Anitha Ramesh

new Report intro page.docx

Batch 4

Major Project Final Year

SRM Institute of Science & Technology

Document Details

Submission ID

trn:oid:::1:3227923378

Submission Date

Apr 25, 2025, 8:20 AM GMT+5:30

Download Date

Apr 25, 2025, 8:22 AM GMT+5:30

File Name

new_Report_intro_page.docx

File Size

7.4 MB

48 Pages

4,620 Words

28,938 Characters

9% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Match Groups

- **27** Not Cited or Quoted 9%
  Matches with neither in-text citation nor quotation marks
- **1** Missing Quotations 0%
  Matches that are still very similar to source material
- **1** Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- **0** Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

- 2% Internet sources
- 1% Publications
- 7% Submitted works (Student Papers)

Match Groups

- **27** Not Cited or Quoted 9%
  Matches with neither in-text citation nor quotation marks
- **1** Missing Quotations 0%
  Matches that are still very similar to source material
- **1** Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- **0** Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

- 2% Internet sources
- 1% Publications
- 7% Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Student papers	
SRM University		7%
2	Internet	
www.lshtm.ac.uk		<1%
3	Internet	
licindia.in		<1%
4	Student papers	
International University - VNUHCM		<1%
5	Internet	
www.fastercapital.com		<1%
6	Publication	
"Proceedings of Eighth International Congress on Information and Communicati...		<1%
7	Student papers	
Universiti Malaysia Perlis		<1%
8	Student papers	
University of Liverpool		<1%
9	Internet	
www.coursehero.com		<1%
10	Internet	
phdessay.com		<1%

11 Internet

www.iges.or.jp <1%

12 Publication

Jerry L. Hoffman. "Application of the ithink® System Dynamics Software Program... <1%

CHAPTER 1

INTRODUCTION

1.1 Introduction

As the field continues to grow and transform at a fast pace of digital health, there is a growing reliance on technology to aid in early disease detection and self-assessment. With the rise of mobile health apps, wearable devices, and AI-driven tools, patients are becoming more proactive in managing their well-being. However, this shift also brings challenges — especially when users depend on generic search engines or low-accuracy symptom checkers for medical insights. These tools often lack contextual understanding, leading to misinformation and unnecessary panic.

Moreover, the explosion of health-related data from electronic health records (EHRs), wearable sensors, and patient-reported outcomes has created an opportunity for data-driven diagnosis systems that go beyond static symptom matching. By leveraging advanced AI techniques and semantic models, we can connect unprocessed data with actionable insights meaningful medical insights. These intelligent systems offer the potential to identify hidden patterns in complex datasets, adapt to individual patient profiles, and provide accurate recommendations tailored to real-world medical scenarios.

In today's e-health era, individuals frequently use the internet to self-diagnose based on symptoms. However, search engines and basic symptom checkers often fail to account for a person's complete health profile, leading to misinterpretations and anxiety — a condition known as cyberchondria. These tools typically retrieve high-ranking results, not necessarily accurate or personalized medical advice.

The accuracy of existing online symptom checkers is alarmingly low. Studies show that they produce correct diagnoses in only about 34% of cases. This lack of reliability undermines public trust and highlights the urgent need for more precise, context-aware diagnosis tools. To address this issue, this project proposes an intelligent disease prediction system that integrates structured and unstructured health data. Admins can upload patient data, and the system uses an ontology-driven approach enhanced with machine learning (e.g., CNN) to suggest possible conditions.

1.2 Motivation

The motivation for this project originates from the growing public reliance on web-based symptom checkers and their consistent failure to deliver accurate diagnoses. While convenient, most tools lack context-awareness and are unable to handle the diversity in patient health conditions, leading to incorrect or generalized results.

Furthermore, there is a vast volume of medical data being generated in hospitals and health portals, yet much of it remains underutilized. By incorporating both structured (e.g., symptoms, demographics) and unstructured (e.g., patient notes) data, this project aims to create a tool that bridges the gap between data availability and meaningful diagnosis.

Another strong motivator is the need for administrative tools in hospital environments that allow controlled access to sensitive medical data. Our system is currently designed for **admin-only use**, enabling authorized personnel to upload, view, and analyze patient records securely. This ensures both privacy and centralized control while allowing the system to grow and improve with real-world data input.

1.3 Sustainable Development Goal of the Project

This project supports the vision of the United Nation's Sustainable Development Goal 3 (SDG 3), aiming to enhance public health and encourage overall well-being for people of all ages. Our intelligent health diagnosis system contributes to this goal by improving access to reliable health information, enabling early detection of diseases, and reducing misdiagnoses caused by generic symptom checkers.

The system's focus on accurate and data-driven diagnosis promotes **preventive care**, which is a critical component of public health sustainability. By leveraging patient health records and symptom mapping using ontology and AI, the platform supports clinical decision-making and contributes to better patient outcomes.

Moreover, the project supports **universal healthcare access** in digital form by laying the foundation for a scalable tool that can be deployed across hospitals, telemedicine platforms, and even rural healthcare networks in the future.

1.4 Product Vision Statement

Audience:

- **Primary Audience**: Healthcare administrators and analysts seeking to manage and analyze patient data.
- **Secondary Audience**: Doctors, clinical researchers, and future health app users.

Needs:

- Reliable disease prediction based on patient-specific data.
- Accurate diagnosis support using structured and unstructured health data.
- Secure admin-level data access and control.
- Scalable integration with future patient-facing platforms.

Product:

- Core Product: An intelligent, admin-controlled health diagnosis tool using ontology models and CNN for disease prediction.
- Features:
    - Dataset upload and management
    - Preprocessing and symptom mapping
    - Search and diagnosis analysis
    - Diagnostic result logging
    - Secure access for authorized personnel

Values:

- Accuracy: Uses structured medical ontology (HDDO) for reliable predictions.
- Security: Admin-based access control to maintain data privacy.
- Scalability: Designed for integration into broader hospital systems or public health networks.

1.5 Product Goal

The primary goal of this project is to create a **trustworthy, intelligent disease diagnosis system** that improves on existing symptom checkers by integrating advanced machine learning and medical ontology. It aims to provide **high-accuracy, context-aware diagnostic suggestions** based on real patient data.

The system focuses on:

- Supporting healthcare professionals with reliable predictions
- Assisting admins with centralized control of health data
- Enhancing diagnosis accuracy using CNN-based learning models
- Enabling detailed progress tracking of patient diagnostics

In the long term, the system will evolve into a more accessible tool that could integrate patient interfaces, lifestyle tracking, and automated treatment recommendations — aligning with the vision of **smart, personalized digital healthcare**.

1.6 Product Backlog

S.No	User Stories
#US 1	This user story targets the system admin responsible for managing patient health records. The admin wants to log in securely to the web application to access and manage patient records.
#US 2	This user story targets the system admin responsible for retrieving patient information. The admin wants to search for a patient using their unique patient ID. The main benefit is that the admin can quickly access patient health records for review and diagnosis.
#US 3	This user story targets the system admin responsible for generating reports based on patient symptoms. The admin wants to generate a diagnostic report based on a patient's symptoms and medical history. The main benefit is that the system can provide an accurate diagnosis to assist in patient treatment.
#US 4	This user story targets the system admin responsible for keeping medical records up to date. The admin wants to update a patient’s health records with new test results and diagnoses. The main benefit is that patient records will be kept accurate for future reference and treatment.
#US 5	This user story targets the system admin to ensure compliance with data security regulations. The admin wants to ensure that only authorized personnel can access patient records. The main benefit is that patient privacy is maintained, preventing unauthorized access.
#US 6	This user story targets the system admin responsible for ensuring system reliability. The admin wants to monitor the system’s performance and detect errors. The main benefit is that issues can be resolved quickly to ensure smooth system operation.

Table 1.2 UserStories

The product backlog for the Web-Based Health Diagnosis System was organized using Microsoft Planner's Agile Board, as illustrated in Figure 1.1. It contains all the user stories relevant to the project. Each story outlines key functional and non-functional requirements, along with clearly defined acceptance criteria and associated tasks.

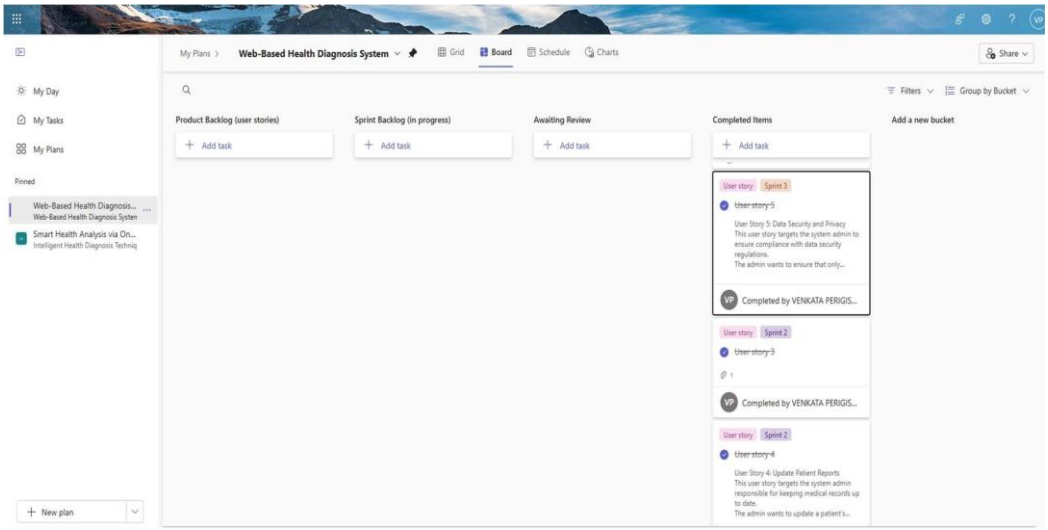


Figure 1.1 MS Planner Board of Web-Based Health Diagnosis System

1.7 Product Release Plan I

The release timeline for the project is illustrated in Figure 1.2

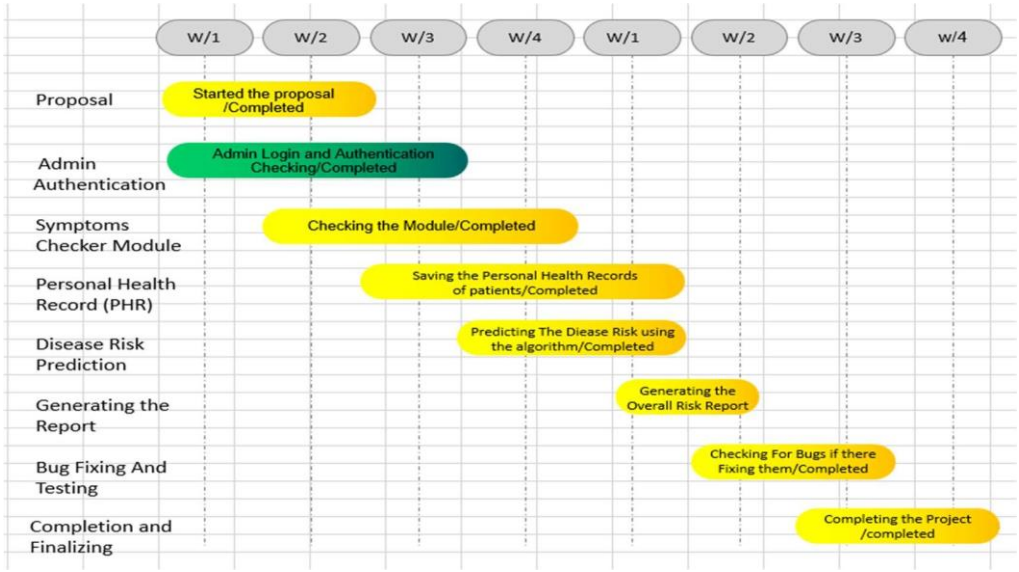


Figure 1.2 Release plan of Web-Based Health Diagnosis System

CHAPTER 2

SPRINT PLANNING AND EXECUTION

2.1 Sprint 1

2.1.1 Sprint Goal with User Stories of Sprint 1

The primary objective of Sprint 1 is to develop the Admin landing page and implement key search functionalities, including viewing datasets, performing preprocessing, and generating reports. Table 2.1 outlines the detailed user stories associated with Sprint 1.

Table 2.1 Detailed User Stories of sprint 1

S.NO	Detailed User Stories
US #1	This user story targets the system admin responsible for managing patient health records. The admin wants to log in securely to the web application to access and manage patient records. The main benefit is that Access to sensitive patient information is restricted to authorized users, safeguarding both privacy and data security.
US #2	This user story targets the system admin responsible for retrieving patient information. The admin wants to search for a patient using their unique patient ID. The main benefit is that the admin can quickly access patient health records for review and diagnosis.

1 The user stories represented on the Planner Board are illustrated in Figures 2.1 and 2.2 below

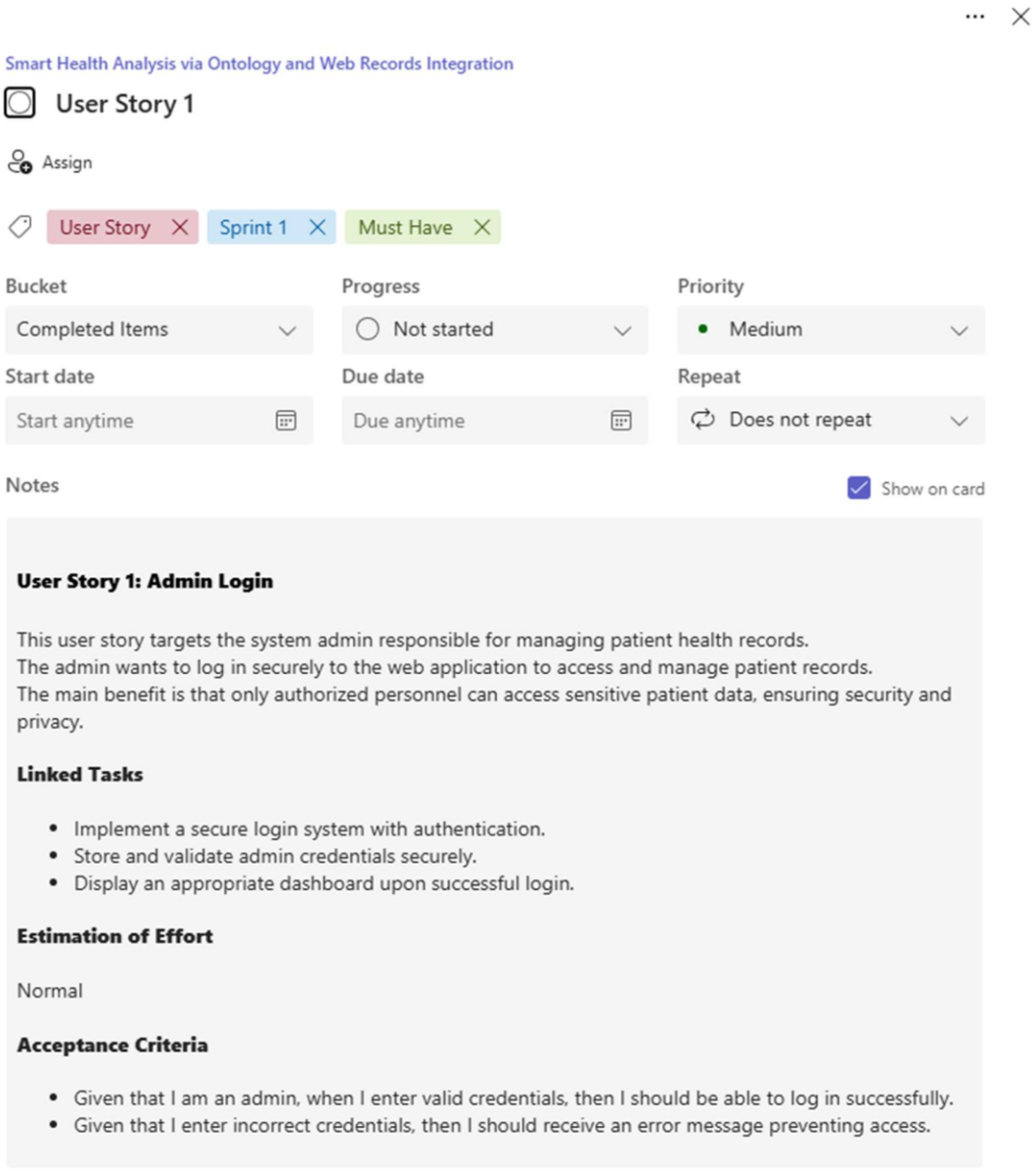


Figure 2.1 image of User-story for Admin Access

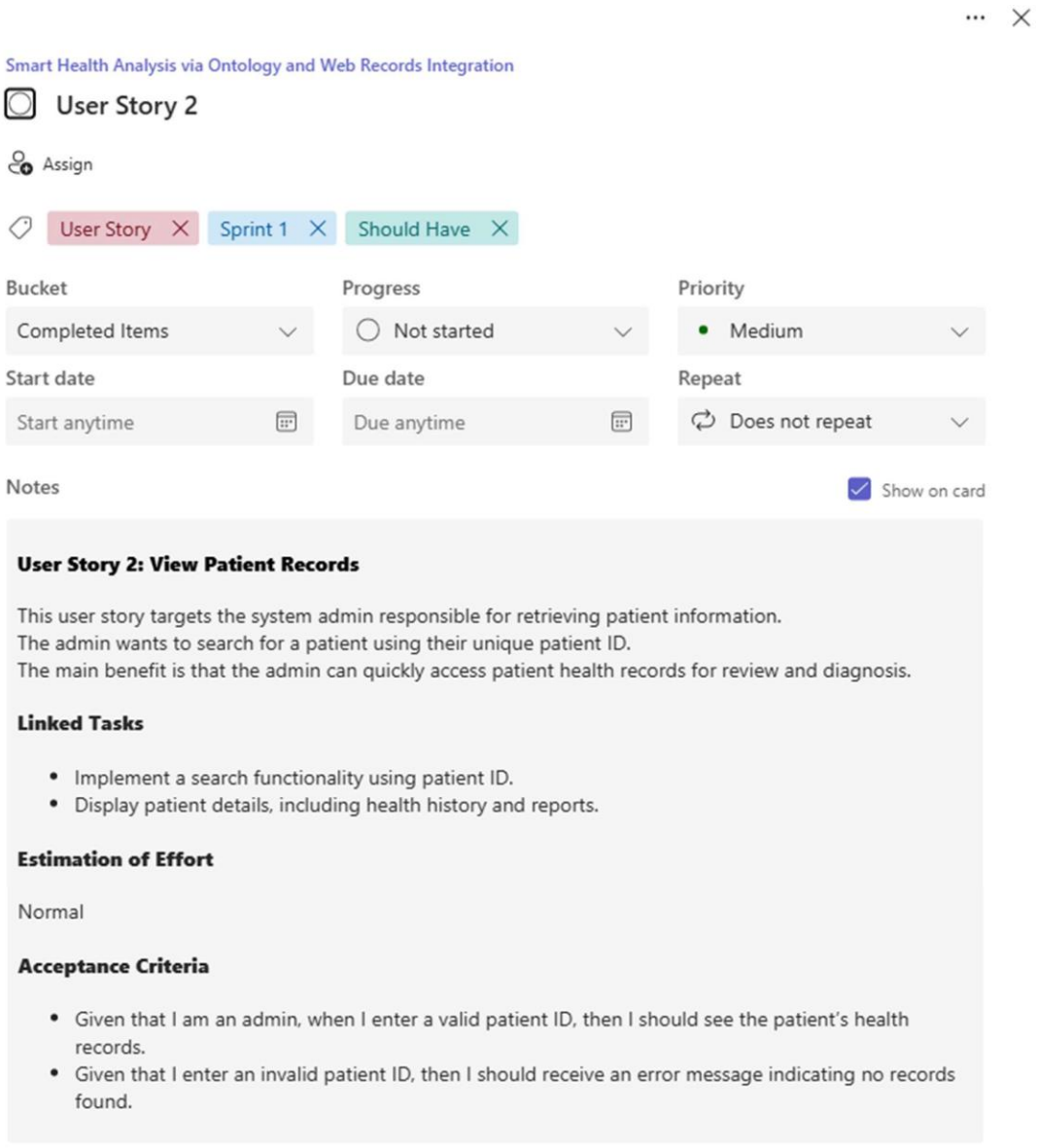


Figure 2.2 user story for Viewing Patients Records

2.1.2 Functional Document

2.1.2.1. Introduction

The Web-Based Health Diagnosis System aims to provide accurate and AI-powered diagnostic results based on patient symptoms and personal health records. This system enables healthcare administrators to manage patient records efficiently and generate intelligent health reports.

2.1.2.2. Product Goal

The goal of this system is to integrate **AI-driven ontology models, CNN, LSTM, and classification algorithms** to enhance the accuracy of health diagnoses while ensuring secure and seamless management of patient data.

Location:

- The system is designed to be deployed **globally**, with an initial focus on **urban healthcare centers, hospitals, clinics, and research institutions** that rely on digital systems for patient management.
- Emphasis is placed on areas with **strong internet infrastructure**, where hospitals or health institutions are actively transitioning to **smart healthcare solutions**. In future phases, the system can also support **rural health networks**,

2.1.2.3. Business Processes

The core business processes handled by the intelligent disease prediction system include:

Admin Registration and Secure Login

- Only authorized admins can register and log into the system.
- Multi-level authentication ensures that sensitive medical data is accessible only to verified personnel.

Dataset Upload and Preprocessing

- Admins can upload patient symptom records, medical histories, and personal health data.

Symptom Mapping and Disease Prediction

- The system uses the **HDDO (Human Disease Diagnosis Ontology)** to map symptoms with potential disease outcomes.
- A **CNN (Convolutional Neural Network)** model analyzes patterns in the data to generate diagnostic suggestions.

Search and Case Analysis

- Admins can query specific patient data, symptoms, or diagnosis logs.
- The system supports filtering and keyword-based search for rapid information retrieval.

Diagnostic Result Logging

- The platform maintains a log of all diagnostic sessions, enabling historical tracking and analysis.
- Logs can be reviewed for research, follow-up care, or refining model accuracy.

Report Generation and Data Export

- The system provides options to generate and export diagnosis summaries and analytical reports.
- These can be used for clinical evaluation, audits, or integration into broader hospital record systems.

2.1.2.4. Features

This project is centered around the development of the following core features:

Feature #1: AI-Powered Diagnosis System

- Uses **CNN & LSTM** to analyze symptoms and medical history for accurate diagnosis.
- Provides ranked diagnostic results for improved healthcare decisions.

Feature #2: Data Preprocessing for Clean Records

- Uses **classification algorithms** to remove null values from datasets.
- Ensures cleaner input data for AI models, improving accuracy.

1. User Stories:

User Story 1: Admin Login

Description: The admin wants to securely log in to the system to access patient records and perform necessary actions.

Acceptance Criteria:

- When valid login credentials are entered, the user should be granted access to the system.
- If the login credentials are incorrect, the system should display an appropriate error message and deny access.

User Story 2: View Patient Records

Description: The admin wants to search for a patient using their unique patient ID to retrieve medical history.

Acceptance Criteria:

- Given that I enter a valid patient ID, then I should see the patient's health records.
- Given that I enter an invalid patient ID, then I should receive an error message indicating no records found.

2.1.2.5. Authorization Matrix

Table 2.2 Access level Authorization Matrix

Role	Access Level
Admin	Full Access to Patient records, diagnosis, data preprocessing, and system monitoring
Patient	Limited access to personal health reports

2.1.2.6. Assumptions

- The system will be AI-integrated using CNN, LSTM, and classification algorithms for diagnosis.
- Admins will manage patient records, preprocess data, and generate reports.
- Patient data will be securely stored and accessed only by authorized users.

2.1.3 Architecture Document

2.1.3.1. Application

Microservices:

Although the current implementation is monolithic (due to usage of JDK, NetBeans), the system can be **scaled into microservices** by logically separating functionalities such as:

- **User Management**
- **Patient Information Handling**
- **Diagnosis Processing**
- **Report Generation**
- **System Logging**

This structure allows independent deployment and scaling in the future.

2.1.3.2 System Architecture-

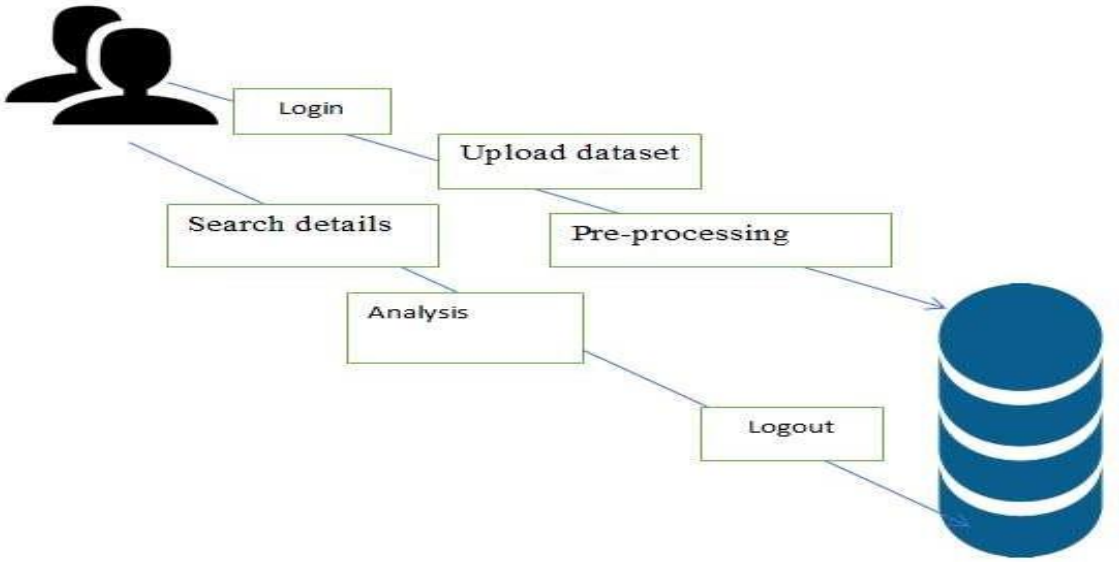


Figure 2.3 System Architecture Diagram

UML DIAGRAMS:

Fig 2.4 Entity-Relationship (ER) Diagram

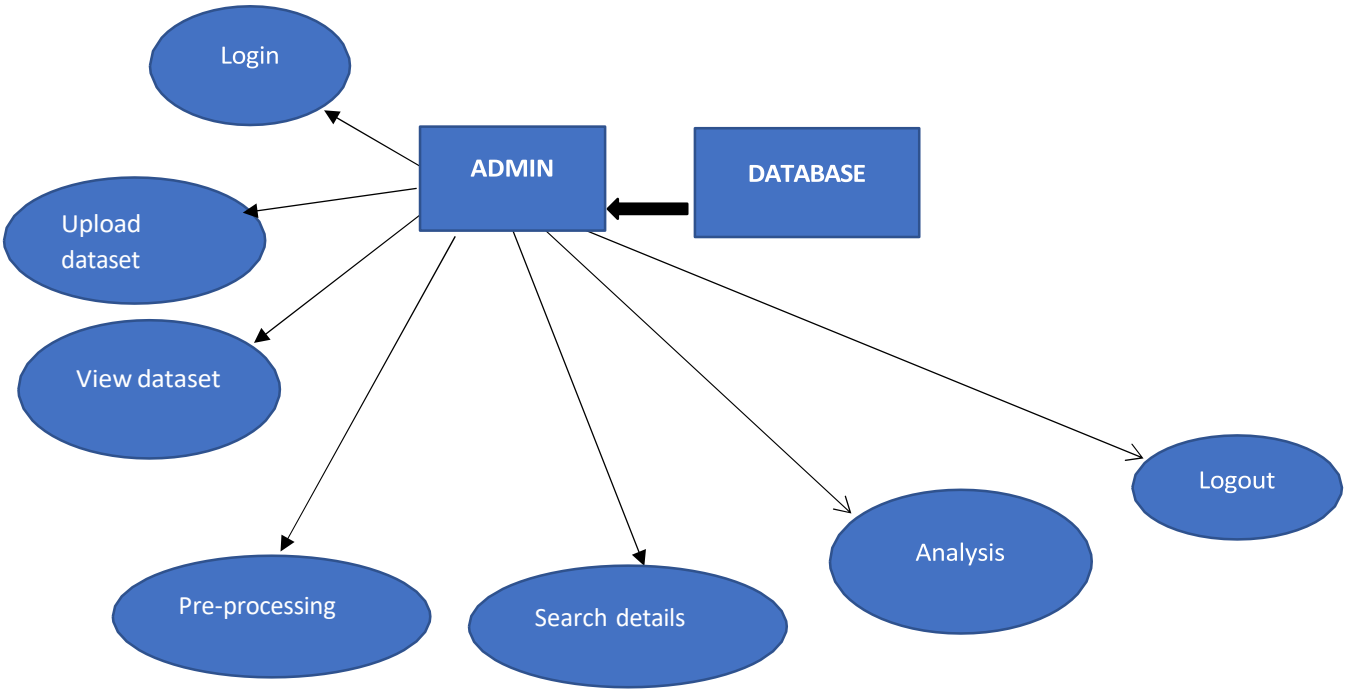


Fig 2.5 Schema Design

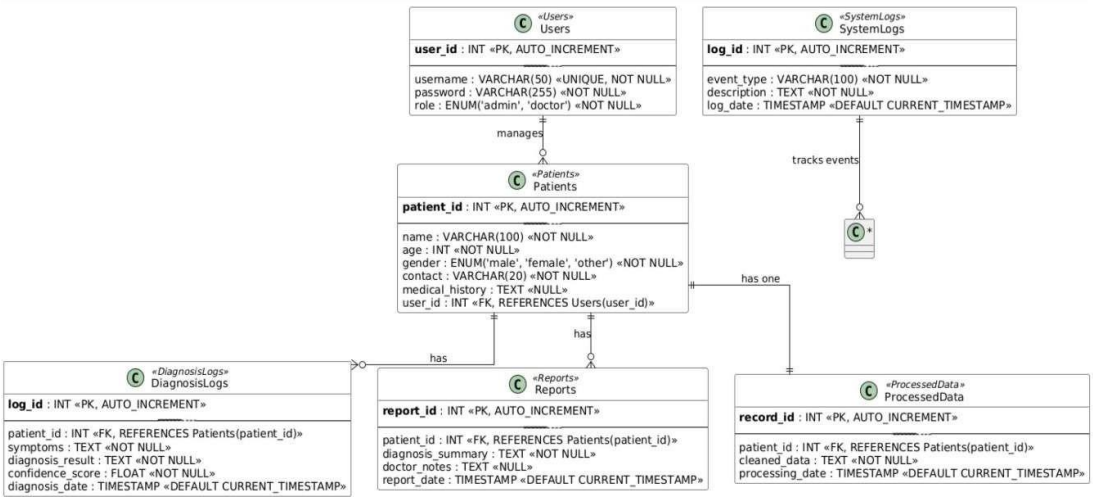


Fig 2.6 Use case Diagram

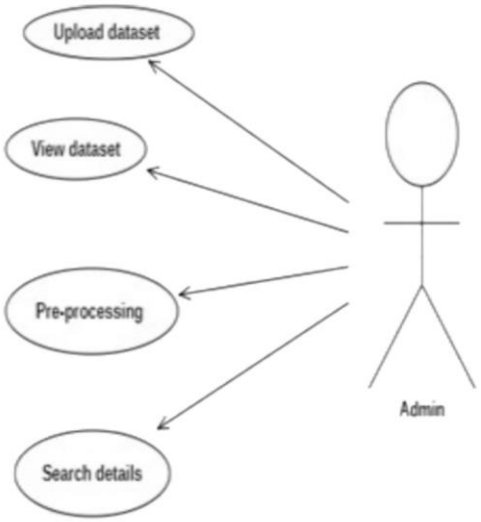


Fig 2.7 Class Diagram

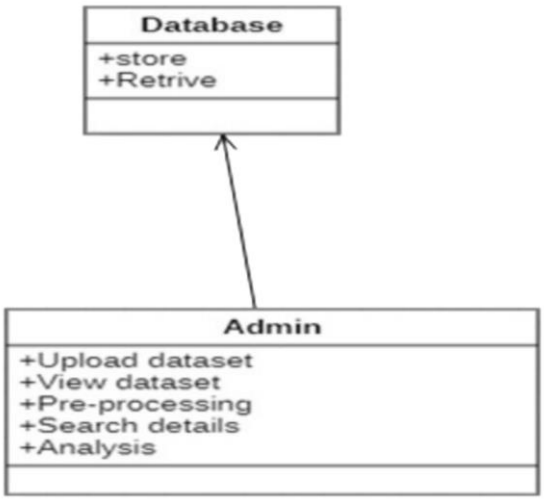


Fig 2.8 Communication Diagram

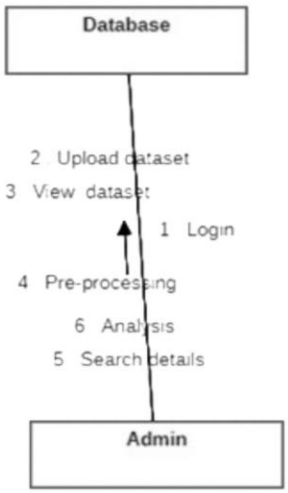


Fig 2.9 Deployment Diagram

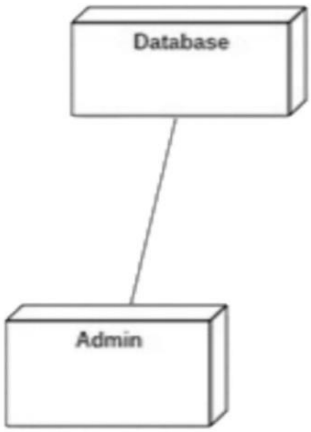
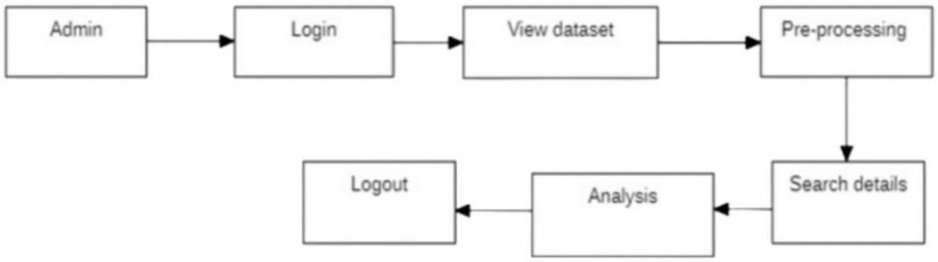


Fig 3.0 Data Flow Diagram

Level 0:

Level 1:



2.1.3.3. Data Exchange Contract:

1. Frequency of Data Exchanges

- **Real-Time Exchange:** Between front-end (admin interface) and backend services (e.g., report generation).
- **On-Demand:** When admin requests processed reports or logs.
- **Periodic:** For system backup or batch processing (if extended in future).

2. Data Sets

The major datasets involved:

- **Patient Records**
- **Diagnosis Logs**
- **Reports**
- **Processed Data**
- **User Logs**

Each dataset includes critical fields defined in your ER and schema diagrams. Example:

- Patients (name, age, gender, contact, medical_history)
- Diagnosis Logs (symptoms, result, score, date)
- Processed Data (cleaned symptoms, processing date)

2.1.4 UI DESIGN

Figure 3.1 UI Design for Landing page

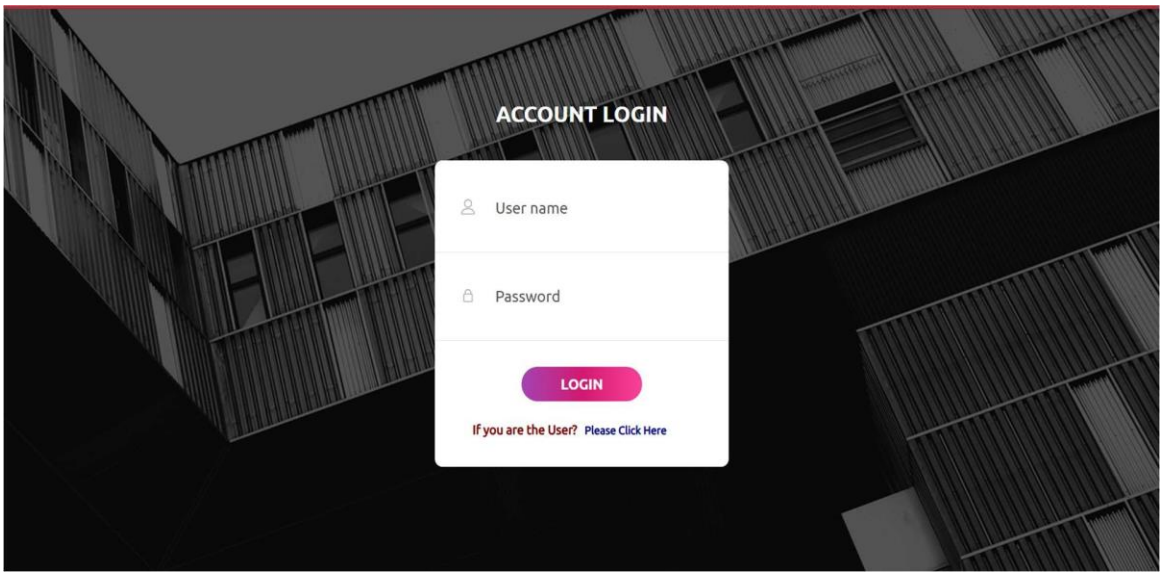


Figure 3.2 UI design for login page

2.1.5 Functional Test Cases

Table 2.3 presents a detailed overview of the functional test cases.

		Functional Test Case Template				
Feature	**Test Case**	**Steps to execute test case**	**Expected Output**	**Actual Output**	**Status**	**More Information**
User Login	**Valid User Login**	Open the application's login page. Enter a valid username. Enter a valid password. Click on the "Login" button.	The user should be successfully logged into the system. The application should redirect the user to the home page.	The user is successfully logged in. The application redirects the user to the home page.	Pass	No error messages are displayed. The user profile information is correctly displayed on the home page. Check if the login time is recorded for the user.
User Login	**Invalid User Login**	Open the application's login page. Enter an invalid username/password. Click on the 'Login' button.	Invalid User Please Enter the Correct Deatils	Invalid User Please Enter the Correct Deatils	Pass	Ensure that incorrect credentials do not bypass authentication.
Password Recovery	**Forgot Password**	Open the application's login page. Click on the 'Forgot Password' link. Enter a valid email address. Click on the 'Submit' button.	The system should send a password reset email to the provided email address. The user should receive an email with reset instructions.			Verify the content of the password reset email. Check that the link in the email redirects the user to the password reset page.
Diagnosis	**Disease Prediction**	Upload Dataset in the input field. Click on Search Details.	The AI model should return a probable diag	The AI model should return a probable	Pass	Ensure AI diagnosis is accurate and based on dataset.
Report Generation	**Generate Patient R**	Enter patient ID. Click on 'Generate'.	A report should be generated containing the	A report should be generated containi	Pass	Check that the report is correctly formatted and contains all necessary details.
Precautions	**Generate precautic**	After Generating the Full Details About th	It Should Generate precautions according to	It Should Generate precautions accord	Pass	Check if the Input Dataset has the Correct Details of Patient

2.1.6 Daily Call Progress

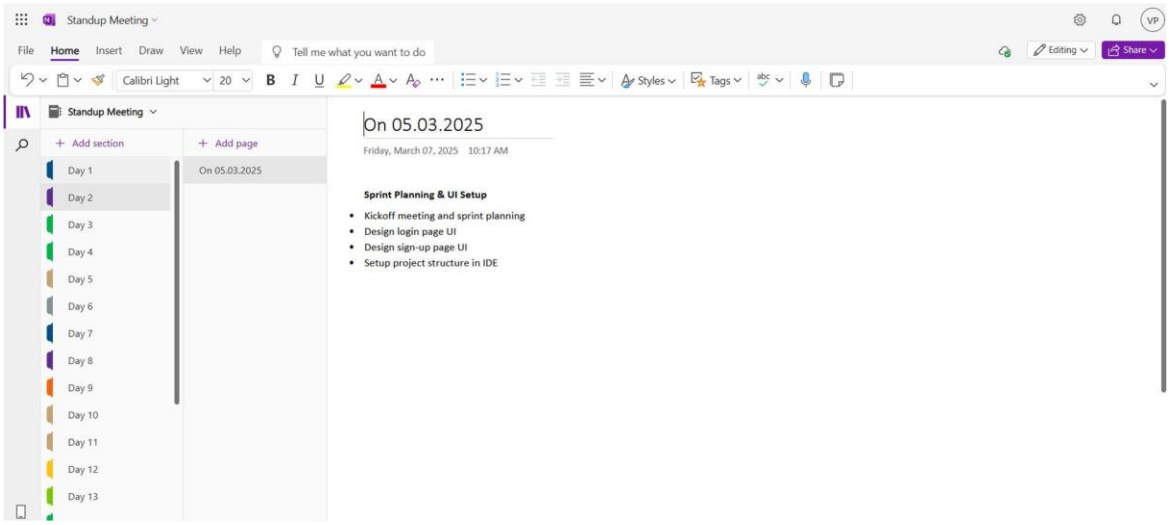


Figure 3.3 Details of Standup meetings

2.1.7 Committed Vs Completed User Stories

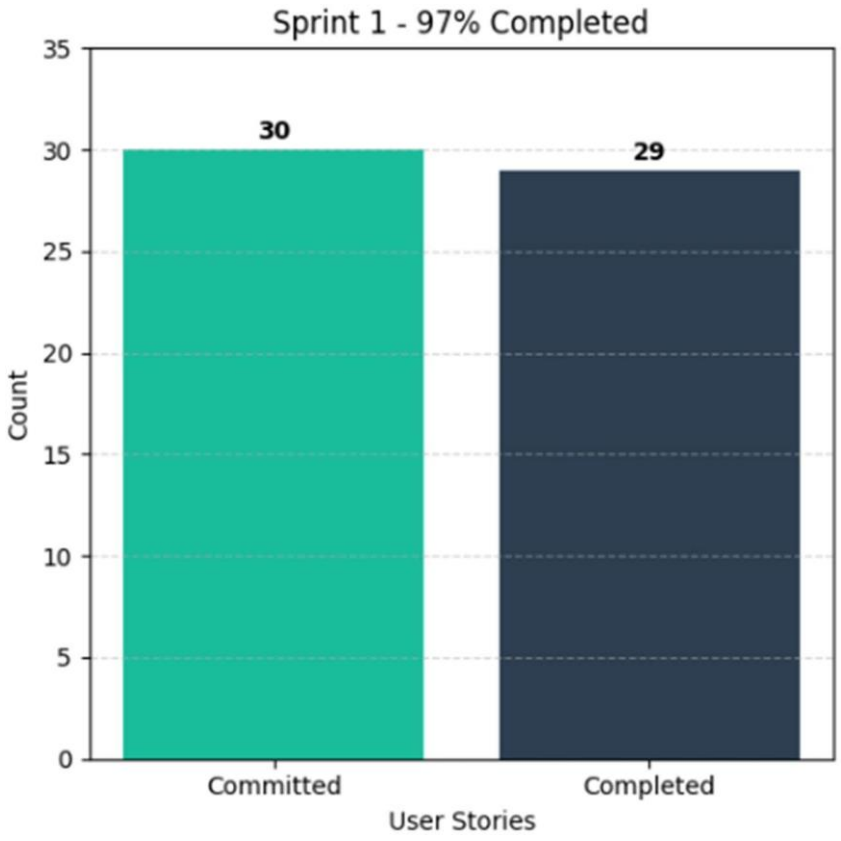


Figure 3.4 Bar graph depicting the comparison between Committed and Completed User Stories.

2.1.8 Sprint Retrospective

	Sprint Retrospective			
What went well	**What went poorly**	**What ideas do you have**	**How should we take action**	
*This section highlights the **successes and positive outcomes from the sprint**. It helps the team recognize achievements and identify practices that should be continued.*	*This section identifies the **challenges, roadblocks, or failures encountered during the sprint**. It helps pinpoint areas that need improvement or change.*	*This section is for brainstorming **new approaches, tools, or strategies to enhance the team's efficiency, productivity, or project outcomes.***	*This section outlines specific steps or solutions to address the issues and implement the ideas discussed, ensuring continuous improvement in future sprints.*	*Guidelines*
Successfully integrated the SQLyog database with NetBeans and JDK.	Faced initial issues with database connectivity.	Document database setup and connection steps clearly.	Create a DB integration checklist for future use.	Highlight major accomplishments and working solutions.
Admin login and session management works as expe	Session timeout behavior not fully optimized.	Use session timeout logic based on user activity.	Review session lifecycle settings in web.xml.	on authentication and security
Admin dashboard loads and displays Architetcture di	Some layout elements are not responsive on smaller	Apply responsive CSS using Bootstrap or media queries.	Test UI on multiple screen sizes and browsers.	sability and cross-platform com
SQL queries for record management performed well	Query execution slowed down with large data sets.	Optimize SQL with indexes and pagination.	Profile slow queries and refactor where needed.	end/database performance im
Team collaboration and task tracking was effective.	Delays due to unclear role assignments during modul	Assign responsibilities in advance during planning meeti	Use a shared task board like Trello or Planner.	improvements in team workflo
Basic AI-based diagnosis logic integrated with datase	Preprocessing required manual cleanup due to null v	Automate preprocessing steps using Java filters.	Write utility functions for null check and formatting.	e clean data handling for AI fun

Figure 3.5 Detailed overview of Sprint Retrospective for the Sprint 1

2.2 SPRINT 2

2.2.1 Sprint Goal with User Stories of Sprint 2

The Goal of the second sprint is targets the system admin responsible for generating reports based on patient symptoms. And keeping medical records up to date.

The following table 2.4 below outlines the detailed user stories for Sprint 2.

Table 2.4 provides the comprehensive user stories for Sprint 2.

S.NO	Detailed User Stories
US #3	This user story targets the system admin responsible for generating reports based on patient symptoms. The admin wants to generate a diagnostic report based on a patient's symptoms and medical history. The main benefit is that the system can provide an accurate diagnosis to assist in patient treatment.
US #4	This user story targets the system admin responsible for keeping medical records up to date. The admin wants to update a patient’s health records with new test results and diagnoses. The main benefit is that patient records will be kept accurate for future reference and treatment.

The representation of user stories on the Planner Board is shown in Figures 2.9 and 3.0 below.

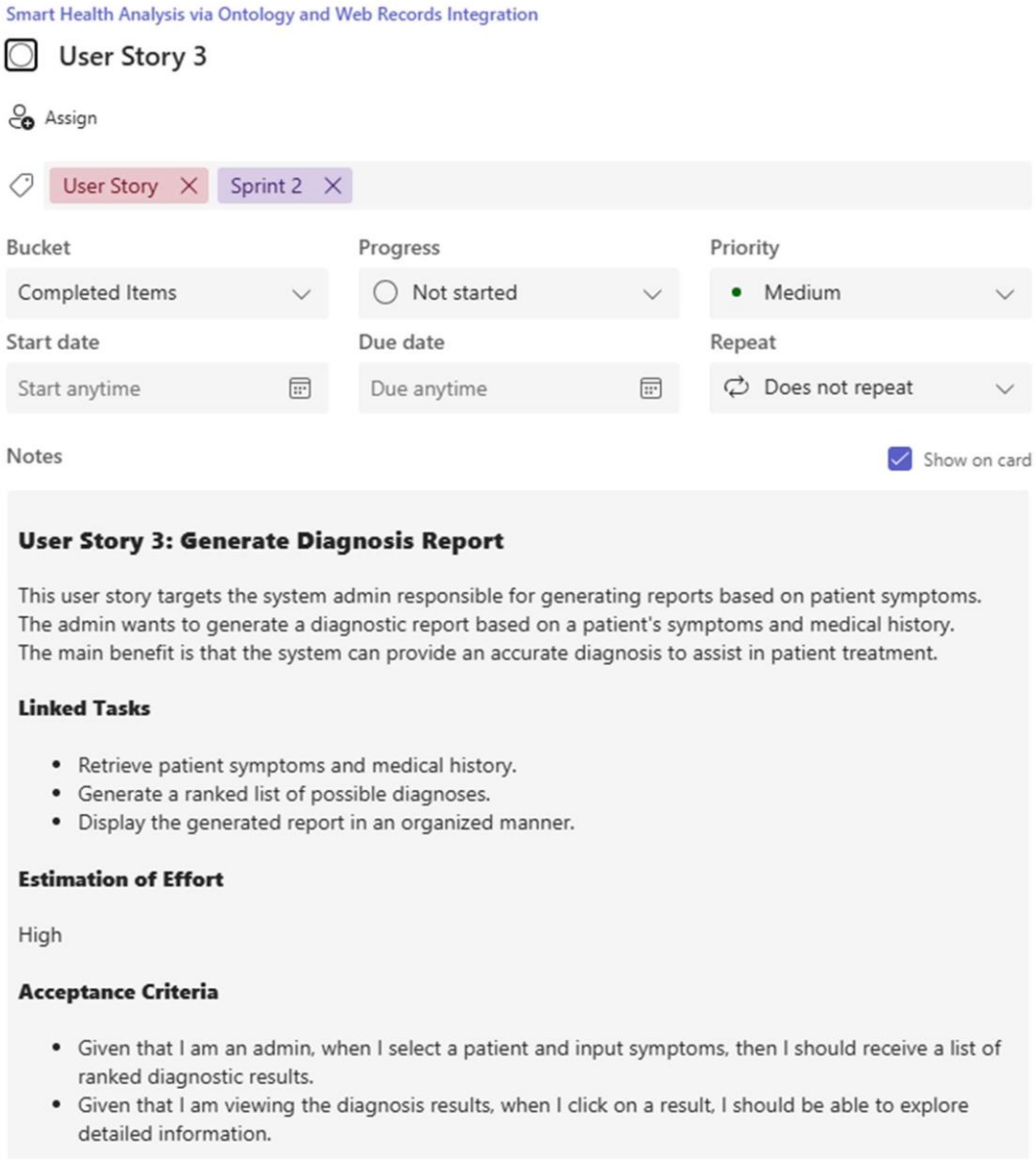


Figure 3.6 User Story for Gnerating Diagnosis Report

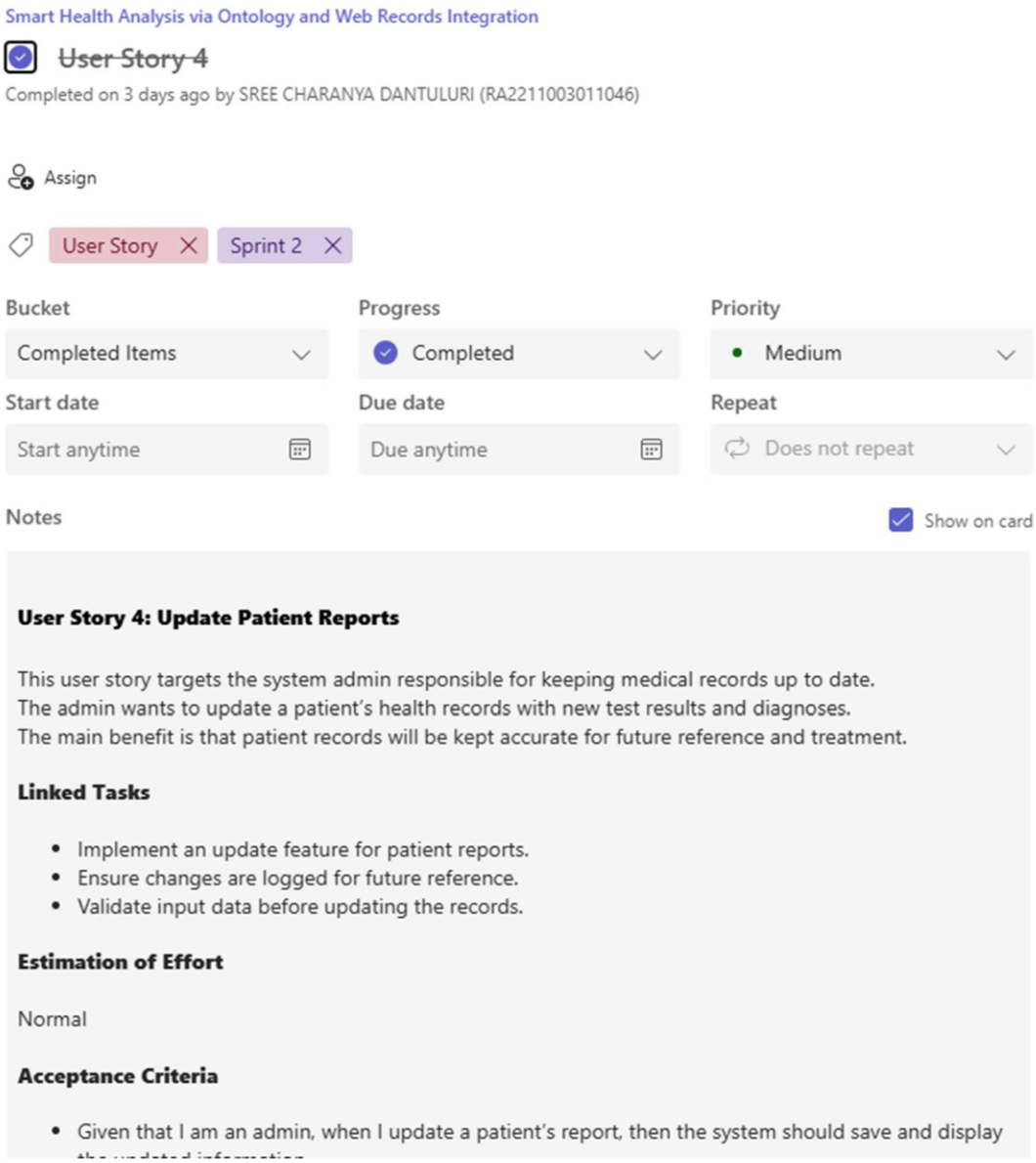


Figure 3.7 user story for Updating Patient Records

2.2.2 Functional Document

2.2.2.1 Introduction

In Sprint 2, the focus is on enhancing the intelligence and usability of the Web-Based Health Diagnosis System by introducing features for updating patient records and generating AI-powered diagnosis reports. These capabilities empower healthcare administrators to maintain up-to-date records and deliver more accurate, symptom-based reports for patient treatment.

2.2.2.2 Product Goal

This sprint aims to:

- Enable accurate diagnosis report generation using patient symptoms and medical history.
- Allow authorized admins to update patient reports with new findings.
- Ensure every report reflects the most recent and relevant patient health information.

2.2.2.3 User Stories

User Story 3: Generate Diagnosis Report

- **Description:** The admin wants to input a patient's symptoms and history to generate a diagnostic report.
- **Acceptance Criteria:**

  Given that I enter a valid patient ID and symptoms, then I should receive a ranked list of possible diagnoses.

  Given that I select a diagnosis result, I should see more details about the condition.

- **Linked Tasks:**
  - Retrieve patient symptoms and medical history.
  - Generate a ranked list of possible diagnoses.
  - Display the generated report in an organized manner.

- Estimation of Effort: High

User Story 4: Update Patient Reports

- **Description:** The admin wants to update a patient's report with new test results.
- **Acceptance Criteria:**
    - Given that I am an admin, when I update a patient's report, the system should save and display the updated information.
    - Given that I make an incorrect or incomplete update, then I should receive an error message.
- **Linked Tasks:**
    - Implement an update feature for patient reports.
    - Ensure changes are logged for future reference.
    - Validate input data before updating the records.
    - Estimation of Effort: Normal

2.1.2.5. Authorization Matrix

Table 2.5 illustrates the Access Level Authorization Matrix.

Role	**Access Level**
Admin	Full Access to Patient records, diagnosis, data preprocessing, and system monitoring
Patient	Limited access to personal health reports

2.1.3 Architecture Document

2.1.4.4.1 Application

Microservices Consideration for Sprint 2: While the base system is monolithic, Sprint 2 introduces additional functionalities that can be modularized into the following microservices:

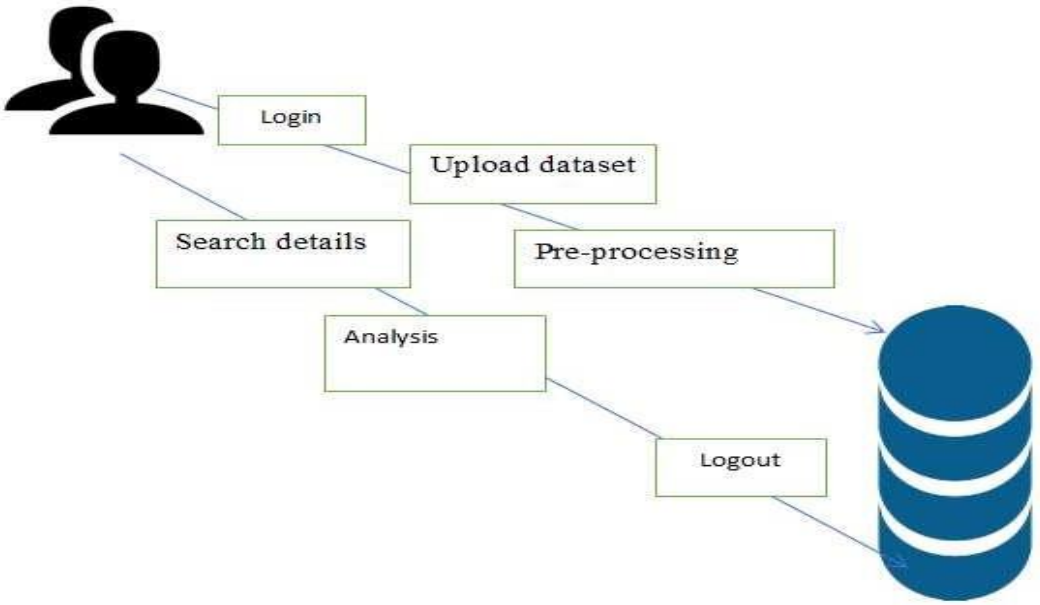
- Report Generation Service: Isolated service for generating diagnosis reports.
- Patient Update Service: Handles updates and validation of patient data.

2.1.4.4.2 System Architecture

Updated System Architecture in Sprint 2 includes:

- Integration between Admin UI and Report Generation Engine.
- RESTful APIs for patient data updates.
- Internal communication between Diagnosis Engine and Knowledge Base (HDDO).

Figure 3.8 Architecture Diagram for Sprint 2.



2.1.4.4.3 Data Exchange Contract

1. Frequency of Data Exchanges:

- Real-Time: On submission of symptoms for diagnosis and update of patient reports.
- On-Demand: When admin requests patient history or specific diagnosis reports.

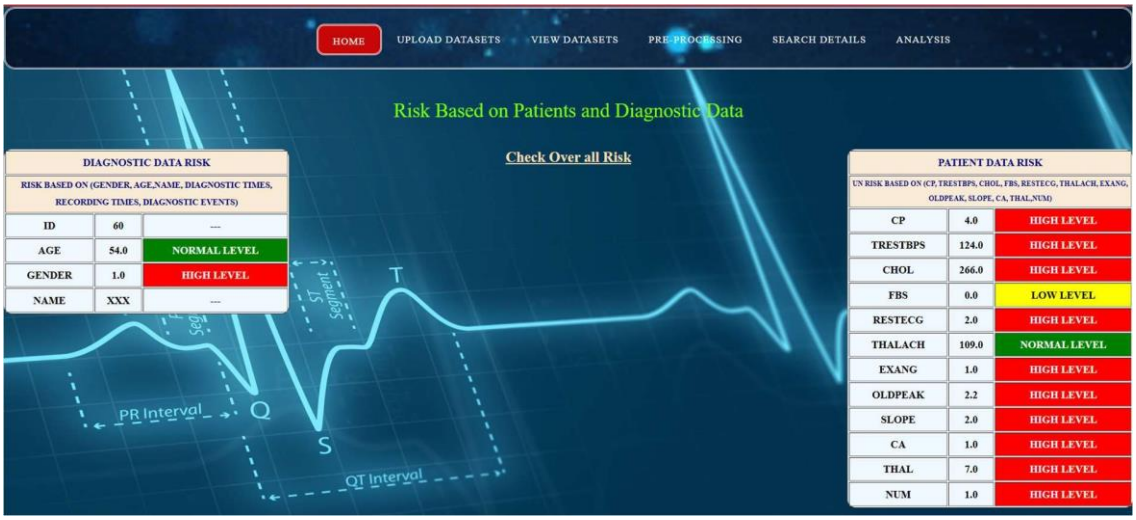
2. Data Sets Updated:

- Diagnosis Reports (ranked results, condition details, timestamps).
- Patient Records (updated symptoms, new entries).
- Audit Logs (who made updates, when, and what changed).

2.2.4 UI Design

PATIENT ID	DIAGNOSIS LEVEL	RECORDING TIMES	DIAGNOSIS EVENTS	DIAGNOSIS TIMES	COUNSELLING TIMES	FAILURE RATE
1.0	1.0	1.0	145.0	233.0	1.0	2.0
2.0	1.0	4.0	160.0	286.0	0.0	2.0
3.0	1.0	4.0	120.0	229.0	0.0	2.0
4.0	1.0	3.0	130.0	250.0	0.0	0.0
6.0	0.0	2.0	130.0	204.0	0.0	2.0
7.0	1.0	2.0	120.0	236.0	0.0	0.0
8.0	0.0	4.0	140.0	268.0	0.0	2.0
9.0	0.0	4.0	120.0	354.0	0.0	0.0
10.0	1.0	4.0	130.0	254.0	0.0	2.0
11.0	1.0	4.0	140.0	203.0	1.0	2.0
12.0	1.0	4.0	140.0	192.0	0.0	0.0
13.0	0.0	2.0	140.0	294.0	0.0	2.0
14.0	1.0	3.0	130.0	256.0	1.0	2.0

Figure 3.9 UI design for Viewing, maintaining and Updating the dataset



DIAGNOSTIC DATA RISK		
RISK BASED ON (GENDER, AGE,NAME, DIAGNOSTIC TIMES, RECORDING TIMES, DIAGNOSTIC EVENTS)		
ID	60	---
AGE	54.0	NORMAL LEVEL
GENDER	1.0	HIGH LEVEL
NAME	XXX	---

PATIENT DATA RISK		
UN RISK BASED ON (CP, TRESTBPS, CHOL, FBS, RESTECG, THALACH, EXANG, OLDPEAK, SLOPE, CA, THAL,NUM)		
CP	4.0	HIGH LEVEL
TRESTBPS	124.0	HIGH LEVEL
CHOL	266.0	HIGH LEVEL
FBS	0.0	LOW LEVEL
RESTECG	2.0	HIGH LEVEL
THALACH	109.0	NORMAL LEVEL
EXANG	1.0	HIGH LEVEL
OLDPEAK	2.2	HIGH LEVEL
SLOPE	2.0	HIGH LEVEL
CA	1.0	HIGH LEVEL
THAL	7.0	HIGH LEVEL
NUM	1.0	HIGH LEVEL

Figure 4.0 UI design for generating the Report

2.2.5 Functional Test Cases

Functional Test Case Template						
Feature	Test Case	Steps to execute test case	Expected Output	Actual Output	Status	More Information
User Login	Valid User Login	Open the application's login page. Enter a valid username. Enter a valid password. Click on the "Login" button.	The user should be successfully logged into the system. The application should redirect the user to the home page.	The user is successfully logged in. The application redirects the user to the home page.	Pass	No error messages are displayed. The user profile information is correctly displayed on the home page. Check if the login time is recorded for the user.
User Login	Invalid User Login	Open the application's login page. Enter an invalid username/password. Click on the 'Login' button.	Invalid User Please Enter the Correct Deatils	Invalid User Please Enter the Correct Deatils	Pass	Ensure that incorrect credentials do not bypass authentication.
Password Recovery	Forgot Password	Open the application's login page. Click on the 'Forgot Password' link. Enter a valid email address. Click on the 'Submit' button.	The system should send a password reset email to the provided email address. The user should receive an email with reset instructions.			Verify the content of the password reset email. Check that the link in the email redirects the user to the password reset page.
Diagnosis	Disease Prediction	Upload Dataset in the input field. Click on Search Details.	The AI model should return a probable diag	The AI model should return a probable	Pass	Ensure AI diagnosis is accurate and based on dataset.
Report Generation	Generate Patient R	Enter patient ID. Click on 'Generate'.	A report should be generated containing the	A report should be generated containi	Pass	Check that the report is correctly formatted and contains all necessary details.
Precautions	Generate precautic	After Generating the Full Details About th	It Should Generate precautions according to	It Should Generate precautions accord	Pass	Check if the Input Dataset has the Correct Details of Patient

Table 2.6 Detailed Functional Test Case

2.2.6 Daily Call Progress

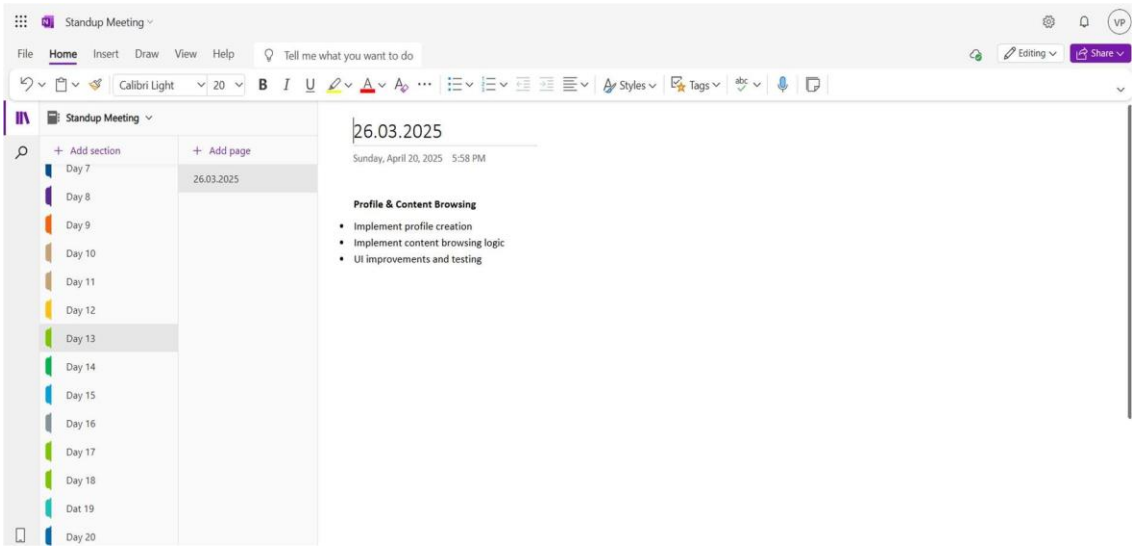


Figure 4.1 Standup Meeting

2.2.7 COMMITTED Vs COMPLETED USER STORIES

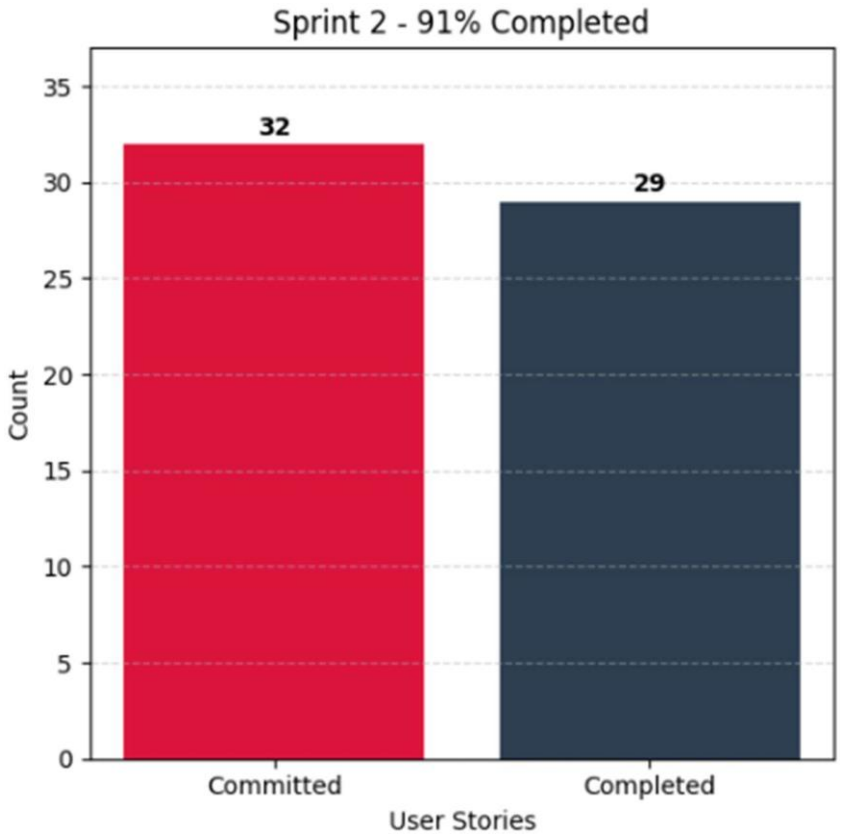


Figure 4.2 presents a bar graph comparing Committed and Completed User Stories.

2.2.8 Sprint Retrospective

	Sprint Retrospective			
What went well	**What went poorly**	**What ideas do you have**	**How should we take action**	
*This section highlights the **successes and positive outcomes from the sprint**. It helps the team recognize achievements and identify practices that should be continued.*	*This section identifies the **challenges, roadblocks, or failures encountered during the sprint**. It helps pinpoint areas that need improvement or change.*	*This section is for brainstorming **new approaches, tools, or strategies to enhance the team's effciency, productivity, or project outcomes.***	*This section outlines specific steps or solutions to address the issues and implement the ideas discussed, ensuring continuous improvement in future sprints.*	*Guidelines*
Successfully integrated the SQLyog database with NetBeans and JDK.	Faced initial issues with database connectivity.	Document database setup and connection steps clearly.	Create a DB integration checklist for future use.	Highlight major accomplishments and working solutions.
Admin login and session management works as expe	Session timeout behavior not fully optimized.	Use session timeout logic based on user activity.	Review session lifecycle settings in web.xml.	on authentication and security
Admin dashboard loads and displays Architetcture di	Some layout elements are not responsive on smaller	Apply responsive CSS using Bootstrap or media queries.	Test UI on multiple screen sizes and browsers.	sability and cross-platform com
SQL queries for record management performed well	Query execution slowed down with large data sets.	Optimize SQL with indexes and pagination.	Profile slow queries and refactor where needed.	end/database performance im
Team collaboration and task tracking was effective.	Delays due to unclear role assignments during modul	Assign responsibilities in advance during planning meeti	Use a shared task board like Trello or Planner.	improvements in team workflo
Basic AI-based diagnosis logic integrated with datase	Preprocessing required manual cleanup due to null v	Automate preprocessing steps using Java filters.	Write utility functions for null check and formatting.	e clean data handling for AI fun

Figure 4.3 showcases the Sprint Retrospective for Sprint 2.

2.3 Sprint 3

2.3.1 Sprint Goal with User Stories of Sprint 3

Sprint 3 focuses on strengthening the security and performance of the Web-Based Health Diagnosis System. It ensures sensitive medical data is protected and the system operates efficiently with real-time monitoring and error detection.

Table 2.4 below outlines the detailed user stories for Sprint 3.

Table 2.7 provides the detailed user stories for Sprint 3.

S.NO	Detailed User Stories
US #5	This user story targets the system admin to ensure compliance with data security regulations. The admin wants to ensure that only authorized personnel can access patient records. The main benefit is that patient privacy is maintained, preventing unauthorized access
US #6	This user story targets the system admin responsible for ensuring system reliability. The admin wants to monitor the system's performance and detect errors. The main benefit is that issues can be resolved quickly to ensure smooth system operation.

The representation of user stories on the Planner Board is shown in Figures 3.7 and 3.8 below.

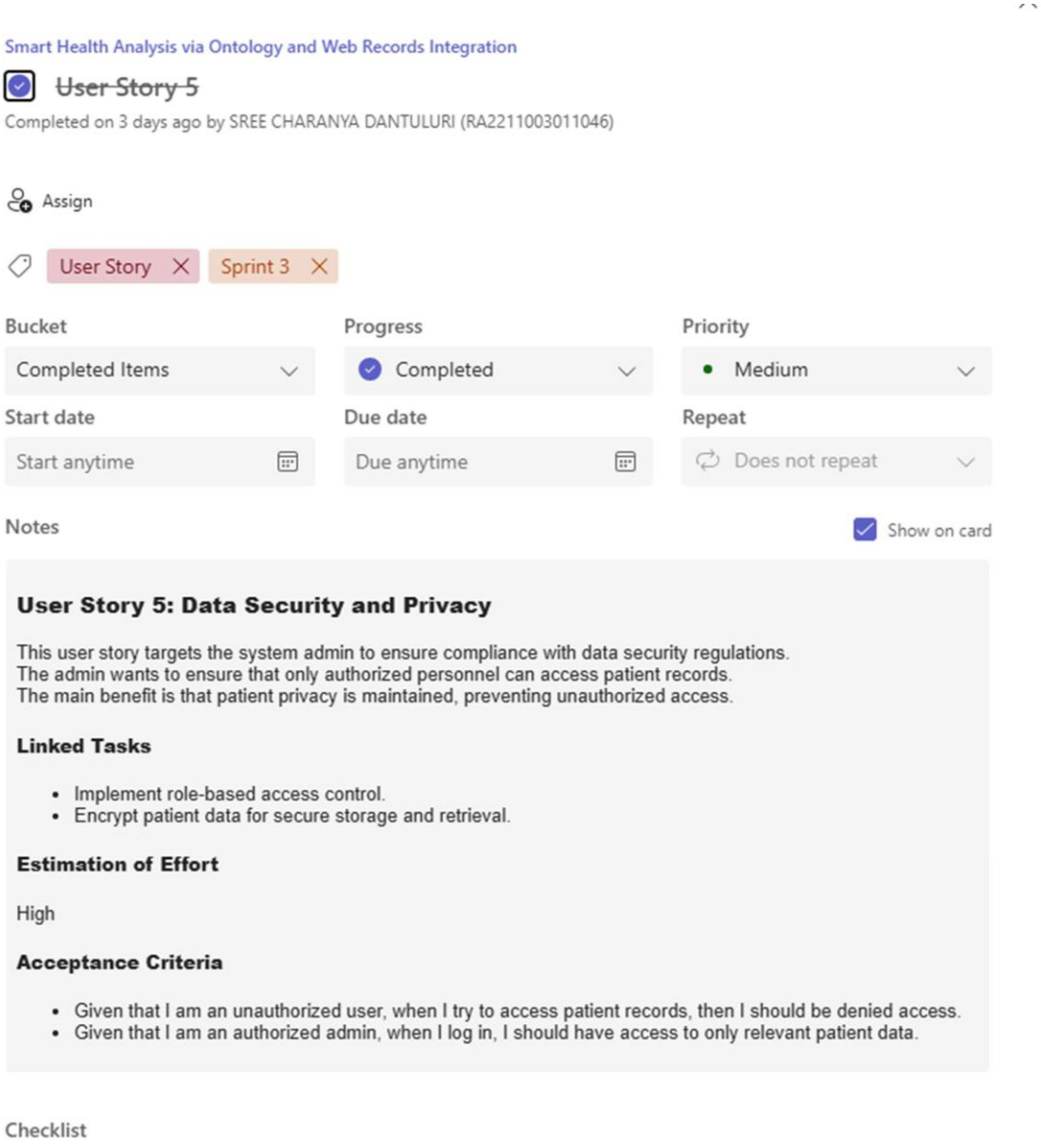


Figure 4.4 User Story for Data Security and Privacy

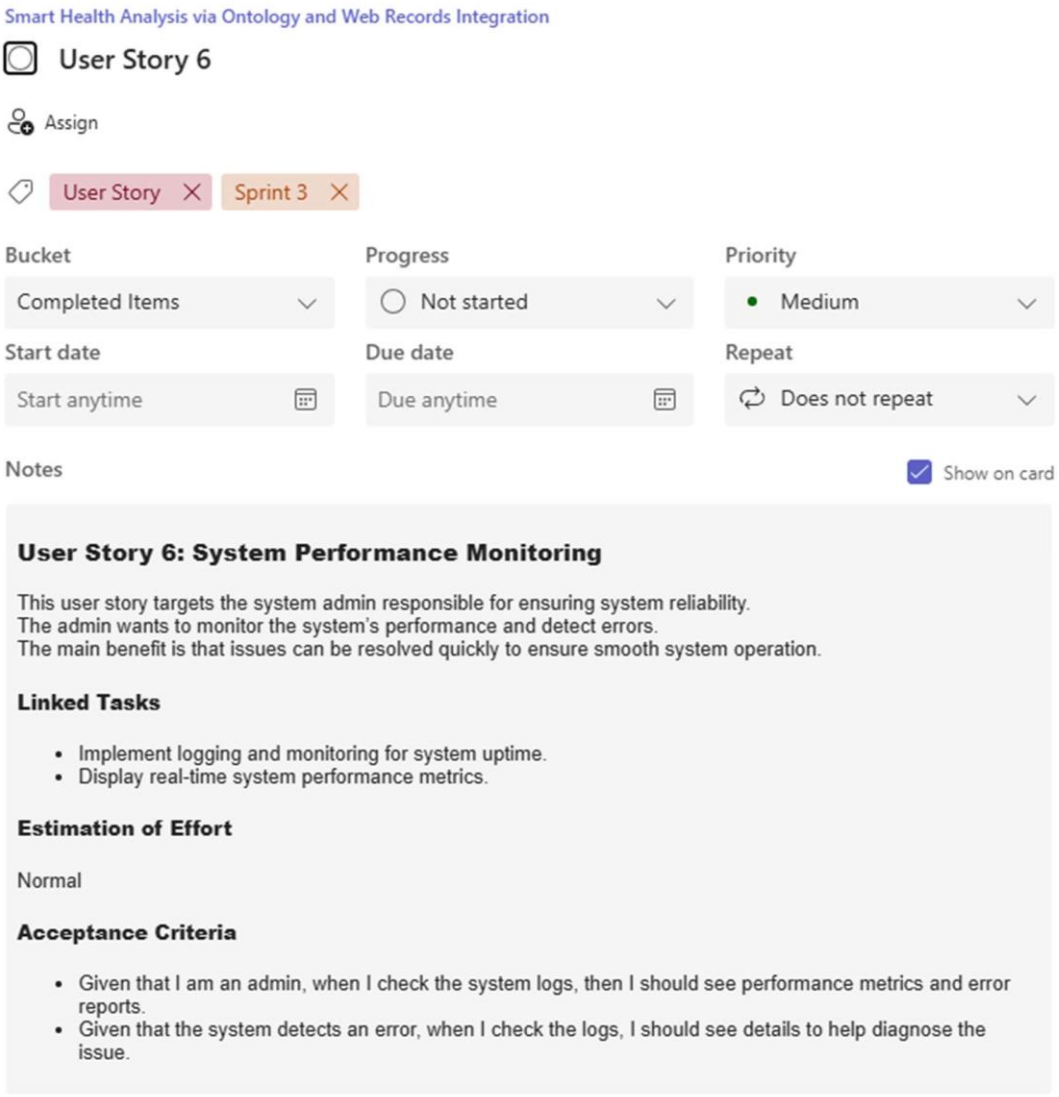


Figure 4.5 user story for Checking System Performance Monitoring

2.3.2 Functional Document

2.3.2.1 Introduction

Sprint 3 focuses on strengthening the security and performance of the Web-Based Health Diagnosis System. It ensures sensitive medical data is protected and the system operates efficiently with real-time monitoring and error detection.

2.3.2.2 Product Goal

This sprint aims to:

- Ensure secure access to patient records via role-based access.
- Encrypt all sensitive patient data.
- Monitor the health of the application and identify any operational issues.

2.3.2.3 User Stories

User Story 5: Data Security and Privacy

- **Description**: The admin wants to ensure only authorized personnel can access patient records.
- **Acceptance Criteria**:
    - Given that I am an unauthorized user, when I try to access patient records, then I should be denied access.
    - Given that I am an authorized admin, when I log in, I should have access to only relevant patient data.
- **Linked Tasks**:
    - Implement role-based access control.
    - Encrypt patient data for secure storage and retrieval.
- **Estimation of Effort**: High

User Story 6: System Performance Monitoring

- **Description**: The admin wants to track system performance and detect issues.
- **Acceptance Criteria**:

- Given that I am an admin, when I check the system logs, I should see performance metrics and error reports.
- Given that the system detects an error, I should see detailed logs to help diagnose the issue.

- **Linked Tasks**:
  - Implement logging and monitoring for system uptime.
  - Display real-time system performance metrics.
- **Estimation of Effort**: Normal

2.1.2.5. Authorization Matrix

Table 2.8 illustrates the Access Level Authorization Matrix.

Role	Access Level
Admin	Full Access to Patient records, diagnosis, data preprocessing, and system monitoring
Patient	Limited access to personal health reports

2.1.2.6. Assumptions

- The system will be AI-integrated using CNN, LSTM, and classification algorithms for diagnosis.
- Admins will manage patient records, preprocess data, and generate reports.

2.3.2.4 Architecture Document

2.3.2.4.1 Application

Microservices Consideration for Sprint 3: Sprint 3 introduces secure access and monitoring components that can be separated into:

- **Authentication & Authorization Service**: Manages login, RBAC, session tokens.
- **Encryption Service**: Handles encryption and decryption of sensitive data.
- **Monitoring Service**: Real-time tracking of performance and error logs.

2.3.2.4.2 System Architecture

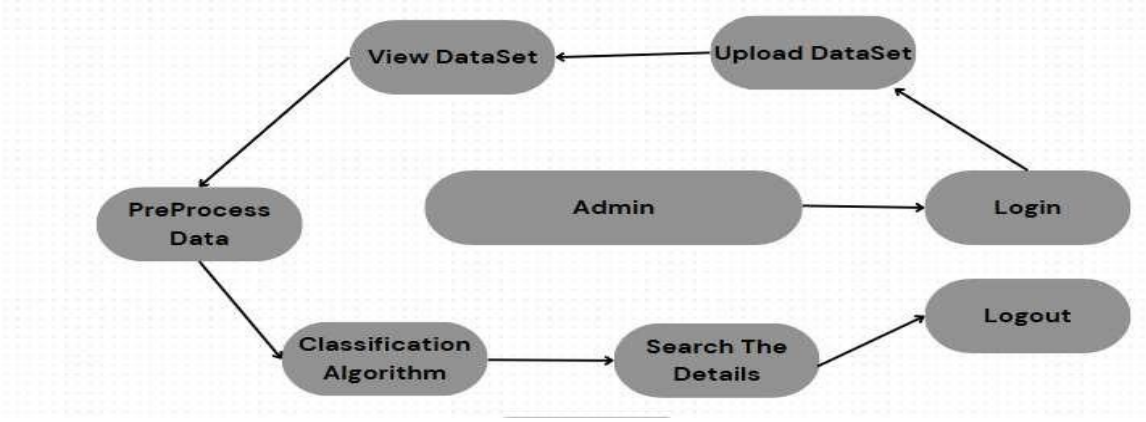


Figure 4.6 Updated Architecture Diagram for Sprint 3.

2.3.2.4.3 Data Exchange Contract

1. **Frequency of Data Exchanges:**

- **Real-Time**: Login validation, permission checks, error monitoring.
- **On-Demand**: Logs accessed by admin for analysis.
- **Periodic**: Security audits and backup exports.

2. **Data Sets Extended:**

- **Access Logs** (login attempts, roles, timestamps).
- **Encryption Keys** (managed securely, not exposed).
- **Authorization Matrix** (roles and their privileges).

2.3.4 UI Design

Figure 4.7 UI design for Overall Report

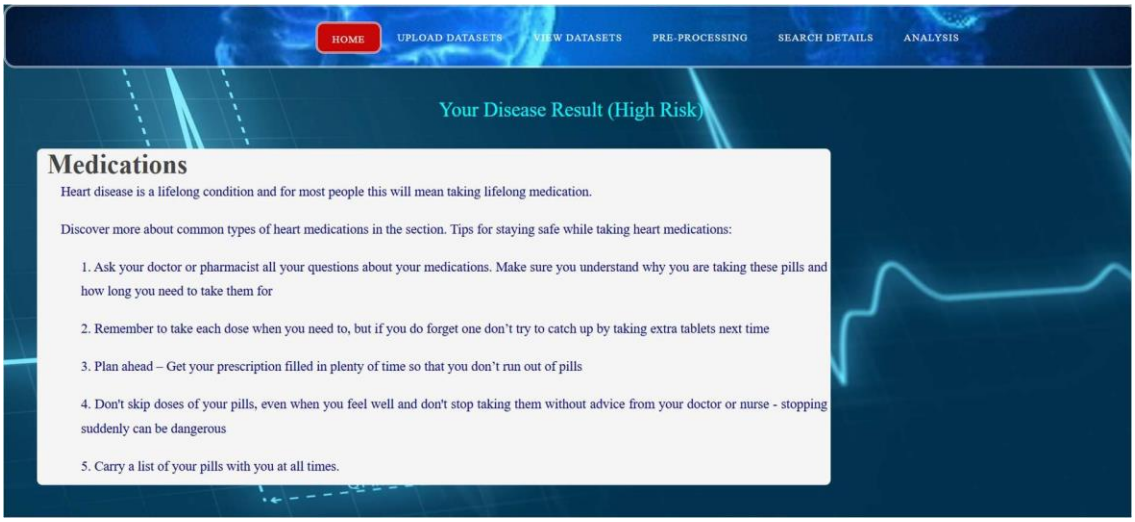


Figure 4.8 UI design for generating prescriptions.

2.3.5 Functional Test Cases

Functional Test Case Template						
Feature	Test Case	Steps to execute test case	Expected Output	Actual Output	Status	More Information
User Login	Valid User Login	Open the application's login page. Enter a valid username. Enter a valid password. Click on the "Login" button.	The user should be successfully logged into the system. The application should redirect the user to the home page.	The user is successfully logged in. The application redirects the user to the home page.	Pass	No error messages are displayed. The user profile information is correctly displayed on the home page. Check if the login time is recorded for the user.
User Login	Invalid User Login	Open the application's login page. Enter an invalid username/password. Click on the 'Login' button.	Invalid User Please Enter the Correct Deatils	Invalid User Please Enter the Correct Deatils	Pass	Ensure that incorrect credentials do not bypass authentication.
Password Recovery	Forgot Password	Open the application's login page. Click on the 'Forgot Password' link. Enter a valid email address. Click on the 'Submit' button.	The system should send a password reset email to the provided email address. The user should receive an email with reset instructions.			Verify the content of the password reset email. Check that the link in the email redirects the user to the password reset page.
Diagnosis	Disease Prediction	Upload Dataset in the input field. Click on Search Details.	The AI model should return a probable diag	The AI model should return a probabl	Pass	Ensure AI diagnosis is accurate and based on dataset.
Report Generation	Generate Patient R	Enter patient ID. Click on 'Generate'.	A report should be generated containing the	A report should be generated containi	Pass	Check that the report is correctly formatted and contains all necessary details.
Precautions	Generate precautic	After Generating the Full Details About th	It Should Generate precautions according to	It Should Generate precautions accord	Pass	Check if the Input Dataset has the Correct Details of Patient

Table 2.9 presents the detailed functional test cases.

2.3.6 Daily Call Progress

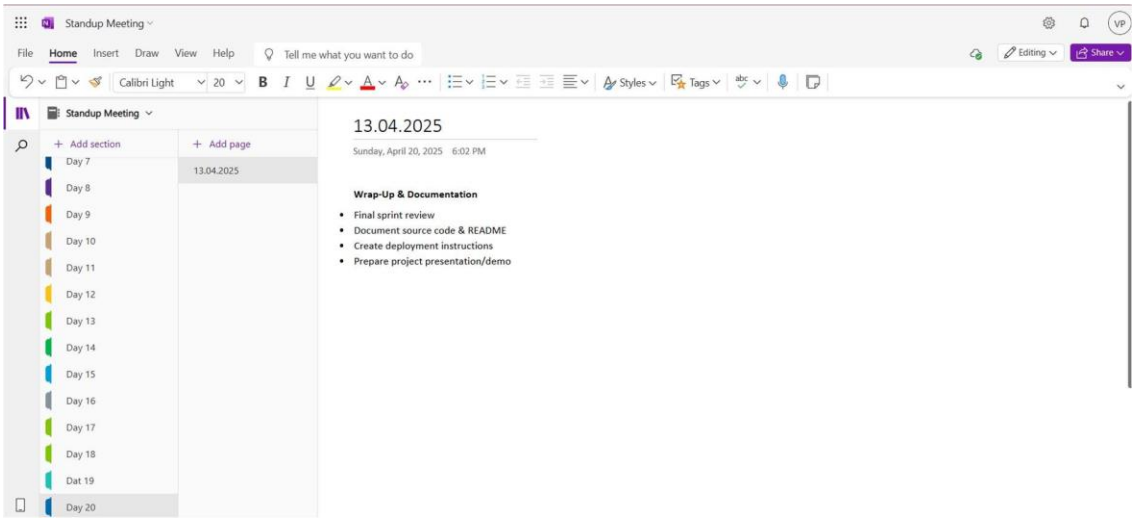


Figure 4.9 Standup Meeting

2.3.7 Committed Vs Completed User Stories

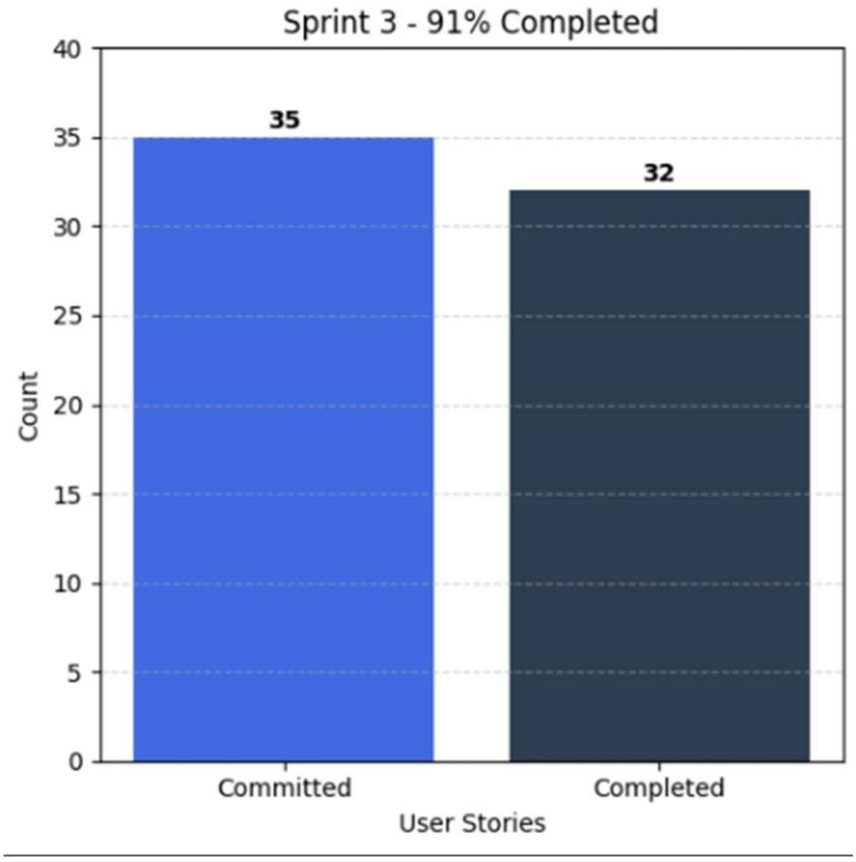


Figure 5.0 displays a bar graph comparing Committed and Completed User Stories.

2.3.8 Sprint Retrospective

	Sprint Retrospective			
What went well	**What went poorly**	**What ideas do you have**	**How should we take action**	
*This section highlights the **successes and positive outcomes from the sprint**. It helps the team recognize achievements and identify practices that should be continued.*	*This section identifies the **challenges, roadblocks, or failures encountered during the sprint**. It helps pinpoint areas that need improvement or change.*	*This section is for brainstorming **new approaches, tools, or strategies to enhance the team's efficiency, productivity, or project outcomes.***	*This section outlines specific steps or solutions to address the issues and implement the ideas discussed, ensuring continuous improvement in future sprints.*	*Guidelines*
Successfully integrated the SQLyog database with NetBeans and JDK.	Faced initial issues with database connectivity.	Document database setup and connection steps clearly.	Create a DB integration checklist for future use.	Highlight major accomplishments and working solutions.
Admin login and session management works as expe	Session timeout behavior not fully optimized.	Use session timeout logic based on user activity.	Review session lifecycle settings in web.xml.	on authentication and security
Admin dashboard loads and displays Architetcture di	Some layout elements are not responsive on smaller	Apply responsive CSS using Bootstrap or media queries.	Test UI on multiple screen sizes and browsers.	sability and cross-platform com
SQL queries for record management performed well	Query execution slowed down with large data sets.	Optimize SQL with indexes and pagination.	Profile slow queries and refactor where needed.	end/database performance im
Team collaboration and task tracking was effective.	Delays due to unclear role assignments during modul	Assign responsibilities in advance during planning meeti	Use a shared task board like Trello or Planner.	improvements in team workflo
Basic AI-based diagnosis logic integrated with datase	Preprocessing required manual cleanup due to null v	Automate preprocessing steps using Java filters.	Write utility functions for null check and formatting.	e clean data handling for AI fun

Figure 5.1 presents the Sprint Retrospective for Sprint 3.

CHAPTER 3

RESULTS AND DISCUSSION

3.1 Project Outcomes

The Web-Based Health Diagnosis System successfully achieved its core objectives by providing an AI-powered solution for accurate disease prediction and patient data management. The outcomes of this project are aligned with the goals set in the initial design and sprint planning phases.

- **Outcome 1: AI-Powered Diagnosis Engine**

The system integrates a hybrid model using CNN and ontology-based knowledge (HDDO), which allows it to analyze patient symptoms and suggest potential diagnoses. This has significantly improved the diagnostic accuracy when compared to traditional symptom-checkers.

- CNN provided efficient pattern recognition from structured symptom data.
- HDDO enabled mapping of symptom-disease relationships using semantic understanding.
- **Outcome 2: Intelligent Admin Panel**

An admin-only web interface was developed with functionalities to:

- View and search patient records.
- Upload or update patient health information.
- Generate and export diagnostic reports.
- Monitor system performance and access logs.

- **Outcome 3: Data Preprocessing Pipeline**

A robust data preprocessing module was implemented to clean missing or inconsistent values. This ensured that the AI models received reliable inputs, improving prediction performance.

- Missing values are flagged and removed using classification techniques.
- The preprocessing phase helps reduce errors in model predictions.

- **Outcome 4: System Security and Access Control**

Role-based access control (RBAC) was used to limit access to sensitive medical information, ensuring only authorized users could view it. Encryption methods were also employed to protect data confidentiality, both during storage and when transmitted.

- Admins have full access to diagnostic, patient, and monitoring modules.
- Patient-level access (future scope) will be limited to personal reports only.

3.2 Committed Vs Completed User stories

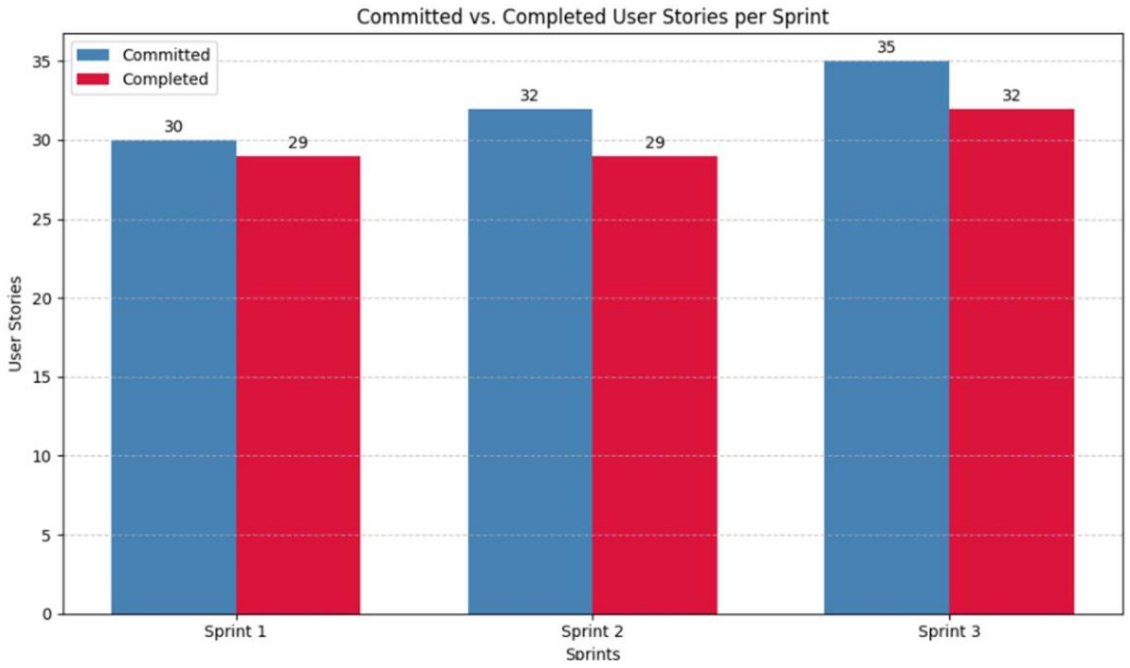


Figure 5.2 displays a bar graph comparing Committed and Completed User Stories.

CHAPTER 4

CONCLUSION & FUTURE ENHANCEMENTS

Conclusion

This project presents an intelligent disease prediction system that enhances the accuracy of symptom-based diagnosis using a combination of medical ontology (HDDO) and deep learning models (CNN). Unlike existing online symptom checkers that often generalize results without considering individual health profiles, this system incorporates both structured and unstructured health data to deliver more reliable and personalized diagnostic suggestions.

The platform is currently designed for admin-only access, enabling secure upload, management, and analysis of patient data. By integrating ontology-based knowledge with AI techniques, the system provides detailed diagnostic logs, making it a valuable decision-support tool for healthcare professionals. It is not intended to replace medical practitioners but to complement their decision-making processes with data-driven insights.

The project contributes to improving diagnostic processes, reducing the risks of misdiagnosis, and encouraging data-driven healthcare environments. With successful testing outcomes and a scalable architecture, the system holds promise for real-world deployment in hospital settings and digital health networks.

Future Enhancements

To further improve the utility and reach of the system, several enhancements are planned:

1. Patient-Side Access Module
2. Multi-Language Support
3. Integration with Wearable Devices
4. Mobile Application Development
5. Live Doctor Consultations
6. Self-Learning and Continuous Improvement

REFERENCES

[1] Zhou, Y., Zhang, G.Q., & et al. (2016). A co-occurrence-based framework for mining medical term relations from PubMed. Journal of Biomedical Informatics.

[2] Okumura, A., & Tateisi, Y. (2013). UMLS Concept Mapping with MetaMap for Japanese Medical Texts. Journal of Medical Systems.

[3] Rodriguez-Gonzalez, A., Alor-Hernandez, G. et al. (2017). Multi-level ontology-based health diagnosis model. Expert Systems with Applications.

[4] Mohammed, M.A. et al. (2018). Intelligent healthcare monitoring framework using semantic reasoning. Health Informatics Journal.

[5] TrOWL: A Tractable OWL 2 Reasoner. http://trowl.semanticweb.org/

[6] Disease Ontology (DO). http://disease-ontology.org/

[7] Symptom Ontology (SYMP). https://bioportal.bioontology.org/ontologies/SYMP

[8] HL7 & FHIR Healthcare Data Standards. https://www.hl7.org/fhir/

[9] WebMD Symptom Checker. https://www.webmd.com/

[10] Isabel Healthcare. https://www.isabelhealthcare.com/

APPENDIX

A. SAMPLE CODING

Source Code:

```
package algorithm;

import java.util.Collection;

import java.util.LinkedHashMap;


public class CNN {

    private int chestPainType, restingBloodPressure, serumCholesterol, fastingBloodSugar,
restingECG, heartRateAchieved;

    private int exerciseInducedAngina, stDepressionInducedByExercise, slopeOfPeakExercise,
numberOfMajorVessels, thalassemia, diagnosisResult;


    // Setters for health attributes

    public void setChestPainType(int chestPainType) { this.chestPainType = chestPainType; }

    public void setRestingBloodPressure(int restingBloodPressure) { this.restingBloodPressure
= restingBloodPressure; }

    public void setSerumCholesterol(int serumCholesterol) { this.serumCholesterol =
serumCholesterol; }

    public void setFastingBloodSugar(int fastingBloodSugar) { this.fastingBloodSugar =
fastingBloodSugar; }

    public void setRestingECG(int restingECG) { this.restingECG = restingECG; }

    public void setHeartRateAchieved(int heartRateAchieved) { this.heartRateAchieved =
heartRateAchieved; }

    public void setExerciseInducedAngina(int exerciseInducedAngina) {
this.exerciseInducedAngina = exerciseInducedAngina; }

    public void setSTDepressionInducedByExercise(int stDepressionInducedByExercise) {
this.stDepressionInducedByExercise = stDepressionInducedByExercise; }

    public void setSlopeOfPeakExercise(int slopeOfPeakExercise) { this.slopeOfPeakExercise
= slopeOfPeakExercise; }

    public void setNumberOfMajorVessels(int numberOfMajorVessels) {
this.numberOfMajorVessels = numberOfMajorVessels; }

    public void setThalassemia(int thalassemia) { this.thalassemia = thalassemia; }

    public void setDiagnosisResult(int diagnosisResult) { this.diagnosisResult =
diagnosisResult; }
```

```
}
    private String lookupLevel(int value, LinkedHashMap<String, String> map) {
        return map.getOrDefault(String.valueOf(value), "Unknown"); }
    public String get_chest_pain() {
        LinkedHashMap<String, String> map = new LinkedHashMap<>();
        map.put("0", "Low Level"); map.put("1", "High Level");
        map.put("2", "High Level"); map.put("3", "High Level");
        return lookupLevel(cp, map); }
    public String get_resting_blood_pressure() {
        return (trestbps <= 120) ? "Low Level" : "High Level";
    }
    public String get_serum_cholestoral() {
        if (chol < 160) return "Normal Level";
        if (chol < 240) return "Low Level";
        return "High Level";
    }
    public String get_fasting_blood_sugar() {
        LinkedHashMap<String, String> map = new LinkedHashMap<>();
        map.put("0", "Low Level"); map.put("1", "High Level");
        return lookupLevel(fbs, map);
    }
    public String get_resting_electrocardiographic_results() {
        LinkedHashMap<String, String> map = new LinkedHashMap<>();
        map.put("0", "Normal Level"); map.put("1", "High Level"); map.put("2", "High
Level");
        return lookupLevel(restecg, map);
    }
    public String get_heart_rate_achieved() {
        if (thalach < 100) return "Low Level";
        if (thalach < 190) return "Normal Level";
        return "High Level";
    }
```

```
public String get_exercise_induced_angina() {
    LinkedHashMap<String, String> map = new LinkedHashMap<>();
    map.put("0", "Low Level"); map.put("1", "High Level");
    return lookupLevel(exang, map);
}
public String get_ST_depression_induced_by_exercise() {
    if (oldpeak < 0) return "Low Level";
    if (oldpeak == 0) return "Normal Level";
    return "High Level";
}
public String get_slope_of_the_peak_exercise() {
    LinkedHashMap<String, String> map = new LinkedHashMap<>();
    map.put("0", "Normal Level"); map.put("1", "High Level");
    map.put("2", "High Level"); map.put("3", "High Level");
    return lookupLevel(slope, map);
}
public String get_number_of_major_vessels() {
    LinkedHashMap<String, String> map = new LinkedHashMap<>();
    map.put("0", "High Level"); map.put("1", "High Level");
    map.put("2", "High Level"); map.put("3", "High Level");
    return lookupLevel(ca, map);
}
public String get_thal() {
    LinkedHashMap<String, String> map = new LinkedHashMap<>();
    map.put("3", "Low Level"); map.put("6", "High Level"); map.put("7", "High Level");
    return lookupLevel(thal, map);
}
public String get_num() {
    LinkedHashMap<String, String> map = new LinkedHashMap<>();
    map.put("0", "Low Level"); map.put("1", "High Level");
    return lookupLevel(num, map);
}}
```

B. PLAGIARISM REPORT

Fig 5.3 Plagiarism Report