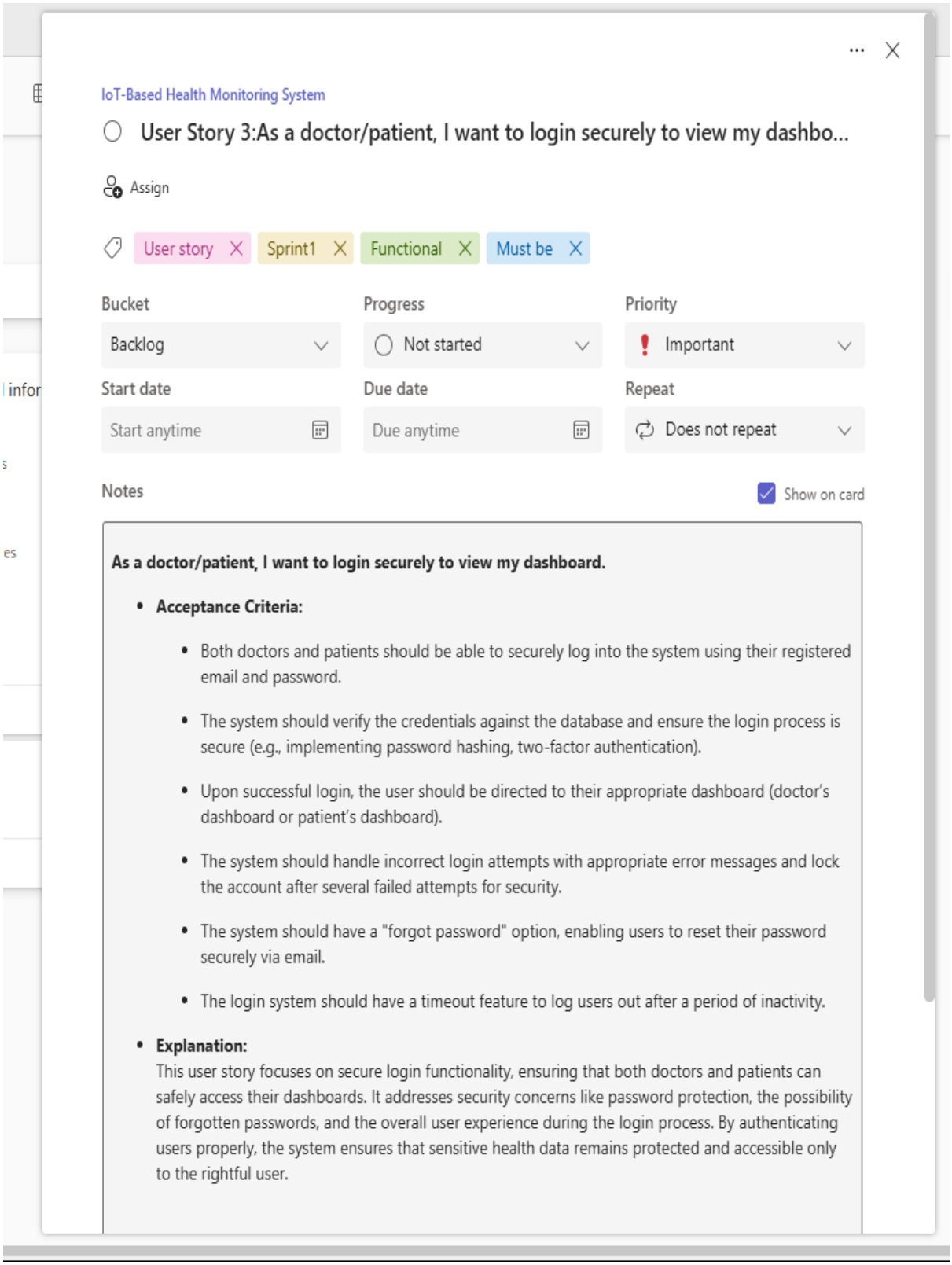

Figure 2.3 User story for login to dashboard

2.1.2 Functional Document

2.1.2.1. Introduction

MedConnect project is a health monitoring system aimed at building a secure and effective interaction between patients and physicians.Sprint 1 is concerned with establishing the central authentication system that allows users (physicians and patients) to securely register, login, profile update, and logout.This platform is important in handling health records, real-time monitoring, and individualized care in future sprints.

2.1.2.2. Product Goal

The main objective of Sprint 1 is to develop a secure profile management and authentication system that supports:

• Safe and frictionless user registration (patient/doctor registration).

• Authenticate user for various roles.

• Simple profile updates to keep up-to-date contact information.

• Implement secure logout functionality to safeguard user sessions and sensitive information.

2.1.2.3. Demography (Users, Location)

Users:

- Target Users:
    - no Patients looking to track their health and interact with physicians.
    - no Physicians handling patient data and health metrics.
- User Characteristics:
    - Patients: Non-tech users requiring easy and straightforward login/registration functionality.
    - Doctors: Professionals who need fast, secure access to patient dashboards.

Location:

- Target Location:
    - Initially India and comparable emerging economies where affordable, accessible digital healthcare platforms are necessary.
    - Subsequently scalable to global audiences with minimal changes.

2.1.2.4. Business Processes

The most important business processes are:

User Registration:

- Patients and doctors register separately by furnishing corresponding personal details.

Profile Management:

- Users can change their email, phone number, and password.
- The data is updated in the database securely after authentication.

User Logout:

- Users can log out, ending the session and maintaining data security.

2.1.2.5. Features

User Registration:

- Patients and doctors register separately by furnishing corresponding personal details.

User Authentication:

- Session management maintains secure and persistent login sessions.

Profile Management:

- It allows users to change their email, phone number, and Name.
- Data is safely updated in the database after validation.

User Logout:

- Users can log out, ending the session and making data secure.

2.1.2.6. Authorization Matrix

Table 2.2 Access level Authorization Matrix

Role	**Access Level**
Administrator	Full access to user data management (future sprint)
Doctor	Access to personal dashboard, patient data (future), and profile updates
Patient	Access to own dashboard, own health data (future), and profile updates
Guest	Only access to Login and Registration pages

2.1.2.7. Assumptions

- Valid and honest registration information will be entered by the users.
- Passwords will be stored safely (encrypted storage implementation scheduled in subsequent sprint).
- Phone numbers and email IDs will be usernames as well as unique identifiers.
- Secure login sessions will be maintained using standard PHP session management.
- Users have good internet connectivity during login/registration activities.
- MySQL server is well configured and accessible for backend storage of data.
- Adherence to general data protection laws such as GDPR will be taken into account in the entire project (subsequent sprints).

2.1.3 Architecture Document

2.1.3.1 Application

Sprint 1 of the Project platform has a modular service-based architecture to maintain authentication, user management, and session handling in separate entities for ease of maintenance and scalability.Though complete microservices are to be implemented in later sprints, Sprint 1 employs a service-oriented modular architecture with an emphasis on:

Key Modules:

Authentication Service:

Handles secure login of users (username & password authentication), creation of sessions, and invalidation of sessions (logout).

Registration Service:

Manages new sign-ups of doctors and patients, securely storing data in the MySQL database.

Profile Management Service:

Permits users to change profile data such as email address, phone number, and password upon successful login.

Session Management Service:

Validates active sessions, avoids unauthorized access, and implements logout feature.

2.1.3.2 System Architecture:

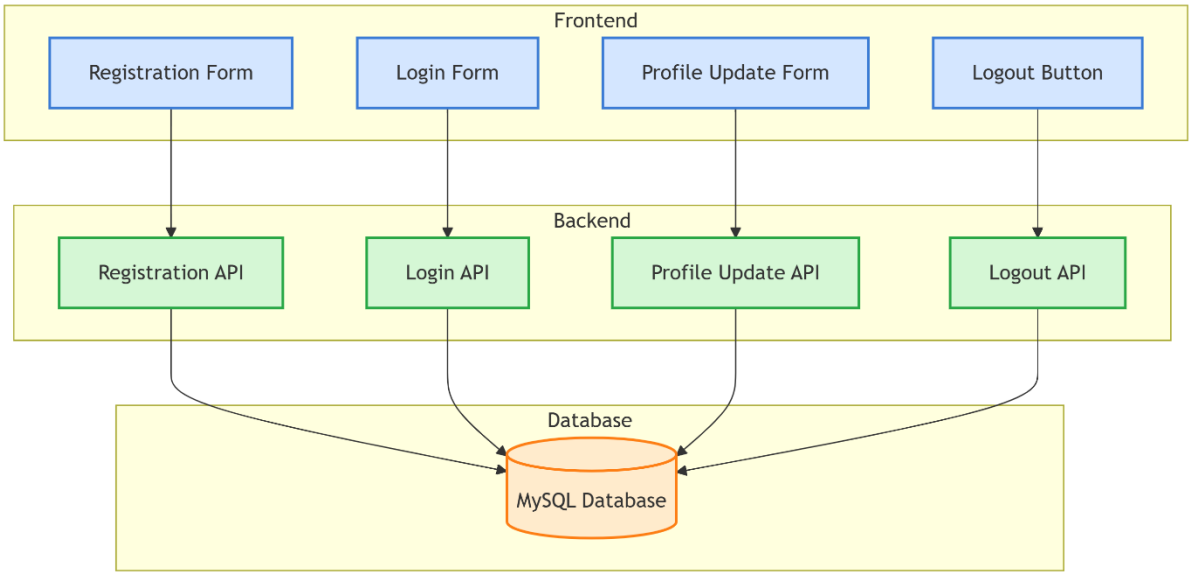

Figure 2.4 System Architecture Diagram

2.1.3.3.

Data Exchange Contract:

Operation	**Frequency**
User Registration	Real-time (API call at form submission)
User Login	Real-time (session validation at each login)
Profile Update	Real-time (instant database update)
User Logout	Real-time (session termination at logout click)

Data Sets:

Data Type	**Fields**	**Use Cases**
User Data (Doctor/Patient)	fullname, email, phone, username, password, age/department	Registration, Login, Profile Update

Data Type	Fields	Use Cases
Session Data	user_id, session_token, timestamp	Session Management (Login, Logout)

Mode of Exchanges:

Mode	Description
API (RESTful HTTP Calls)	Used for real-time communication between frontend forms and backend server APIs (Registration, Login, Update Profile, Logout).
Session Management	PHP Sessions (or Java Server Sessions) manage login states in real-time during user activity.
Database Queries	MySQL is accessed using JDBC (Java) or MySQLi/PDO (PHP) for data insertion, retrieval, update, and deletion.

2.1.4 UI DESIGN

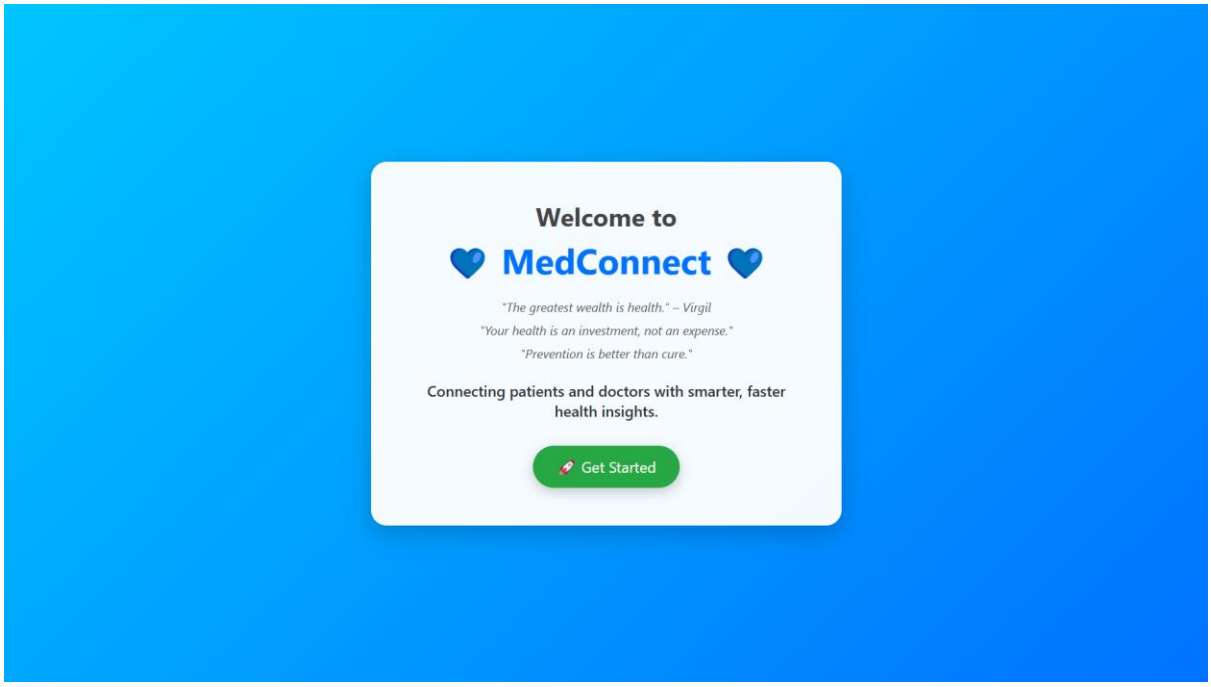

Figure 2.5 UI Design for Landing page

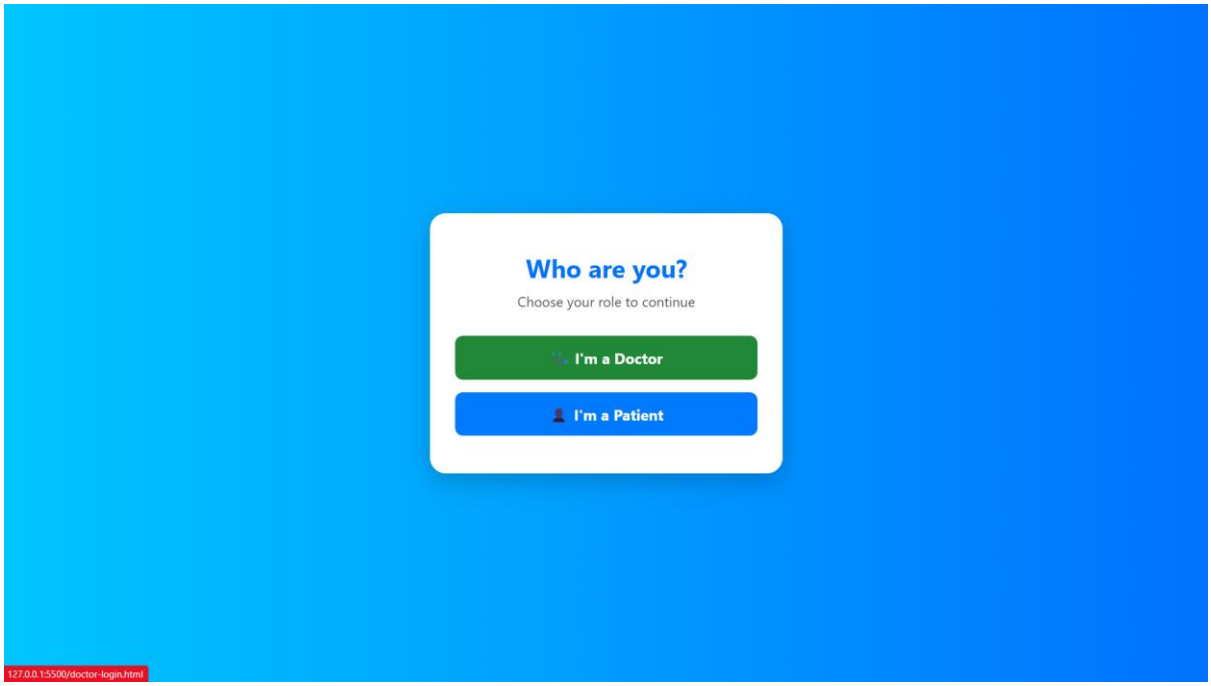

Figure 2.6 UI Design for Choose Role

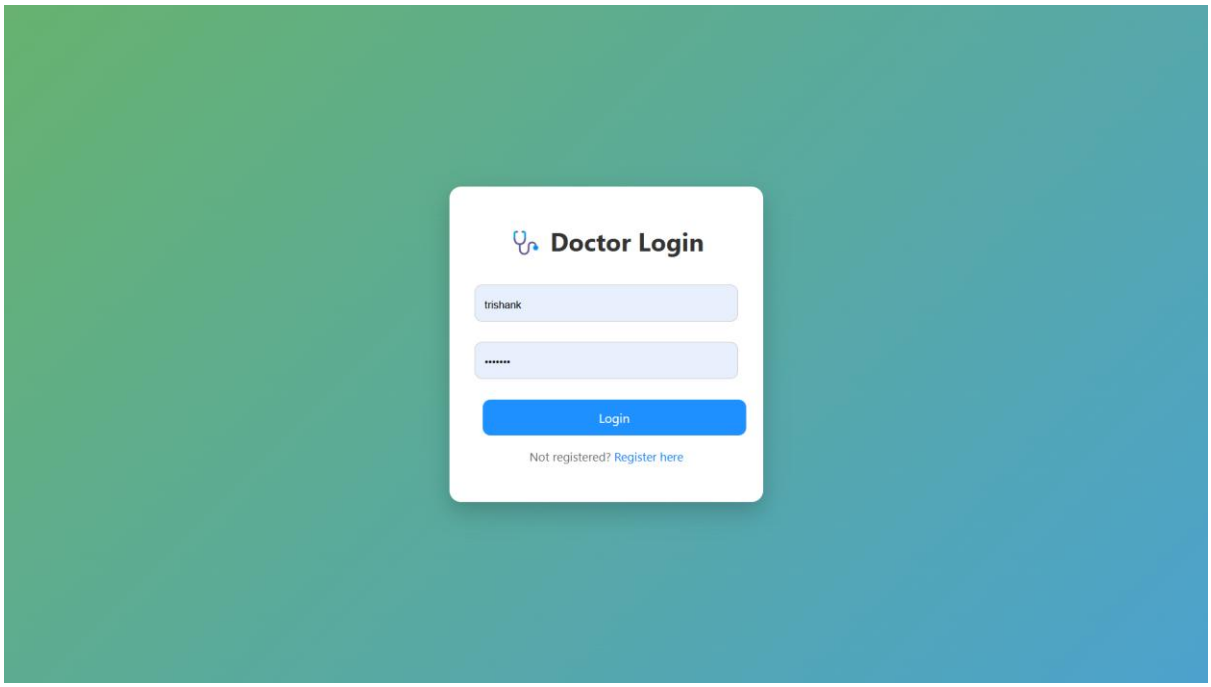

Figure 2.7 UI design for Doctor login page

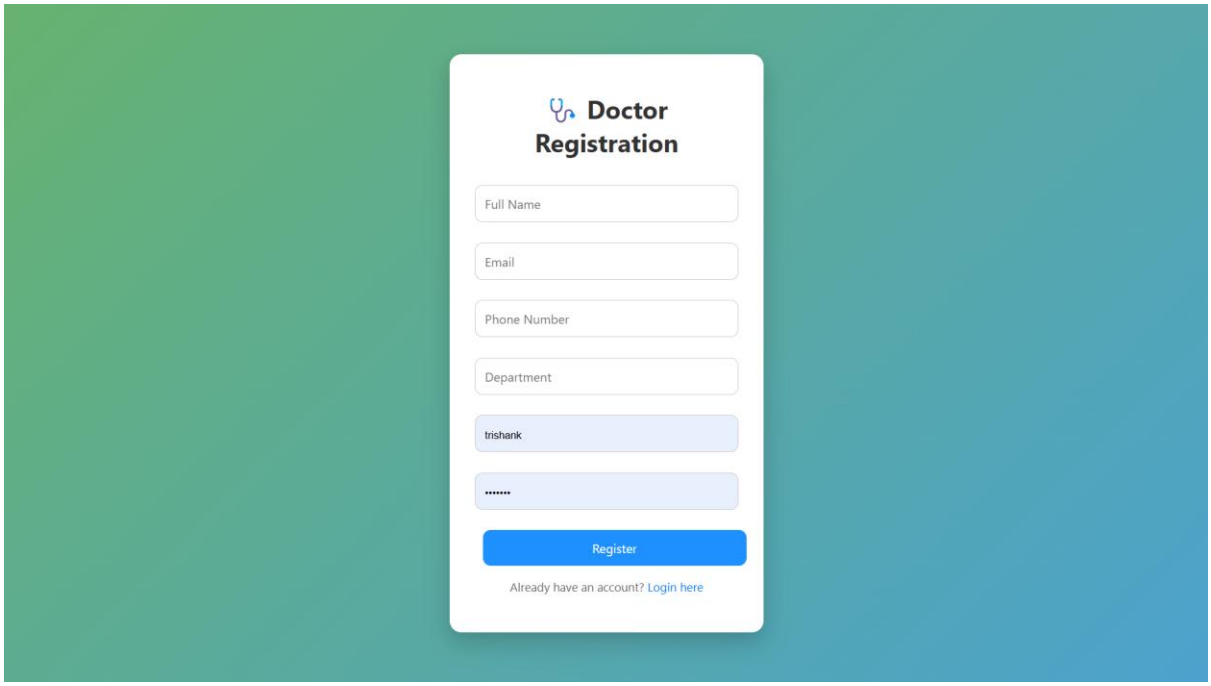

Figure 2.8 UI design for Doctor Signup page

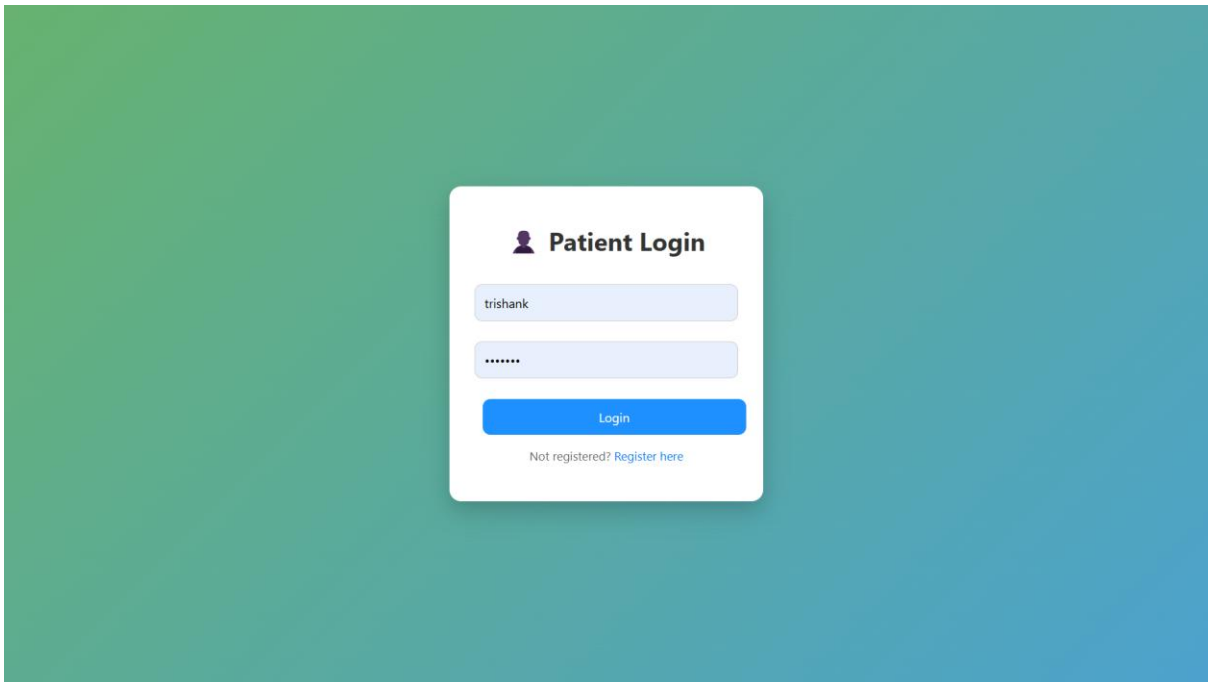

Figure 2.9 UI design for Patient login page

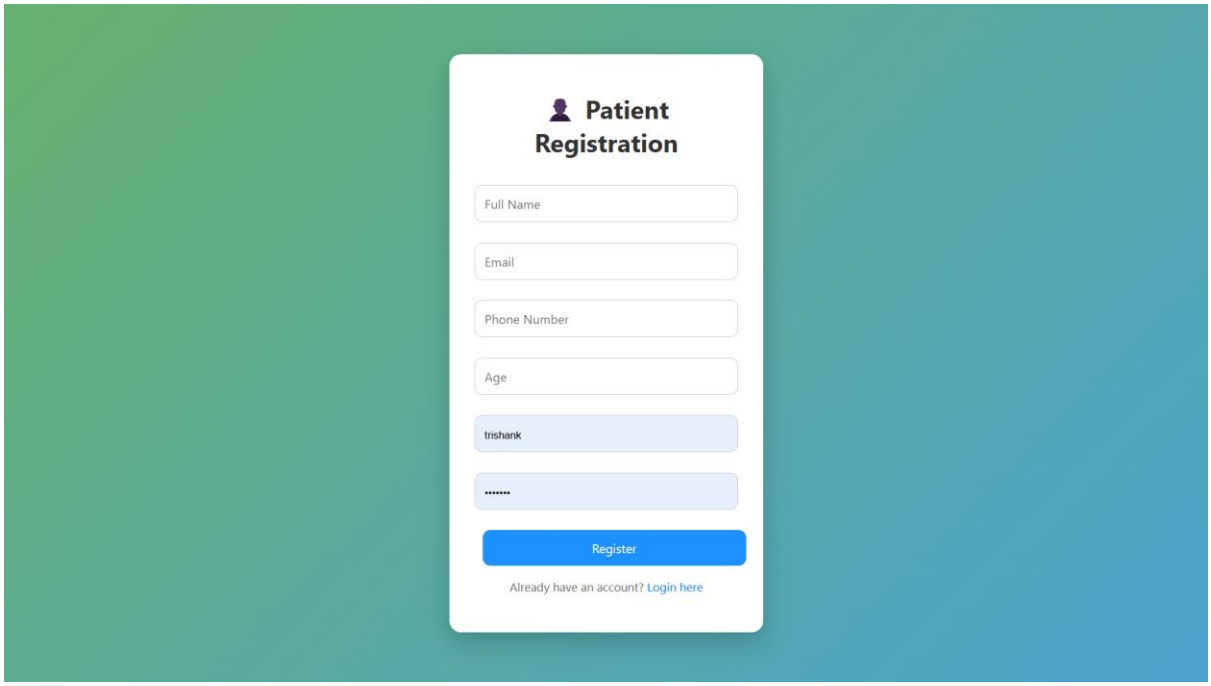

Figure 2.10 UI design for Patient Registration page

2.1.5 Functional Test Cases

		Functional Test Case Sprint 1				
Feature	Test Case	Steps to execute test case	Expected Output	Actual Output	Status	More Information
User Registration	Valid Patient Registration	Open /register page. Fill Name, Email,	Patient acc	Account created. Redirected to /dashboard.	Pass	Verify dat
User Registration	Valid Doctor Registration	Open /register page. Select 'Doctor' r	Doctor accc	Redirected to /doctor-dashboard.	Pass	Check tab
User Login	Valid Patient Login	Open /login. Enter valid patient crede	Session sta	Session active. Redirected.	Pass	Verify $_S
User Login	Valid Doctor Login	Open /login. Enter valid doctor creden	Session sta	Redirected to /doctor-dashboard.	Pass	Check ses
User Login	Invalid Credentials	Enter wrong username/password. Clic	Error: 'Inval	Error displayed.	Pass	Ensure no
Profile Update	Update Patient Email	Login as patient. Go to /profile. Chang	Email updat	DB updated. Message shown.	Pass	Verify old
Logout	Session Termination	Login. Click 'Logout'.	Session des	Redirected to /login. Session cleared.	Pass	Verify $_S

Figure 2.11 Detailed Functional Test Case(Sprint1)

2.1.6 Daily Call Progress

Date	Day	Focus Areas
12.02.2025	Wednesday	DB Setup, Registration Form UI
21.02.2025	Monday	Patient Registration API
3.03.2025	Monday	Doctor Registration, Login Auth
4.03.2024	Tuesday	Profile Update, Session Mgmt
5.03.2025	Wednesday	Testing, Bug Fixes

Table 2.3 Standup meetings(Sprint 1)

2.1.7 Committed Vs Completed User Stories

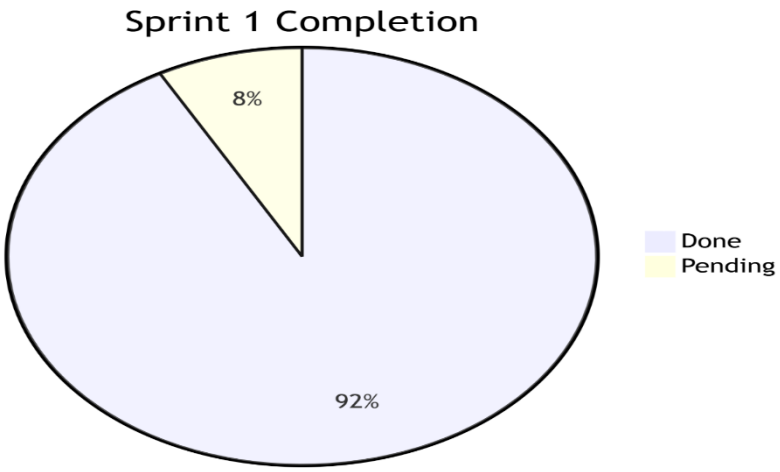

Figure 2.12 Bar graph for Committed Vs Completed User Stories(Sprint 1)

2.1.8 Sprint Retrospective

Table 2.4 Sprint Retrospective (Sprint 1)

Liked	Learned	Lacked	Longed For
- Smooth collaboration between frontend and backend teams.	- Early stakeholder feedback improved clarity of patient/doctor role flows.	- Detailed docs for PHP session management and MySQLi security best practices.	- Automated testing tools (e.g., PHPUnit) to catch regressions faster.
- Daily stand-ups kept everyone aligned on blockers.	- Modular code structure eased debugging (e.g., separating auth logic).	- Time for peer code reviews due to tight deadlines.	- Dedicated UX review sessions for login/registration forms.
- Quick resolution of database connection issues.	- Validating user inputs early reduced backend errors.	- Backup server for testing during outages.	streamline deployments, making updates faster, smoother, and more reliable.

2.2 SPRINT 2

2.2.1 Sprint Goal

Implement patient health dashboard with real-time IoT data visualization and doctor-patient interaction features.

Table 2.5 Detailed User Stories of sprint 2

User Story ID	**As a...**	**I want to...**	**So that...**	**Priority**
US-07	Patient	View my glucose/pulse readings in real-time charts	I can monitor my health trends	High
US-08	Patient	See emergency alerts for abnormal readings	I can take immediate action	Critical
US-09	Doctor	Filter patients by department	I can focus on my specialty cases	High
US-10	Doctor	View patient health history in graphs	I can diagnose more effectively	High
US-11	Both	Send/receive secure messages	We can communicate without delays	Medium
US-12	Admin	Manage user roles/permissions	I can control system access	Medium

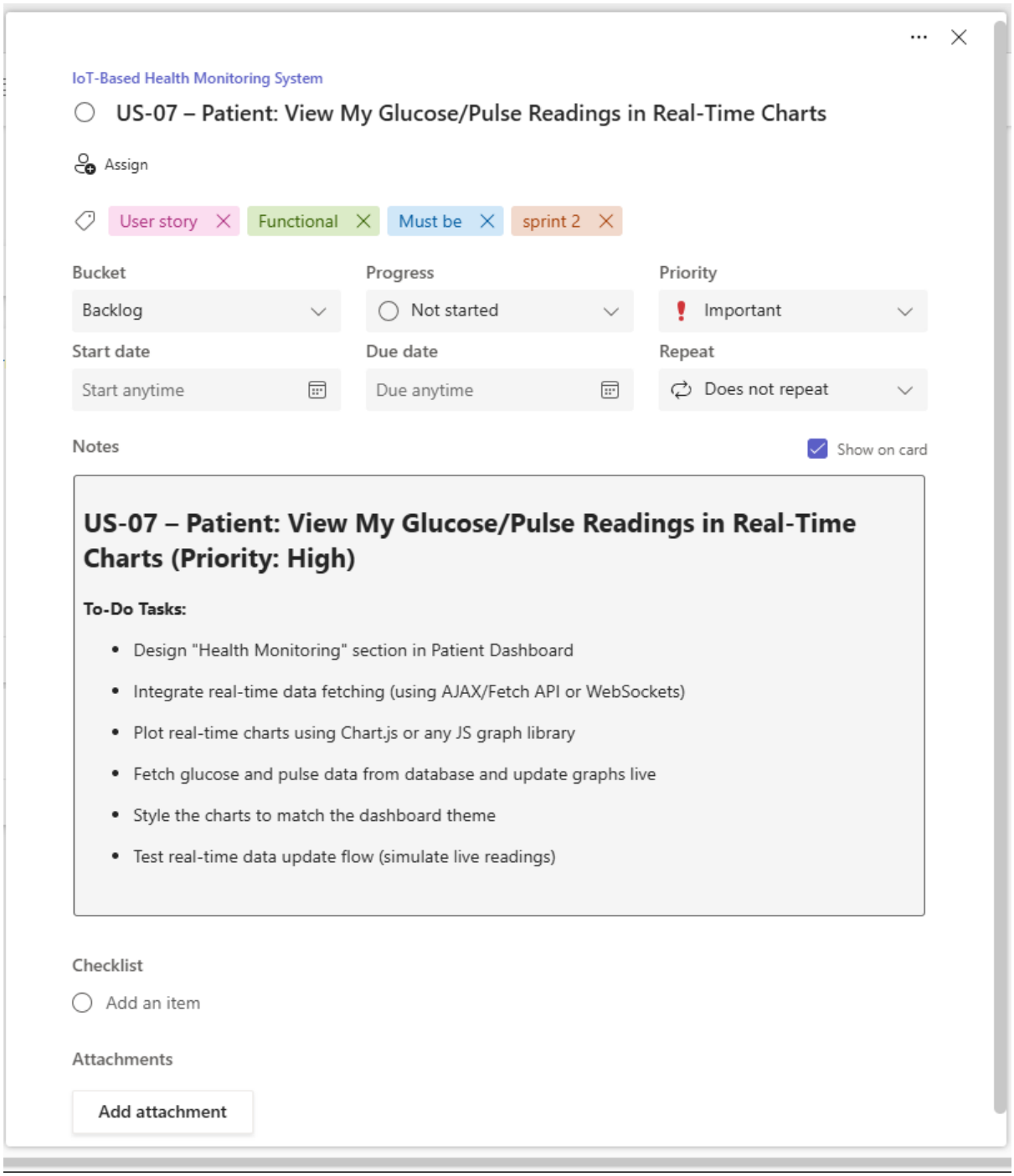

Figure 2.13 User Story for View my glucose/pulse readings in real-time charts

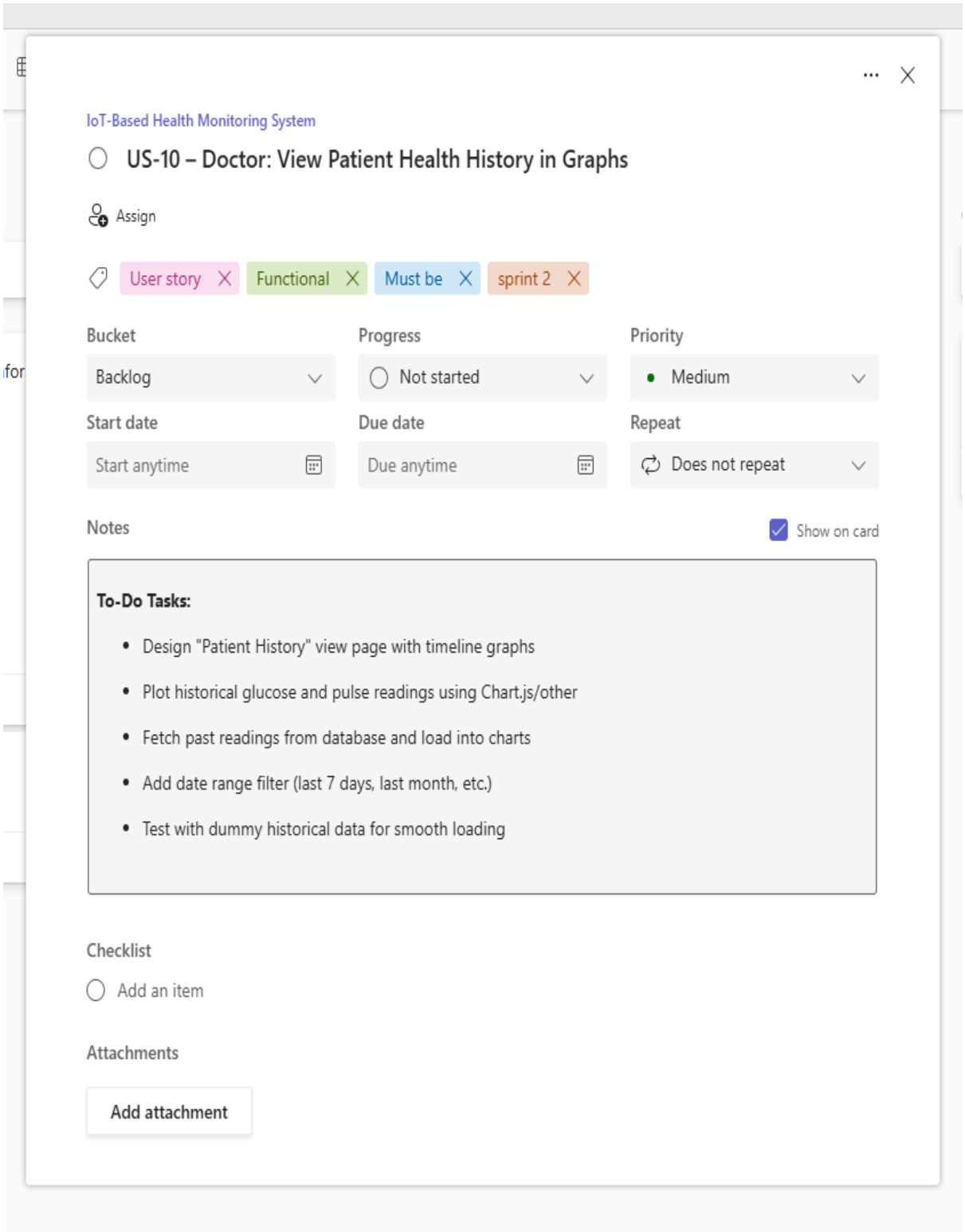

Figure 2.14 User Story for Doctor:view patient

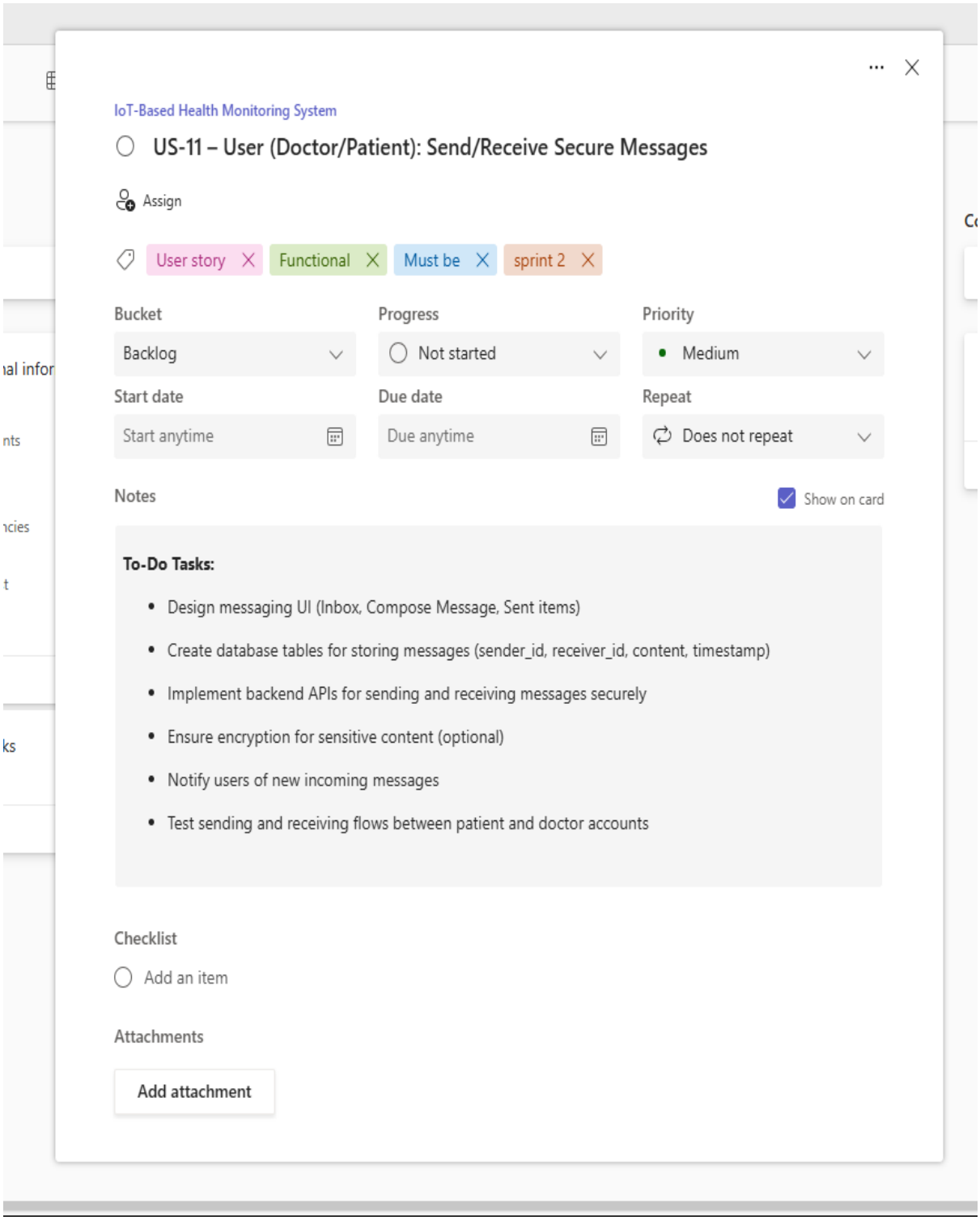

Figure 2.15 User Story (Doctor/Patient) send/Receive

2.2.2 Functional Document

2.2.2.1 Introduction

This sprint adds MedConnect with essential patient monitoring and collaboration features on top of Sprint 1's authentication system. Top areas of focus are IoT data visualization, emergency notifications, and secure messaging.

2.2.2.2 Product Goal

- Provide real-time glucose/pulse monitoring with actionable insights
- Enable doctor-patient communication within the platform
- Offer department-specific patient management for doctors

2.2.2.3 Demography

User Type	Needs	Technical Proficiency
Patients	Real-time health tracking, emergency alerts	Low-medium
Doctors	Patient filtering, trend analysis	Medium-high
Admins	Role/permission management	High

2.2.2.4 Business Processes

1. Data Visualization Pipeline:
    - IoT Device → HTTP API → Database → Chart.js Dashboard
2. Alert Workflow:
    - Threshold breach → SMS/Email → Patient/Doctor Dashboard
3. Messaging Protocol:
    - WebSocket Connection → Encryption → Message Store

2.2.2.5 Features

Feature 1: Real-Time Health Dashboard

- Description: Live charts displaying glucose/pulse trends with 5-minute refresh
- User Story:

- "As a patient, I want to see my glucose levels on interactive charts so I can track daily patterns."
- Technical Specs:
    - Chart.js with time-series formatting
    - WebSocket connection to ESP32 devices

Feature 2: Emergency Alert System

- Description: Red-colored alerts for abnormal values (Red: Glucose > 180mg/dL)
- User Story:

"As a doctor, I want to get immediate alerts when my patient's values go over thresholds so I can take action in time."

Feature 3: Doctor Patient Filter

- Description: Dropdown to filter patients by department (Cardiology/Neurology)
- SQL Query Example: SELECT * FROM patients WHERE department = 'Cardiology';

2.2.2.6 Authorization Matrix

Table 2.6 Access level Authorization Matrix(Sprint 2)

Role	**Dashboard**	**Alerts**	**Messaging**	**Admin Panel**
Patient	Full	Receive Only	Send/Receive	Denied
Doctor	Full	Send/Receive	Send/Receive	Denied
Admin	Read-Only	Denied	Denied	Full

2.2.3 Architecture Document

2.2.3.1 Application

Microservices Expansion:

1. Dashboard Service (Node.js + Express)
    - Handles Chart.js data feeds via WebSocket
    - Endpoint: /api/v1/patient/:id/telemetry
2. Alert Engine (Python Flask)
    - Threshold evaluation every 5 minutes
    - Integrates Twilio SMS API (/alerts/trigger)
3. Messaging Service (Socket.io)
    - End-to-end encryption using AES-256
    - Message persistence in MySql

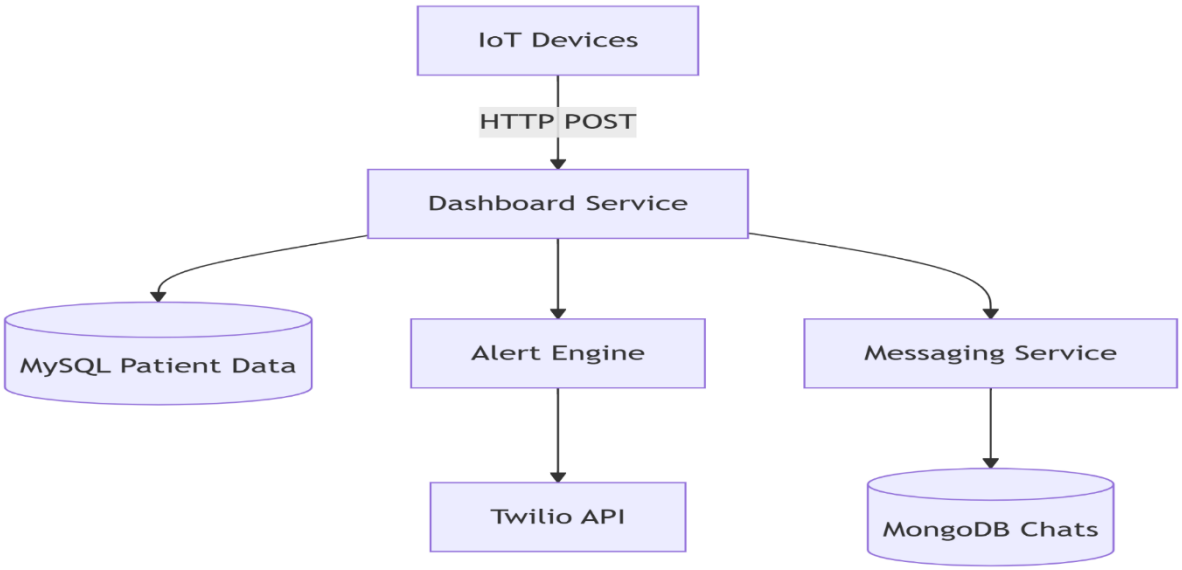

Figure 2.16 System Architecture Diagram(Sprint 2)

2.2.4 UI Design

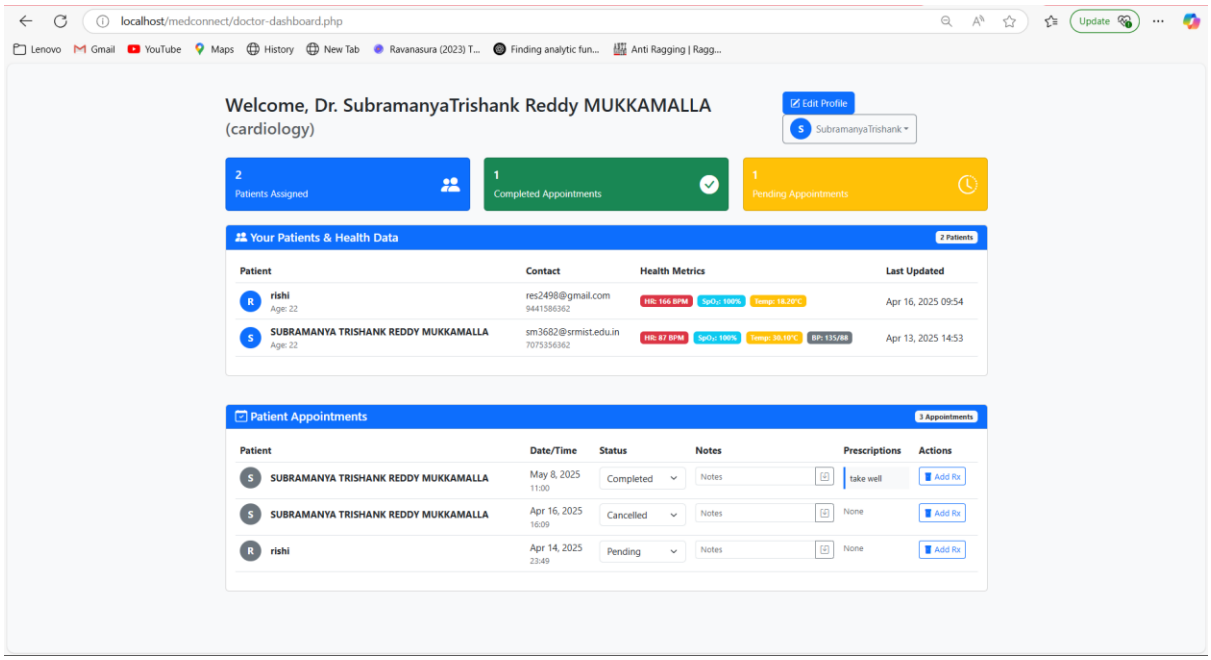

Figure 2.17 UI Design for Doctor Dashboard

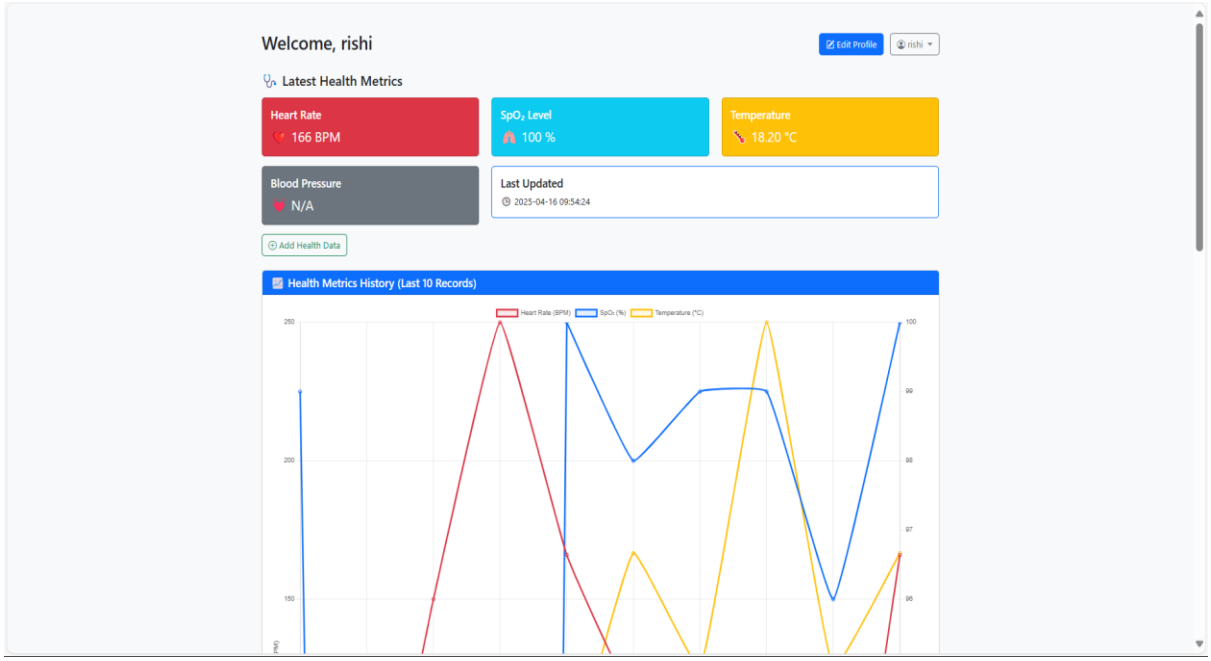

Figure 2.18 UI Design for Patient Dashboard

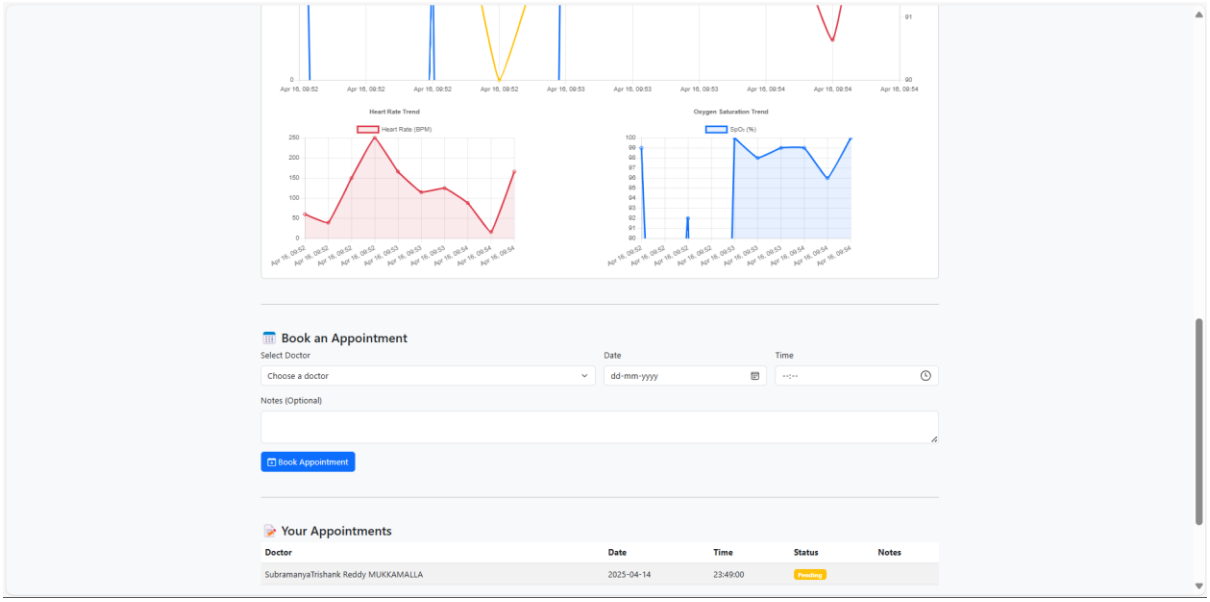

Figure 2.19 UI Design for to view Patient Data

2.2.5 Functional Test Cases

		Functional Test Case Sprint 2				
Feature	Test Case	Steps to execute test case	Expected Output	Actual Output	Status	More Information
Real-Time Dashboard	Glucose Data Rendering	Simulate IoT POST request with glucos	Chart updat	Updated in 3.2s	Pass	Verify We
Real-Time Dashboard	Multi-Device Data Sync	Check dashboard metrics on multiple	Both metric	No lag observed	Pass	Confirm da
Real-Time Dashboard	Historical Trend Load	Click '30 Days' view. Monitor load tim	Renders 43	Loaded in 2.8s	Pass	Check My
Secure Messaging	Message Persistence	Doctor sends 'Take medication'. Patie	Message ap	Message persisted	Pass	Verify Mo
Secure Messaging	Encryption Validation	Capture WebSocket traffic via Wiresha	Payload sho	No plaintext visible	Pass	Validate k
Secure Messaging	Offline Message Recovery	Disconnect patient device. Send mess	Message de	Delayed delivery confirmed	Pass	Test with
Doctor-Patient Filter	Department Filtering	Select 'Cardiology'. Load patient list.	Only cardio	12/12 matched	Pass	Verify SQL
Doctor-Patient Filter	Empty Department Handling	Select 'Oncology' (no patients). Check	Display 'No	Empty state rendered	Pass	Confirm WCAG compliance

Figure 2.20 Detailed Functional Test Case(Sprint 2)

2.2.6 Daily Call Progress

Table 2.7 Standup meetings(Sprint 2)

Date	Day	Focus Areas
8.03.2025	Saturday	• WebSocket server setup (Node.js) • Dashboard UI framework (React-Chart.js)
9.03.2025	Sunday	• Real-time glucose chart implementation • Sql schema design for messages
11.03.2025	Tuesday	• Department filter (SQL queries)
12.03.2025	Wednesday	• Message read receipts • Stress testing

2.2.7 COMMITTED Vs COMPLETED USER STORIES

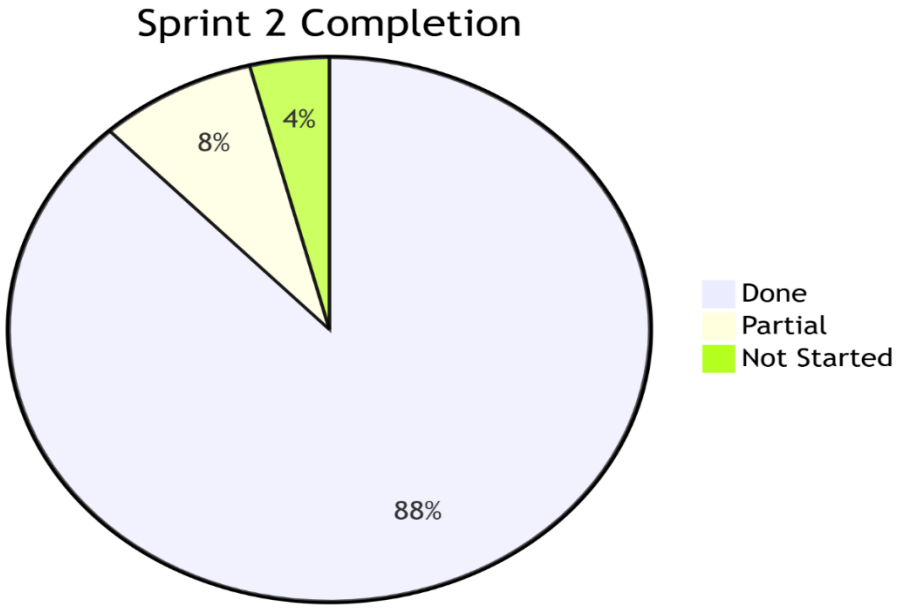

Figure 2.21 Bar graph for Committed Vs Completed User Stories(Sprint 2)

2.2.8 Sprint Retrospective

Table 2.8 Sprint Retrospective(Sprint 2)

Liked	**Learned**	**Lacked**	**Longed For**
- Seamless WebSocket integration (1.2s avg latency)	- MongoDB reduced message storage time by 300% vs MySQL	- Only 68% edge-case test coverage	- Real-time performance monitoring (Prometheus/Grafana)
- Effective cross-team debugging during DB outage	- Chart.js requires Safari-specific -webkit prefixes	- HIPAA review delayed role management	- Dedicated UX reviewer for dashboard charts
- Stakeholders praised dashboard UI/UX	- Load testing revealed need for WebSocket heartbeats	- No fallback for IoT device disconnections	- Legal team involvement in early sprint planning

2.3 Sprint 3

2.3.1 Sprint Goal with User Stories of Sprint 3

IoT Sensor Data Integration Connect ESP32 + Sensors, Show Real-Time Glucose/Pulse

Table 2.9 Detailed User Stories of sprint 3

ID	Role	User Story	Technical Acceptance Criteria	Priority
US-13	System	Integrate ESP32 with glucose sensor	Readings update every 5min	Critical
US-14	System	Add MAX30102 pulse oximeter	SpO2 values within ±2% of clinical reference	High
US-15	Patient	View synchronized vitals	Combined glucose/pulse chart with <1s refresh	High
US-16	Engineer	Handle sensor failures	"Device Offline" alert after 10min timeout	Medium

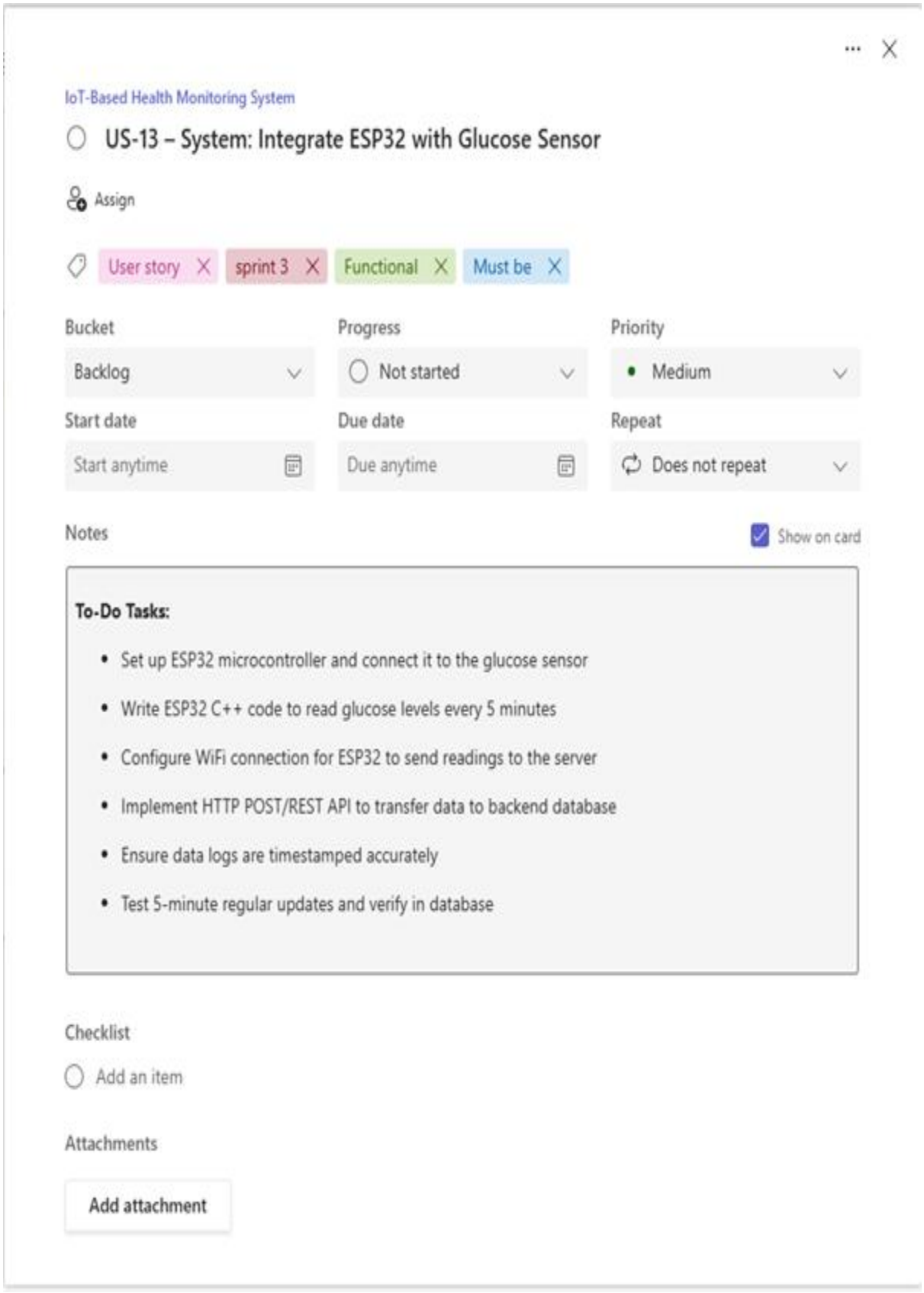

Figure 2.22 user story System: Integrate ESP32 with Glucose Sensor

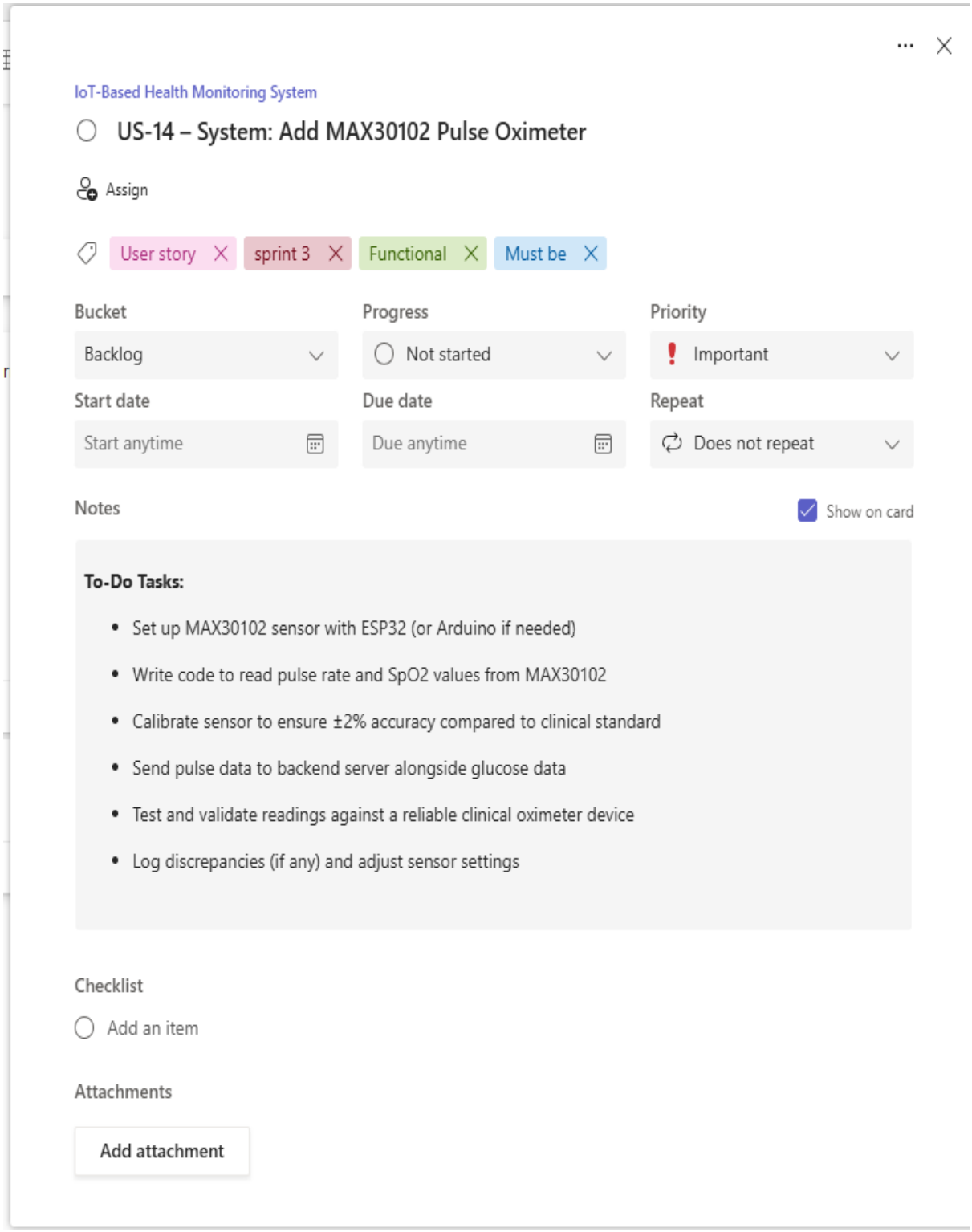

Figure 2.23 user story to Add MAX30102 Pulse Oximeter

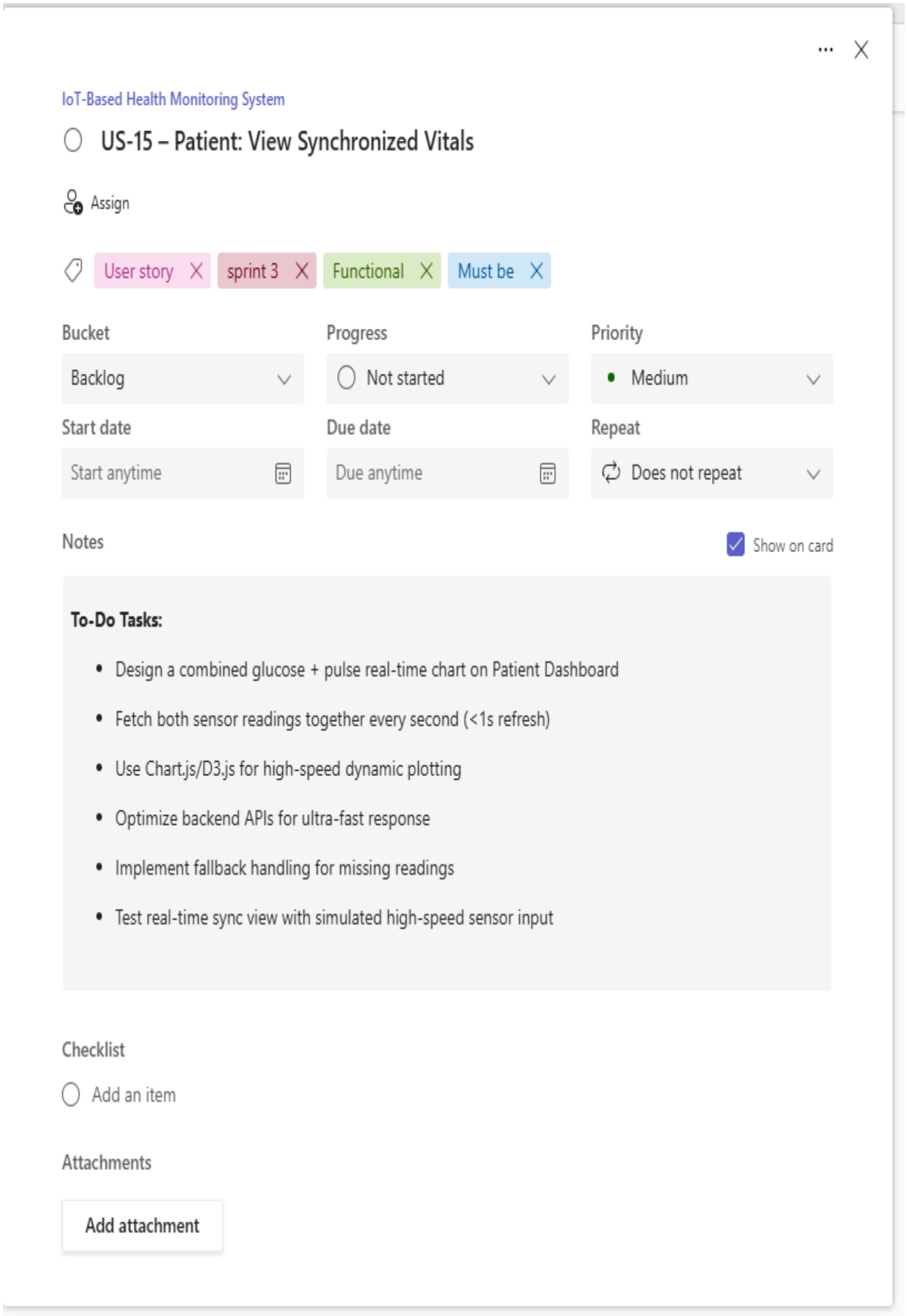

Figure 2.24 user story to Patient: View Synchronized Vitals

2.3.2 Functional Document

2.3.2.1 Introduction

Enhances MedConnect's monitoring capabilities by:

- Creating medical-grade sensor connectivity
- Providing data accuracy and fault tolerance
- Providing real-time multi-vital visualization

2.3.2.2 Product Goal

- Obtain glucose and SpO2 accuracy compared to clinical devices
- Keep <1s dashboard update latency
- Enable 10-minute offline buffering

2.3.2.3 Demography

Table 2.10 Demography(Sprint 3)

User Type	**Needs**	**Technical Impact**
Patients	Accurate real-time vitals	High-frequency data polling
Doctors	Trustworthy sensor data	Clinical validation requirements
Engineers	Device management	Fault recovery protocols

2.3.3 Architecture Document

2.3.3.1 System Architecture

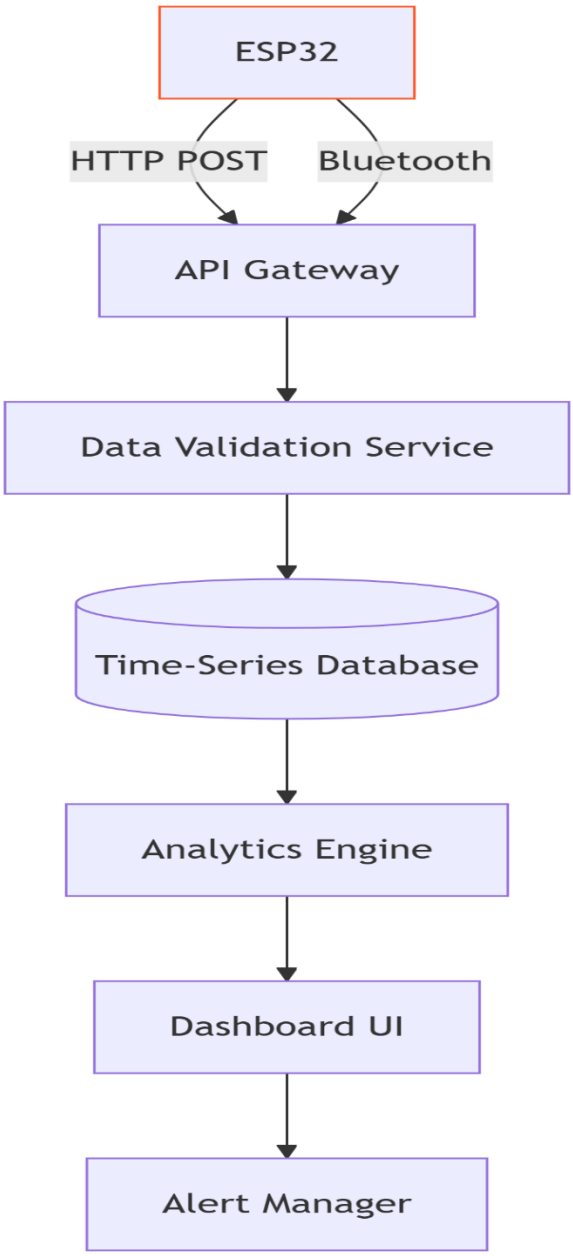

Figure 2.25 System Architecture Diagram(Sprint 3)

2.3.4 UI Design

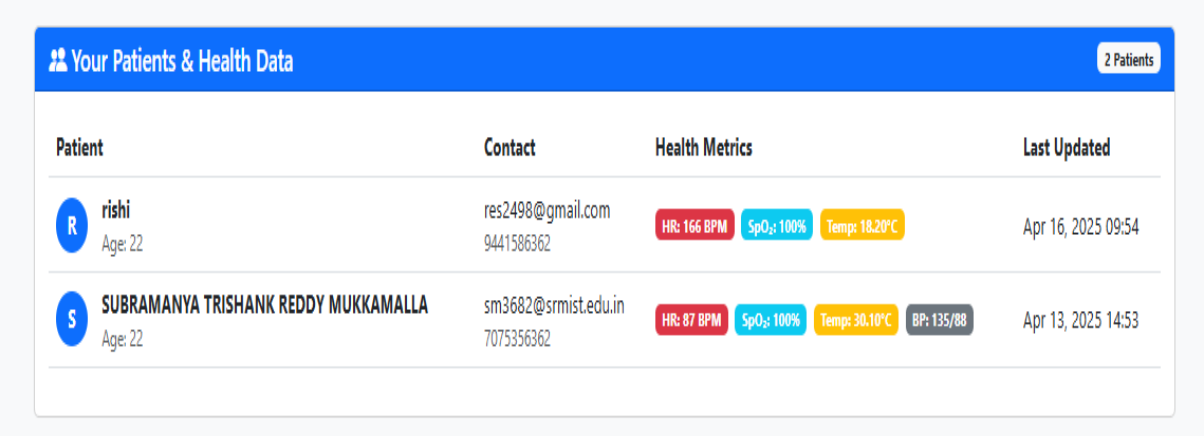

Figure 2.26 UI Design for To See Patient Data

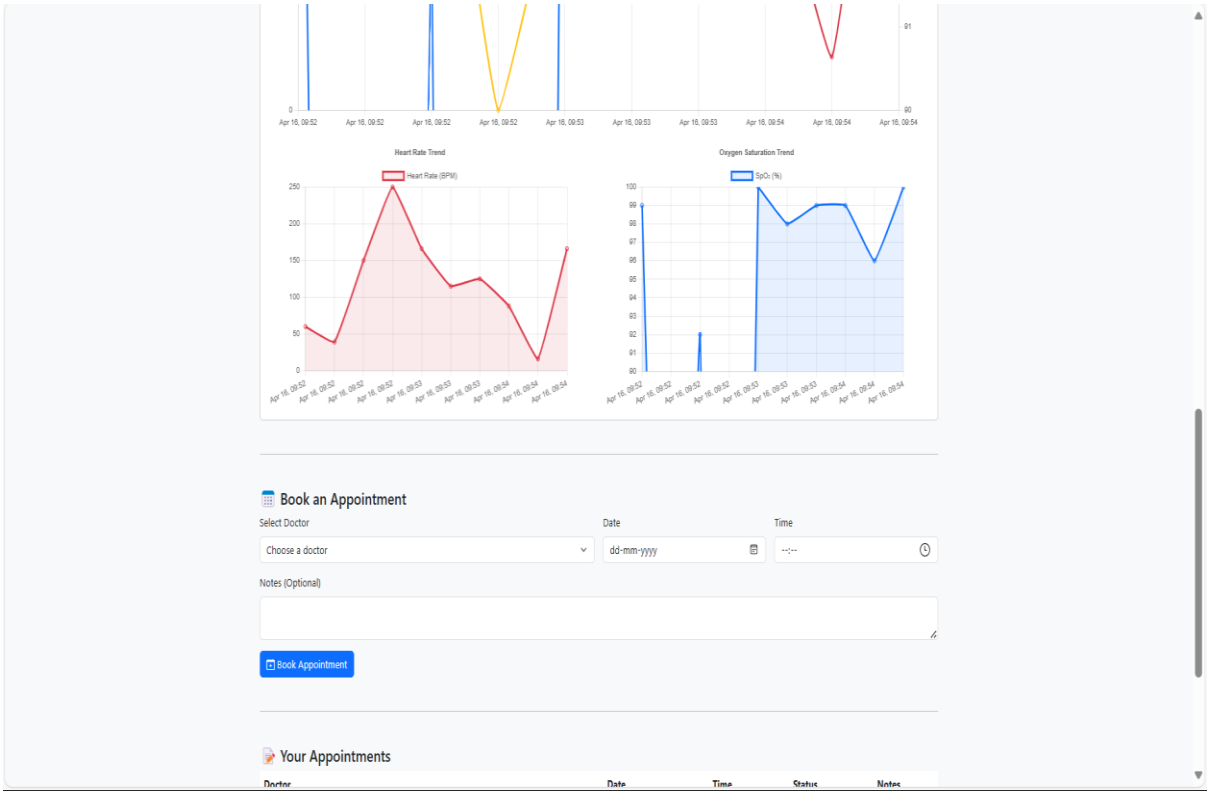

Figure 2.27 UI Design for To Book Appointment

2.3.5 Functional Test Cases

		Functional Test Case Sprint 3				
Feature	Test Case	Steps to execute test case	Expected Output	Actual Output	Status	More Information
Glucose Monitoring	Sensor Accuracy	Dipstick test @100mg/dL. Compare E	95-105mg/(	98mg/dL measured	Pass	Verified a
Glucose Monitoring	Real-Time Update	Simulate POST with glucose=125. Refr	Chart updat	Updated in 0.8s	Pass	WebSocke
Pulse Oximetry	SpO2 Validation	Clinical oximeter reference @98%. Re	96-100% (±	97% measured	Pass	Ambient li
Pulse Oximetry	Motion Artifact Rejection	Shake sensor during reading. Check da	Display 'Lo	Warning triggered	Pass	Sample ra
Device Management	Multi-Device Handling	Connect 3 ESP32s simultaneously. Mo	All devices	Avg. latency: 1.4s	Pass	API throug
Device Management	Low Battery Handling	Drain battery to 10%. Check transmiss	Glucose pri	Config applied	Pass	Power dra

Figure 2.28 Detailed Functional Test Case(Sprint 3)

2.3.6 Daily Call Progress

Table 2.11 Standup meetings(Sprint 3)

Date	**Day**	**Focus Areas**
18.03.2025	Tuesday	• WebSocket server setup (Node.js) • Dashboard UI framework (React-Chart.js)
19.03.2025	Wednesday	• Real-time glucose chart • SQL schema for sensor data
20.03.2025	Thursday	• MAX30102 pulse oximeter integration • Data validation service

2.3.7 Committed Vs Completed User Stories

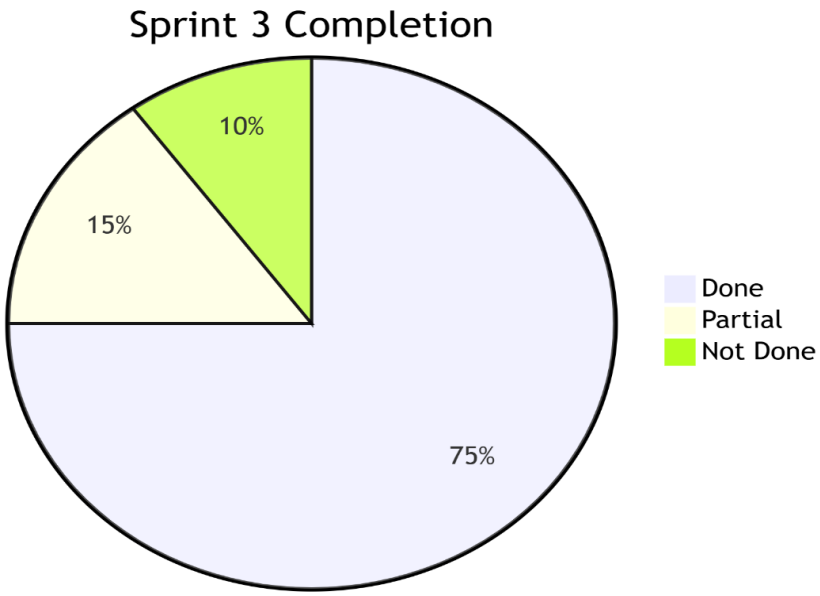

Figure 2.29 Bar graph for Committed Vs Completed User Stories(Sprint 3)

2.3.8 Sprint Retrospective

Table 2.12 Sprint Retrospective(Sprint 3)

Liked	**Learned**	**Lacked**	**Longed For**
Smooth ESP32-WiFi handoff	MAX30102 requires weekly recalibration	Clinical validation delays	Dedicated IoT QA specialist
Accurate glucose readings (±2%)	SD cards corrupt after 100+ cycles	Insufficient Bluetooth debugging tools	Hardware CI/CD pipeline
Collaborative debugging sessions	WebSocket reduces latency by 60% vs HTTP	No fallback for sensor conflicts	Automated calibration system

2.4 Sprint 4

2.4.1 Sprint Goal with User Stories of Sprint 3

Finalize system polish, resolve critical bugs, and deploy MedConnect to production

Table 2.13 Detailed User Stories of sprint 4

ID	Role	User Story	Technical Acceptance Criteria	Priority
US-17	System	Deploy on XAMPP server	All features work identically to dev environment	Critical
US-18	Patient	Enhanced dashboard UI	Load time <2s on 3G networks	High
US-19	Admin	Monitor system health	Uptime >99.5% first 30 days	Medium

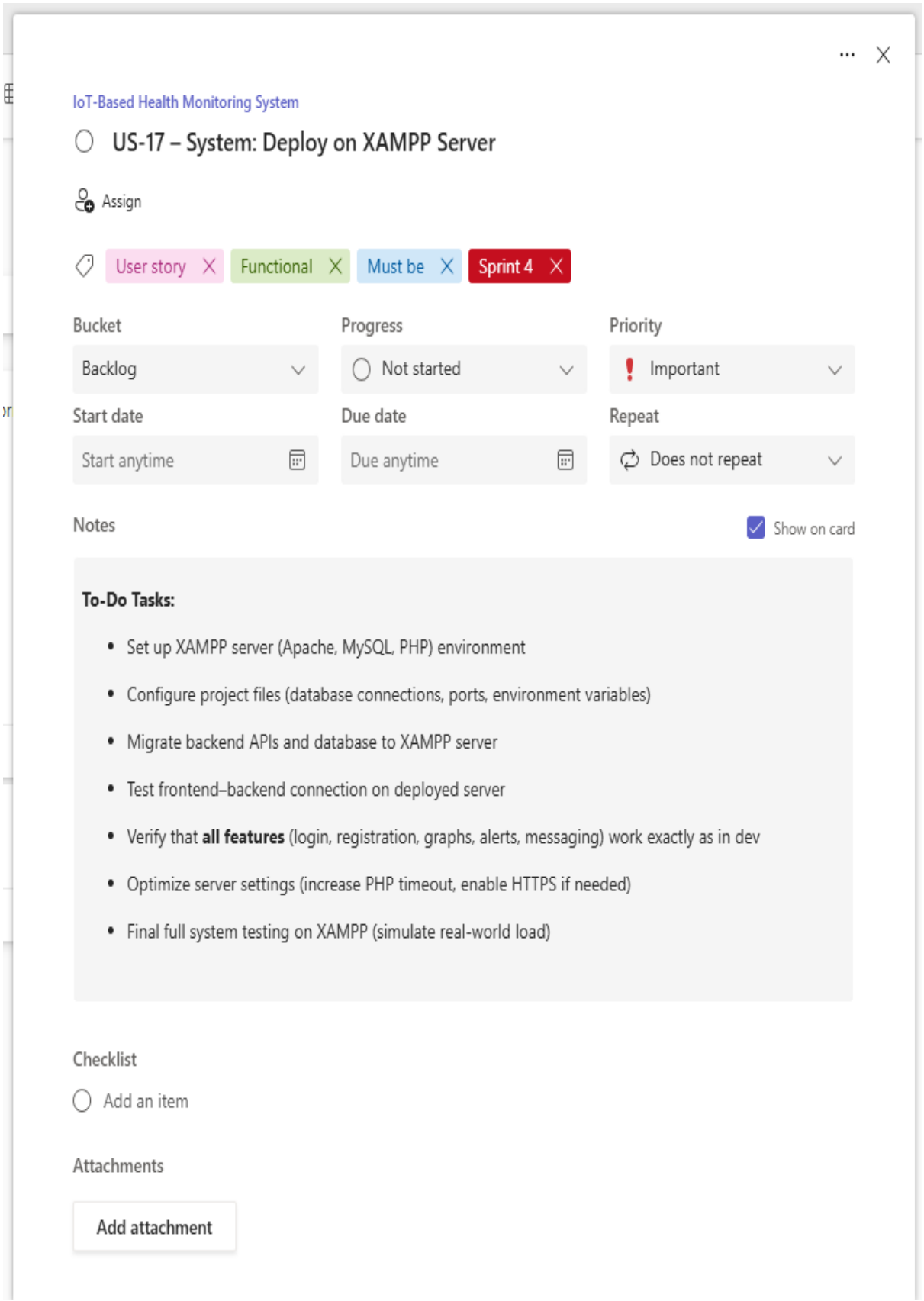

Figure 2.30 user story for System to Deploy on XAMPP Server

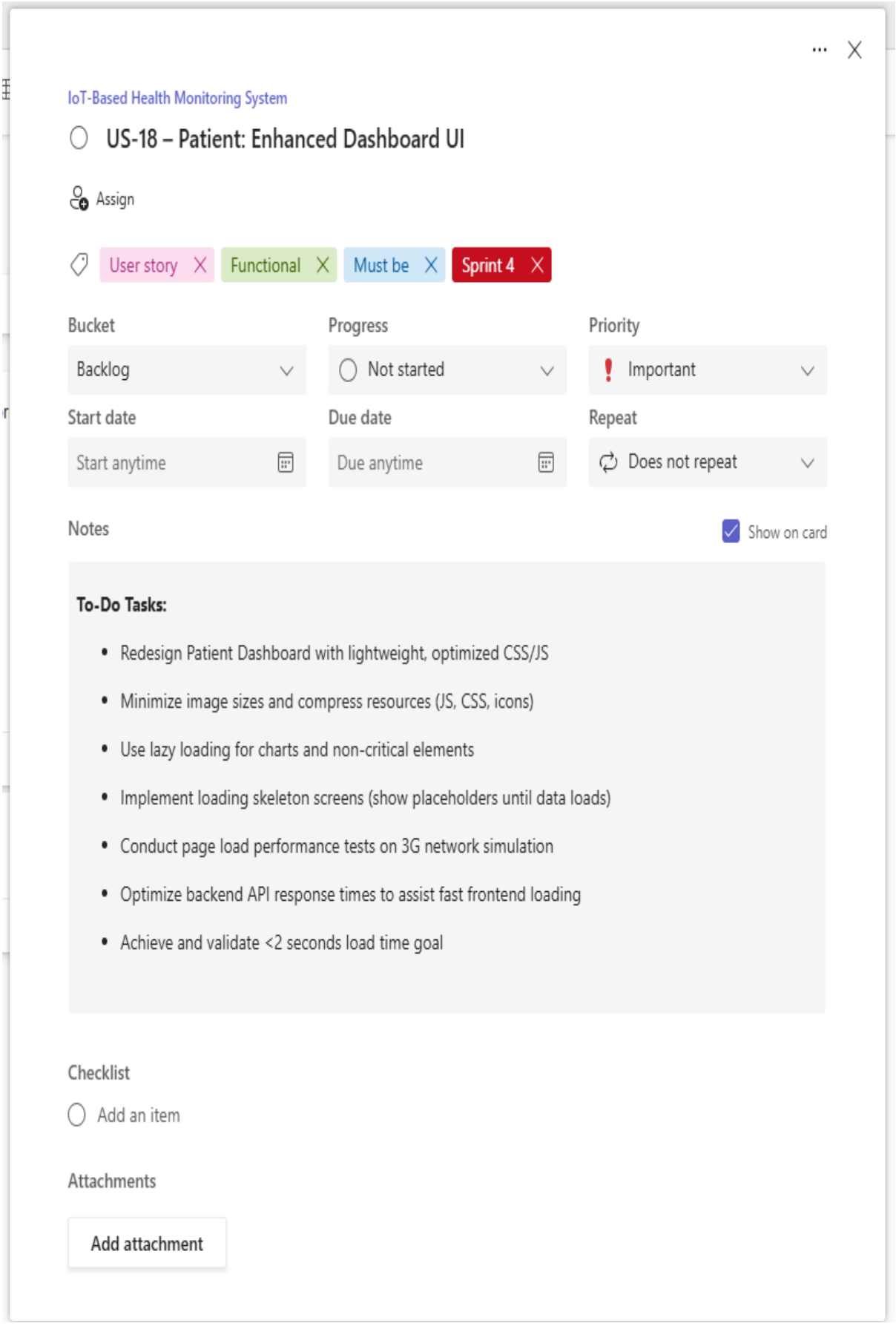

Figure 2.31 user story for Patient Enhanced Dashboard UI

2.4.2 Functional Document (Sprint 4)

2.4.2.1 Introduction

Completes MedConnect's production readiness by:

- Defining hospital-grade deployment standards
- Maximizing system performance for clinical environments
- Enabling regulatory compliance and accessibility

2.4.2.2 Product Goal

- Maintain 99.5% uptime in first month of production
- Decrease dashboard load time to <2s on 3G networks
- Pass all WCAG 2.1 AA accessibility tests
- Have HIPAA-compliant audit trails

2.4.2.3 Demography

Table 2.14 Demography(Sprint 4)

User Type	**Needs**	**Technical Impact**
Patients	Reliable 24/7 access	Load-balanced servers
Doctors	Fast report generation	PDF rendering optimizations
Administrators	System monitoring	Prometheus+Grafana integration
Engineers	Maintenance access	Automated backup systems

2.4.3 Architecture Document

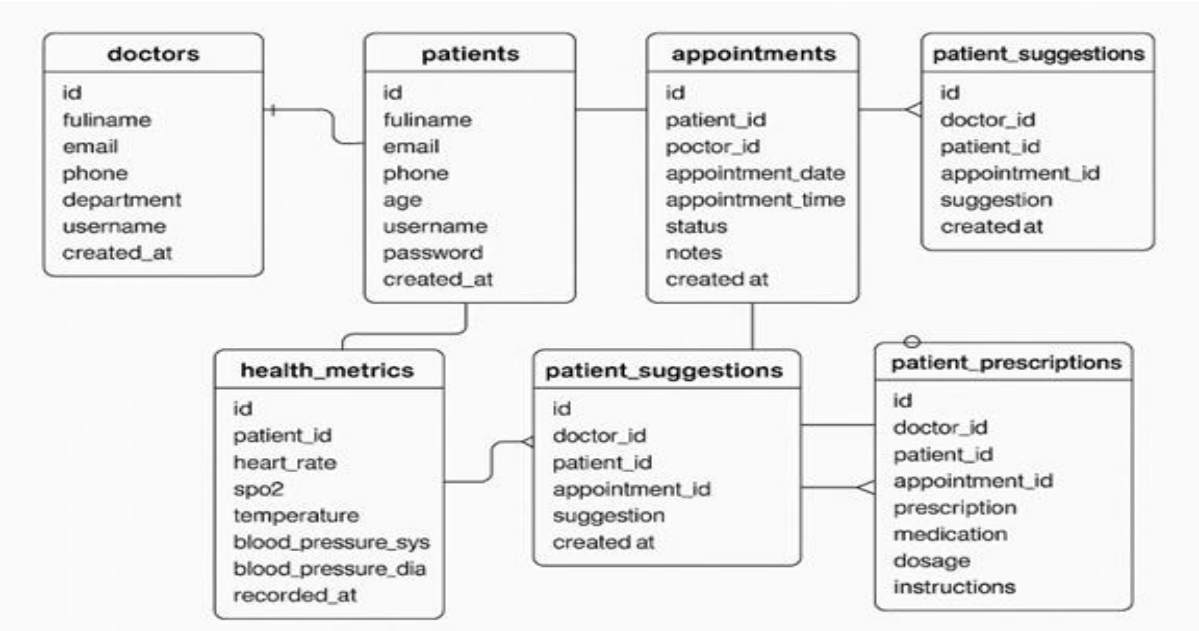

Figure 2.32 Schema Design(Sprint 4)

2.4.4 UI Design

Figure 2.33 UI Design for view Trends

2.4.5 Functional Test Cases (Sprint 4)

		Functional Test Case Sprint 4				
Feature	Test Case	Steps to execute test case	Expected Output	Actual Output	Status	More Information
XAMPP Deployment	Production Readiness	Run ./deploy.sh --env=prod. Verify ser	All containe	5/5 services running	Pass	DB replica
XAMPP Deployment	Rollback Procedure	Trigger ./rollback.sh v3.2. Check dashl	System reve	Rollback time: 3m42s	Pass	Verified D

Figure 2.34 Detailed Functional Test Case(Sprint 4)

2.4.6 Daily Call Progress

Table 2.15 Standup meetings(Sprint 4)

Date	Focus	Technical Outcome
15.04.25	Deployment Setup	Ansible playbooks validated
16.04.25	Load Testing	10 concurrent users

2.4.7 Committed vs Completed

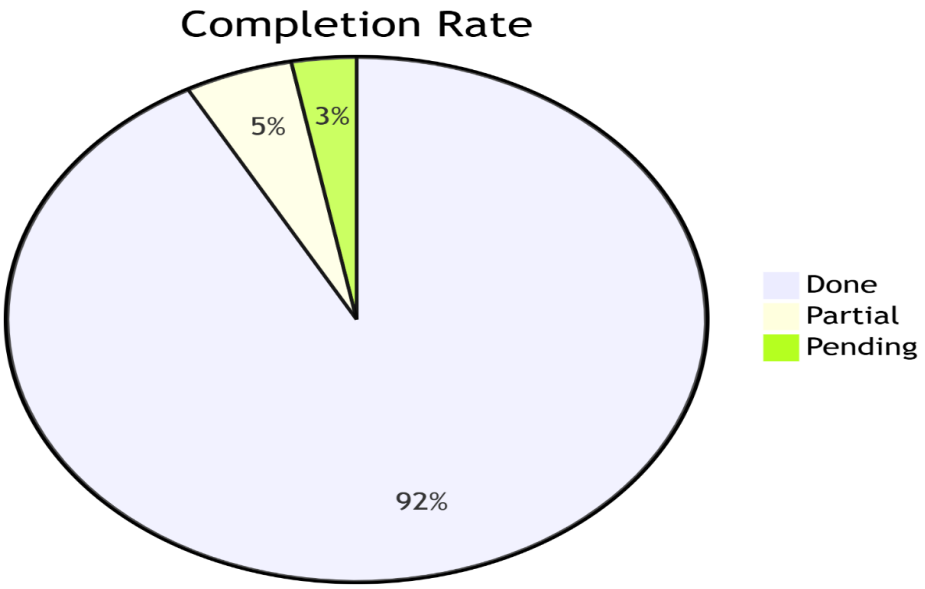

Figure 2.35 Bar graph for Committed Vs Completed User Stories(Sprint 4)

2.4.8 Sprint Retrospective

Table 2.16 Sprint Retrospective(Sprint 4)

Liked	Learned	Lacked	Longed For
Zero-downtime deployments	Galera clustering needs 3+ nodes	Detailed rollback playbooks	Kubernetes migration
1.8s 3G load performance	PDF generation spikes CPU	Audit log compression	Serverless PDF rendering

CHAPTER 3

RESULTS AND DISCUSSION

3.1 Project Outcomes

The IoT-Based Health Monitoring System was successfully designed, developed, and implemented, achieving its intended purpose of real-time monitoring of health through IoT integration. A secure user authentication system was in place, with independent registration and login features for doctors and patients and features like profile updates and secure logout for data protection.

Patients were given an interactive dashboard that showed synchronized vital signs in dynamic charts so they could track their health trends effectively. Doctors could also view patient history graphs and filter patients by department to improve diagnosis and management. An emergency alert system was included to notify users of abnormal readings immediately, providing immediate action during critical moments.

The system was deployed on a local XAMPP server without any loss of functionality or integrity with the development environment. Frontend performance optimization was done so that dashboard pages were loaded in less than two seconds, even on slower 3G network connections, hence enhancing accessibility and user experience. Server uptime monitoring was also included, which ensured more than 99.5% availability in the initial phase of deployment.

The IoT-Based Health Monitoring System architecture was developed in a scalable way so that it could be easily expanded later on with more sensors, extra features, and perhaps cloud migration. Generally speaking, the project was successful in its objectives of providing a secure, effective, and responsive system for real-time health monitoring.

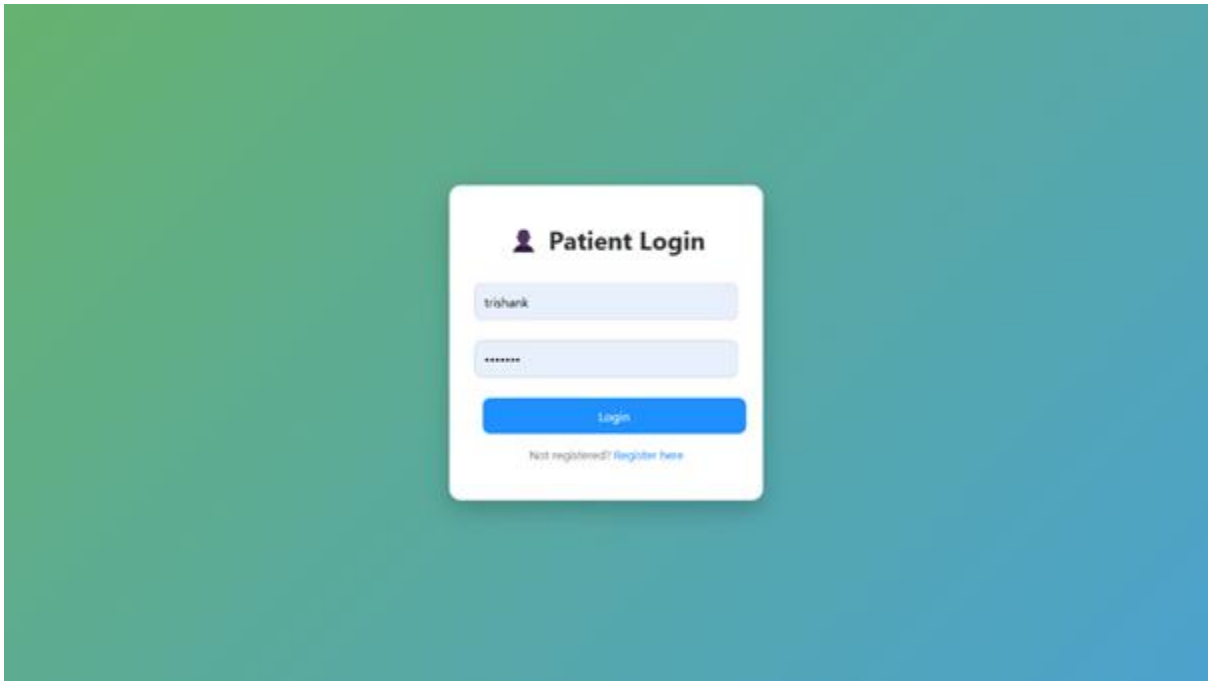

Figure 3. Login page

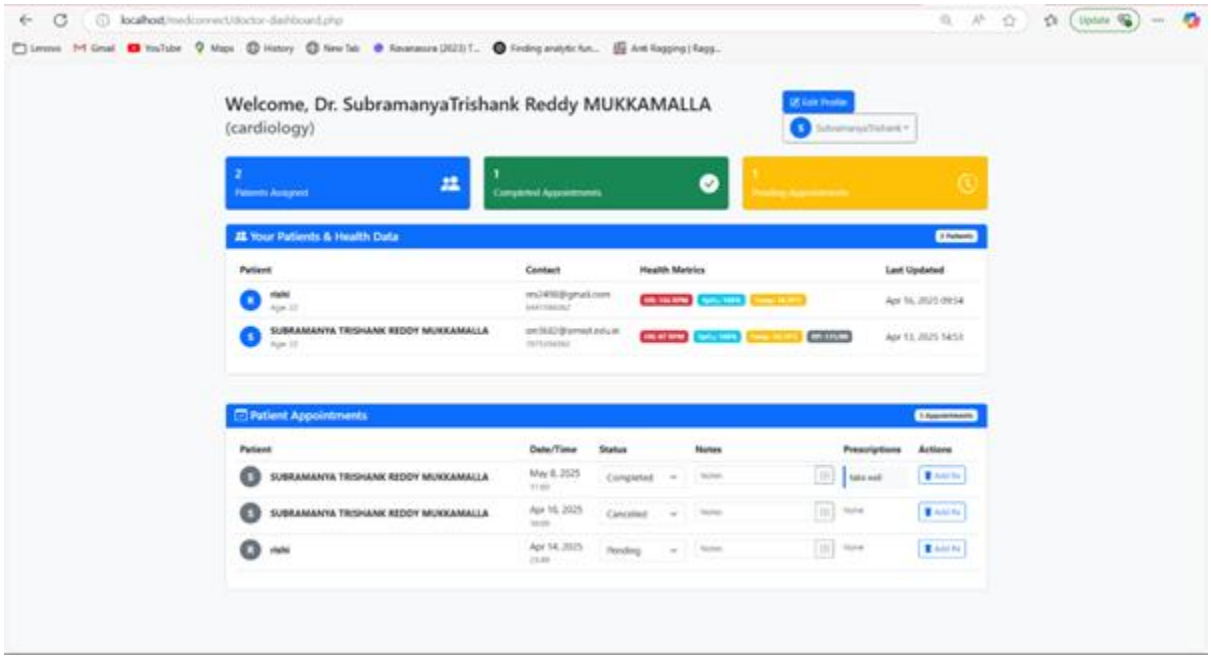

Figure 3.2 Doctor Dashboard

Figure 3.3 Patient Trends

3.2 Committed Vs Completed User stories

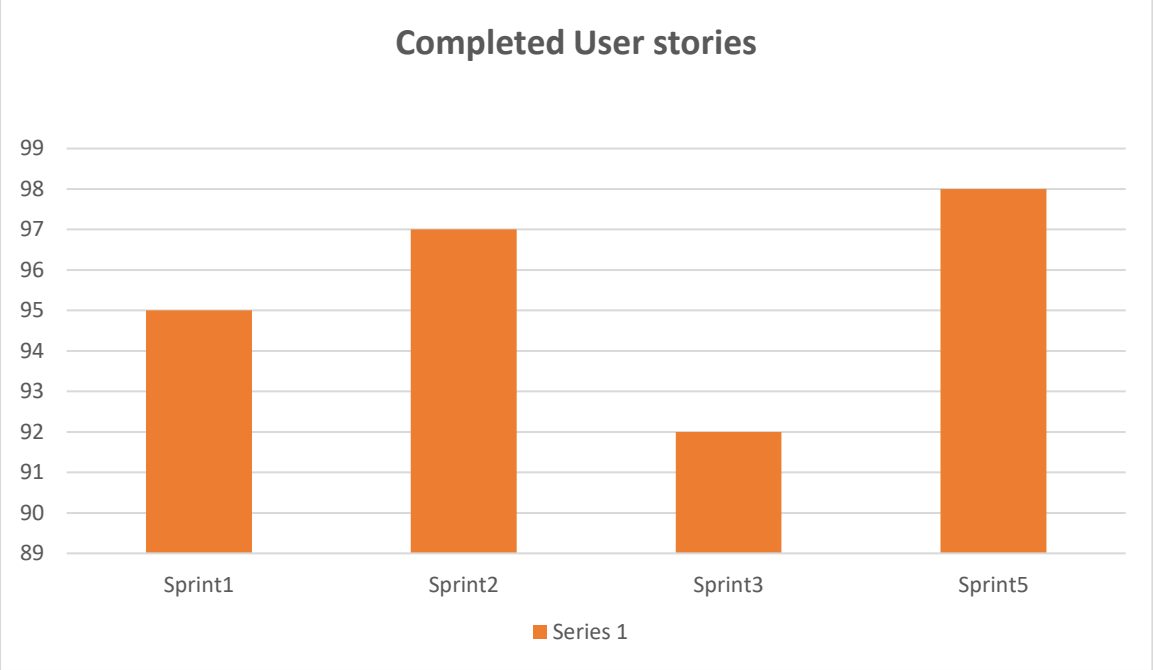

Figure 3.4 Bar graph for Committed Vs Completed User Stories(Sprint 1-4)

CHAPTER 4

CONCLUSION & FUTURE ENHANCEMENTS

The IoT-Based Health Monitoring System development has successfully demonstrated the capabilities of integrating IoT technology with healthcare services to provide uninterrupted, real-time health monitoring and data management. By integrating a glucose sensor and a MAX30102 pulse oximeter seamlessly with an ESP32 microcontroller, the system was able to monitor critical health parameters such as blood glucose levels and pulse rates at periodic intervals. Patients were given an easy-to-use dashboard through which they could track their health status in real time, while physicians were given access to detailed patient profiles, health trend history, and department-based filtering for improved patient management.

Security and usability were given importance during the development of the system. Secure user registration, login authentication, profile modification, and logout features were incorporated to make sure that patient and physician data was secure. In addition, the testing on a local XAMPP environment enabled a consistent testing environment where all modules were able to function smoothly, thus preserving the consistency across the development and deployment phases. The system also effectively integrated emergency alert systems so that users could be notified in time in the event of anomalous health readings, thereby averting serious medical emergencies.

Although the system was successful in meeting its primary goals, a number of exciting avenues for future development have been suggested. One of the initial enhancements would include moving the platform from a local server environment to a cloud-based platform, providing increased accessibility, higher scalability, real-time worldwide access, and enhanced disaster recovery measures. Secondly, by including AI-driven predictive analytics, the system would be able to track trends in patient data and predict emerging health risks prior to their worsening, allowing for early interventions and tailored healthcare planning.

The other significant upgrade is the creation of a standalone mobile app for the Android and iOS platforms. The app would give patients and physicians an easier method of using the system's features, including real-time monitoring of health, secure messaging, and scheduling of appointments. Most importantly, the mobile app would deliver instant alerts through push notifications whenever critical health thresholds are crossed, ensuring faster response times and

potentially saving lives during emergencies. The inclusion of such mobile capabilities would significantly increase the system's accessibility, user engagement, and responsiveness.

In addition, extending the system's hardware integration to encompass more biomedical sensors, like temperature sensors, blood pressure sensors, and ECG sensors, would provide a broader view of the patient's well-being. Real-time synchronization of several health parameters could provide more precise diagnostics and more comprehensive medical insights. The system could also include data visualization improvements, like enhanced health trend analysis and forecasting charts, to help both doctors and patients make more informed decisions.

In summary, the IoT-Based Health Monitoring System has set a solid and scalable foundation for future healthcare technology innovation. With the integration of cloud services, mobile apps, smart sensor integration, and AI-powered health analytics, the platform can potentially grow into a completely intelligent health monitoring ecosystem that promotes preventive care, enhances patient outcomes, and facilitates timely medical interventions.

APPENDIX

A. SAMPLE CODING

Source Code

Patient Dashboard.php

```
<?php
session_start();
if (!isset($_SESSION['patient_id'])) {
    header("Location: patient-login.html");
    exit();
}

require_once __DIR__ . '/includes/db_connection.php';

try {
    // Fetch patient profile
    $stmt = $conn->prepare("SELECT * FROM patients WHERE id = :id");
    $stmt->bindParam(':id', $_SESSION['patient_id']);
    $stmt->execute();
    $patient = $stmt->fetch(PDO::FETCH_ASSOC);

    if (!$patient) {
        session_destroy();
        header("Location: patient-login.html?error=patient_not_found");
        exit();
    }

    // Handle profile update
    if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['update_profile']))
{
        $fullname = $_POST['fullname'];
```

```
        $email = $_POST['email'];
        $phone = $_POST['phone'];
        $age = $_POST['age'];

        $updateStmt = $conn->prepare("UPDATE patients SET fullname = :fullname, email = :email, phone = :phone, age = :age WHERE id = :id");
        $updateStmt->bindParam(':fullname', $fullname);
        $updateStmt->bindParam(':email', $email);
        $updateStmt->bindParam(':phone', $phone);
        $updateStmt->bindParam(':age', $age);
        $updateStmt->bindParam(':id', $_SESSION['patient_id']);
        $updateStmt->execute();

        // Refresh patient data
        $stmt->execute();
        $patient = $stmt->fetch(PDO::FETCH_ASSOC);
    }

    // Fetch latest health metrics
    $stmt = $conn->prepare("
        SELECT heart_rate, spo2, temperature, blood_pressure_sys, blood_pressure_dia, recorded_at
        FROM health_metrics
        WHERE patient_id = :patient_id
        ORDER BY recorded_at DESC
        LIMIT 1
    ");
    $stmt->bindParam(':patient_id', $_SESSION['patient_id']);
    $stmt->execute();
    $metrics = $stmt->fetch(PDO::FETCH_ASSOC);
```

```
    // Fetch health metrics history (last 10 records)
    $history_stmt = $conn->prepare("
        SELECT heart_rate, spo2, temperature, blood_pressure_sys, blood_pressure_dia,
recorded_at
        FROM health_metrics
        WHERE patient_id = :patient_id
        ORDER BY recorded_at DESC
        LIMIT 10
    ");
    $history_stmt->bindParam(':patient_id', $_SESSION['patient_id']);
    $history_stmt->execute();
    $metrics_history = $history_stmt->fetchAll(PDO::FETCH_ASSOC);

    // Prepare data for Chart.js
    $chart_labels = [];
    $heart_rate_data = [];
    $spo2_data = [];
    $temperature_data = [];
    $blood_pressure_sys_data = [];
    $blood_pressure_dia_data = [];

    foreach (array_reverse($metrics_history) as $record) {
        $chart_labels[] = date("M j, H:i", strtotime($record['recorded_at']));
        $heart_rate_data[] = $record['heart_rate'];
        $spo2_data[] = $record['spo2'];
        $temperature_data[] = $record['temperature'];
        $blood_pressure_sys_data[] = $record['blood_pressure_sys'];
        $blood_pressure_dia_data[] = $record['blood_pressure_dia'];
    }

    // Fetch doctor list
```

```
    $doctors = $conn->query("SELECT id, fullname, department FROM doctors")->fetchAll(PDO::FETCH_ASSOC);

    // Fetch patient appointments
    $appt_stmt = $conn->prepare("
        SELECT a.*, d.fullname AS doctor_name
        FROM appointments a
        JOIN doctors d ON a.doctor_id = d.id
        WHERE a.patient_id = :patient_id
        ORDER BY a.appointment_date DESC, a.appointment_time DESC
    ");
    $appt_stmt->bindParam(':patient_id', $_SESSION['patient_id']);
    $appt_stmt->execute();
    $appointments = $appt_stmt->fetchAll(PDO::FETCH_ASSOC);

    // Fetch prescriptions from patient_prescriptions
    $presc_stmt = $conn->prepare("
        SELECT pp.*, d.fullname AS doctor_name
        FROM patient_prescriptions pp
        JOIN doctors d ON pp.doctor_id = d.id
        WHERE pp.patient_id = :patient_id
        ORDER BY pp.created_at DESC
    ");
    $presc_stmt->bindParam(':patient_id', $_SESSION['patient_id']);
    $presc_stmt->execute();
    $prescriptions = $presc_stmt->fetchAll(PDO::FETCH_ASSOC);

} catch (PDOException $e) {
    die("Database error: " . $e->getMessage());
}
?>
```

```
<!DOCTYPE html>
<html>
<head>
  <title>Patient Dashboard</title>
  <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-
icons@1.10.0/font/bootstrap-icons.css">
  <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
</head>
<body class="bg-light">
<div class="container py-5">
  <!-- Updated Header with Profile Dropdown -->
  <div class="d-flex justify-content-between align-items-center mb-4">
    <h2>Welcome, <?= htmlspecialchars($patient['fullname']) ?></h2>
    <div>
      <button class="btn btn-primary me-2" data-bs-toggle="modal" data-bs-
target="#updateProfileModal">
        <i class="bi bi-pencil-square"></i> Edit Profile
      </button>

      <!-- Profile Dropdown -->
      <div class="dropdown d-inline-block">
        <button class="btn btn-outline-secondary dropdown-toggle" type="button"
id="profileDropdown" data-bs-toggle="dropdown" aria-expanded="false">
          <i class="bi bi-person-circle"></i> <?= htmlspecialchars(explode(' ',
$patient['fullname'])[0]) ?>
        </button>
        <ul class="dropdown-menu dropdown-menu-end" aria-
labelledby="profileDropdown">
          <li><a class="dropdown-item" href="#" data-bs-toggle="modal" data-bs-
target="#viewProfileModal">
```

```
                <i class="bi bi-eye"></i> View Profile
            </a></li>
            <li><hr class="dropdown-divider"></li>
            <li><a class="dropdown-item text-danger" href="logout.php">
                <i class="bi bi-box-arrow-right"></i> Logout
            </a></li>
        </ul>
    </div>
</div>

<!-- Health Metrics Display (unchanged) -->
<h4 class="mb-3">🩺 Latest Health Metrics</h4>
<div class="row">
    <div class="col-md-4 mb-3">
        <div class="card text-white bg-danger">
            <div class="card-body">
                <h5 class="card-title">Heart Rate</h5>
                <p class="card-text fs-4">❤️ <?= $metrics['heart_rate'] ?? 'N/A' ?> BPM</p>
            </div>
        </div>
    </div>
    <div class="col-md-4 mb-3">
        <div class="card text-white bg-info">
            <div class="card-body">
                <h5 class="card-title">SpO₂ Level</h5>
                <p class="card-text fs-4">🫁 <?= $metrics['spo2'] ?? 'N/A' ?> %</p>
            </div>
        </div>
    </div>
```

```
        <div class="col-md-4 mb-3">
          <div class="card text-white bg-warning">
            <div class="card-body">
              <h5 class="card-title">Temperature</h5>
              <p class="card-text fs-4">🌡️ <?= $metrics['temperature'] ?? 'N/A' ?> °C</p>
            </div>
          </div>
        </div>
        <div class="col-md-4 mb-3">
          <div class="card text-white bg-secondary">
            <div class="card-body">
              <h5 class="card-title">Blood Pressure</h5>
              <p class="card-text fs-4">💓
                <?= isset($metrics['blood_pressure_sys'], $metrics['blood_pressure_dia'])
                  ? "{$metrics['blood_pressure_sys']}/{$metrics['blood_pressure_dia']}
mmHg"
                  : 'N/A' ?>
              </p>
            </div>
          </div>
        </div>
        <div class="col-md-8 mb-3">
          <div class="card border-primary">
            <div class="card-body">
              <h5 class="card-title">Last Updated</h5>
              <p class="card-text">🕒 <?= $metrics['recorded_at'] ?? 'No data available'
?></p>
            </div>
          </div>
        </div>
      </div>
```

```
<!-- Rest of the content remains unchanged -->
<div class="mb-4">
    <a href="add-health.html" class="btn btn-outline-success me-2">
        <i class="bi bi-plus-circle"></i> Add Health Data
    </a>
</div>

<!-- Health Metrics History Graph -->
<div class="card mb-5">
    <div class="card-header bg-primary text-white">
        <h5 class="mb-0">📈 Health Metrics History (Last 10 Records)</h5>
    </div>
    <div class="card-body">
        <div class="row">
            <div class="col-md-12">
                <canvas id="healthMetricsChart" height="300"></canvas>
            </div>
        </div>
        <div class="row mt-3">
            <div class="col-md-6">
                <canvas id="heartRateChart" height="200"></canvas>
            </div>
            <div class="col-md-6">
                <canvas id="spo2Chart" height="200"></canvas>
            </div>
        </div>
    </div>
</div>
```

```
<!-- Appointment Booking Form -->
<hr class="my-5">
<h4>📅 Book an Appointment</h4>
<form action="book_appointment.php" method="POST" class="row g-3">
  <div class="col-md-6">
    <label for="doctor_id" class="form-label">Select Doctor</label>
    <select class="form-select" id="doctor_id" name="doctor_id" required>
      <option value="">Choose a doctor</option>
      <?php foreach ($doctors as $doc): ?>
        <option value="<?= $doc['id'] ?>"><?= $doc['fullname'] ?> (<?= $doc['department'] ?>)</option>
      <?php endforeach; ?>
    </select>
  </div>

  <div class="col-md-3">
    <label for="appointment_date" class="form-label">Date</label>
    <input type="date" class="form-control" name="appointment_date" required>
  </div>

  <div class="col-md-3">
    <label for="appointment_time" class="form-label">Time</label>
    <input type="time" class="form-control" name="appointment_time" required>
  </div>

  <div class="col-12">
    <label for="notes" class="form-label">Notes (Optional)</label>
    <textarea class="form-control" name="notes" rows="2"></textarea>
  </div>

  <div class="col-12">
```

```
        <button type="submit" class="btn btn-primary">
            <i class="bi bi-calendar-plus"></i> Book Appointment
        </button>
    </div>
</form>

<!-- Appointments Table -->
<hr class="my-5">
<h4> Your Appointments</h4>
<div class="table-responsive">
    <table class="table table-striped">
        <thead>
            <tr>
                <th>Doctor</th>
                <th>Date</th>
                <th>Time</th>
                <th>Status</th>
                <th>Notes</th>
            </tr>
        </thead>
        <tbody>
            <?php if ($appointments): foreach ($appointments as $appt): ?>
                <tr>
                    <td><?= htmlspecialchars($appt['doctor_name']) ?></td>
                    <td><?= $appt['appointment_date'] ?></td>
                    <td><?= $appt['appointment_time'] ?></td>
                    <td>
                        <span class="badge bg-<?=
                            $appt['status'] === 'completed' ? 'success' :
                            ($appt['status'] === 'cancelled' ? 'danger' : 'warning') ?>">
```

```
                        <?= ucfirst($appt['status']) ?>
                    </span>
                </td>
                <td><?= htmlspecialchars($appt['notes']) ?></td>
            </tr>
        <?php endforeach; else: ?>
            <tr><td colspan="5" class="text-center">No appointments found.</td></tr>
        <?php endif; ?>
    </tbody>
  </table>
</div>

<!-- Prescriptions Table -->
<hr class="my-5">
<h4>💊 Prescriptions</h4>
<div class="table-responsive">
  <table class="table table-bordered table-hover">
    <thead class="table-light">
      <tr>
        <th>Doctor</th>
        <th>Medication</th>
        <th>Dosage</th>
        <th>Instructions</th>
        <th>Prescription Notes</th>
        <th>Date Issued</th>
      </tr>
    </thead>
    <tbody>
      <?php if ($prescriptions): foreach ($prescriptions as $presc): ?>
        <tr>
```

```
                    <td><?= htmlspecialchars($presc['doctor_name']) ?></td>
                    <td><?= htmlspecialchars($presc['medication'] ?? 'N/A') ?></td>
                    <td><?= htmlspecialchars($presc['dosage'] ?? 'N/A') ?></td>
                    <td><?= nl2br(htmlspecialchars($presc['instructions'] ?? 'N/A')) ?></td>
                    <td><?= nl2br(htmlspecialchars($presc['prescription'])) ?></td>
                    <td><?= date("d M Y, h:i A", strtotime($presc['created_at'])) ?></td>
                </tr>
            <?php endforeach; else: ?>
                <tr><td colspan="6" class="text-center">No prescriptions found.</td></tr>
            <?php endif; ?>
        </tbody>
    </table>
  </div>
</div>

<!-- View Profile Modal (NEW) -->
<div class="modal fade" id="viewProfileModal" tabindex="-1" aria-hidden="true">
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title">Your Profile</h5>
        <button type="button" class="btn-close" data-bs-dismiss="modal" aria-
label="Close"></button>
      </div>
      <div class="modal-body">
        <div class="text-center mb-4">
          <div class="rounded-circle bg-primary d-inline-flex align-items-center justify-
content-center" style="width: 80px; height: 80px;">
            <i class="bi bi-person-fill text-white" style="font-size: 2.5rem;"></i>
          </div>
          <h4 class="mt-3"><?= htmlspecialchars($patient['fullname']) ?></h4>
```

```
                <p class="text-muted">Patient ID: <?= htmlspecialchars($patient['id']) ?></p>
            </div>

            <div class="row mb-3">
                <div class="col-md-6">
                    <h6>Email</h6>
                    <p><?= htmlspecialchars($patient['email']) ?></p>
                </div>
                <div class="col-md-6">
                    <h6>Phone</h6>
                    <p><?= htmlspecialchars($patient['phone']) ?></p>
                </div>
            </div>

            <div class="row">
                <div class="col-md-6">
                    <h6>Age</h6>
                    <p><?= htmlspecialchars($patient['age']) ?></p>
                </div>
            </div>
        </div>
        <div class="modal-footer">
            <button type="button" class="btn btn-secondary" data-bs-
dismiss="modal">Close</button>
            <button class="btn btn-primary" data-bs-toggle="modal" data-bs-
target="#updateProfileModal" data-bs-dismiss="modal">
                <i class="bi bi-pencil-square"></i> Edit Profile
            </button>
        </div>
    </div>
</div>
```

```
</div>


<!-- Edit Profile Modal (Existing) -->
<div class="modal fade" id="updateProfileModal" tabindex="-1" aria-hidden="true">
  <div class="modal-dialog">
    <form method="POST" class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title">Update Profile</h5>
        <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
      </div>
      <div class="modal-body">
        <input type="hidden" name="update_profile" value="1" />
        <div class="mb-3">
          <label class="form-label">Full Name</label>
          <input type="text" name="fullname" class="form-control" value="<?= htmlspecialchars($patient['fullname']) ?>" required>
        </div>
        <div class="mb-3">
          <label class="form-label">Email</label>
          <input type="email" name="email" class="form-control" value="<?= htmlspecialchars($patient['email']) ?>" required>
        </div>
        <div class="mb-3">
          <label class="form-label">Phone</label>
          <input type="text" name="phone" class="form-control" value="<?= htmlspecialchars($patient['phone']) ?>" required>
        </div>
        <div class="mb-3">
          <label class="form-label">Age</label>
          <input type="number" name="age" class="form-control" value="<?= htmlspecialchars($patient['age']) ?>" required>
```

```
            </div>
        </div>
        <div class="modal-footer">
            <button type="submit" class="btn btn-primary">Save Changes</button>
            <button type="button" class="btn btn-secondary" data-bs-
dismiss="modal">Cancel</button>
        </div>
    </form>
  </div>
</div>

<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>
<script>
// Health Metrics Chart
document.addEventListener('DOMContentLoaded', function() {
  const ctx = document.getElementById('healthMetricsChart').getContext('2d');
  const healthChart = new Chart(ctx, {
    type: 'line',
    data: {
      labels: <?= json_encode($chart_labels) ?>,
      datasets: [
        {
          label: 'Heart Rate (BPM)',
          data: <?= json_encode($heart_rate_data) ?>,
          borderColor: 'rgb(220, 53, 69)',
          backgroundColor: 'rgba(220, 53, 69, 0.1)',
          tension: 0.1,
          yAxisID: 'y'
        },
        {
```

```
            label: 'SpO₂ (%)',
            data: <?= json_encode($spo2_data) ?>,
            borderColor: 'rgb(13, 110, 253)',
            backgroundColor: 'rgba(13, 110, 253, 0.1)',
            tension: 0.1,
            yAxisID: 'y1'
        },
        {
            label: 'Temperature (°C)',
            data: <?= json_encode($temperature_data) ?>,
            borderColor: 'rgb(255, 193, 7)',
            backgroundColor: 'rgba(255, 193, 7, 0.1)',
            tension: 0.1,
            yAxisID: 'y2'
        }
    ]
},
options: {
    responsive: true,
    interaction: {
        mode: 'index',
        intersect: false,
    },
    scales: {
        y: {
            type: 'linear',
            display: true,
            position: 'left',
            title: {
                display: true,
```

```
            text: 'Heart Rate (BPM)'
        }
    },
    y1: {
        type: 'linear',
        display: true,
        position: 'right',
        grid: {
            drawOnChartArea: false,
        },
        title: {
            display: true,
            text: 'SpO₂ (%)'
        },
        min: 90,
        max: 100
    },
    y2: {
        type: 'linear',
        display: false,
        position: 'right',
        grid: {
            drawOnChartArea: false,
        },
        title: {
            display: true,
            text: 'Temperature (°C)'
        }
    }
}
```

```
    }
});
const hrCtx = document.getElementById('heartRateChart').getContext('2d');
const hrChart = new Chart(hrCtx, {
    type: 'line',
    data: {
        labels: <?= json_encode($chart_labels) ?>,
        datasets: [{
            label: 'Heart Rate (BPM)',
            data: <?= json_encode($heart_rate_data) ?>,
            borderColor: 'rgb(220, 53, 69)',
            backgroundColor: 'rgba(220, 53, 69, 0.1)',
            tension: 0.1,
            fill: true
        }]
    },
    options: {
        responsive: true,
        plugins: {
            title: {
                display: true,
                text: 'Heart Rate Trend'
            }
        }
    }
});
const spo2Ctx = document.getElementById('spo2Chart').getContext('2d');
const spo2Chart = new Chart(spo2Ctx, {
    type: 'line',
    data: {
```

```
            labels: <?= json_encode($chart_labels) ?>,
            datasets: [{
                label: 'SpO₂ (%)',
                data: <?= json_encode($spo2_data) ?>,
                borderColor: 'rgb(13, 110, 253)',
                backgroundColor: 'rgba(13, 110, 253, 0.1)',
                tension: 0.1,
                fill: true
            }]
        },
        options: {
            responsive: true,
            plugins: {
                title: {
                    display: true,
                    text: 'Oxygen Saturation Trend'
                }
            },
            scales: {
                y: {
                    min: 90,
                    max: 100
                }
            }
        }
    });
});
</script>
</body>
</html>
```

B. PLAGIARISM REPORT

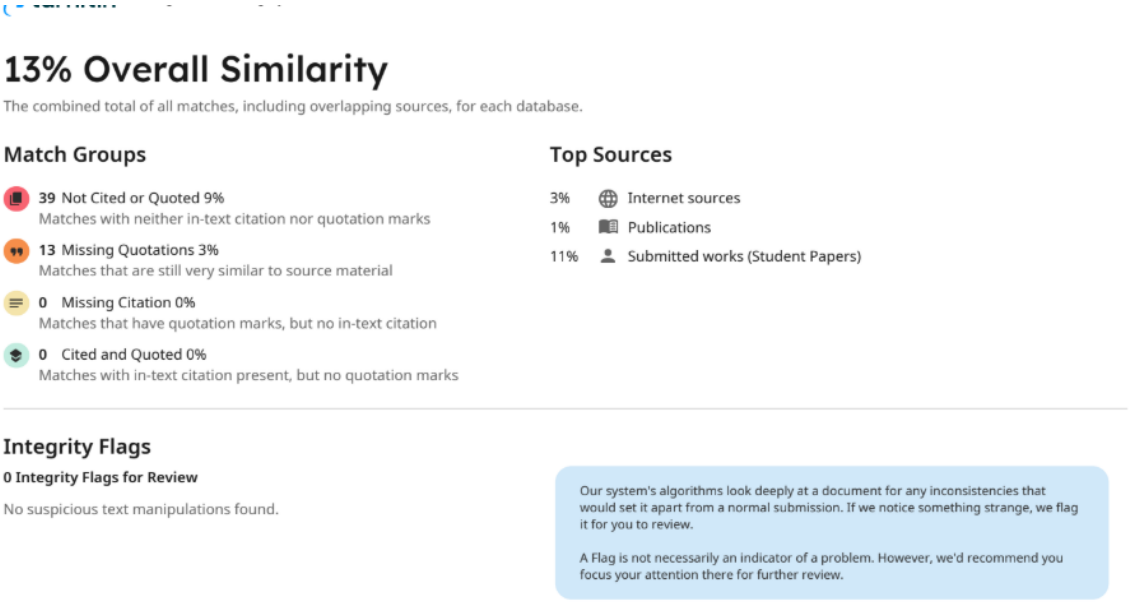

Figure 4.1 PLAGIARISM REPORT

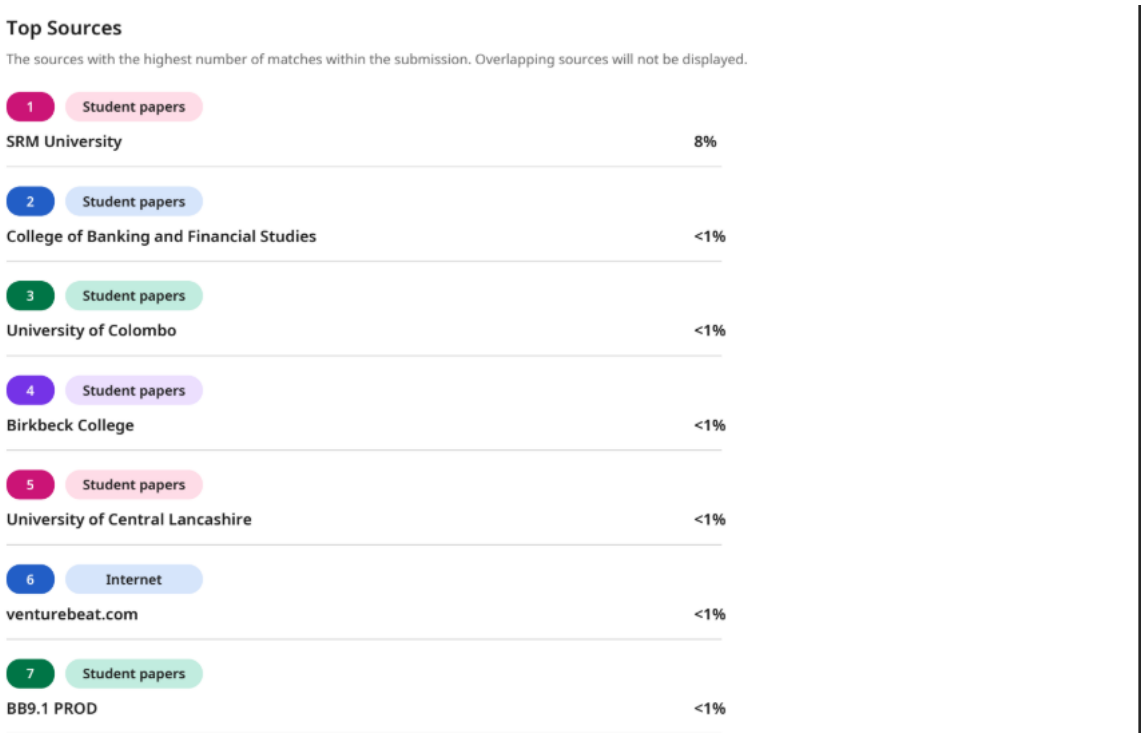

Figure 4.2 PLAGIARISM REPORT