



CureWise - A Medicine Recommendation System

A Project Report

Submitted in partial fulfilment of the
Requirements for the award of the Degree of



BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY)

By

Harsh Jain – 53003220051

Under the esteemed guidance of

Mr. Prashant Chaudhary

Asst. Professor & Coordinator B.Sc. (IT)



DEPARTMENT OF INFORMATION TECHNOLOGY

Shri Vile Parle Kelavani Mandal's

**USHA PRAVIN GANDHI COLLEGE OF ARTS, SCIENCE, AND
COMMERCE**

NAAC Accredited 'A+' Grade

(Affiliated to University of Mumbai)

**MUMBAI, 400056
MAHARASHTRA**

PROFORMA FOR THE APPROVAL PROJECT PROPOSAL

PRN No.: 2022016400554764

Roll no: C048

1.Name of the Student

Harsh Jain

2.Title of the Project

CureWise – A Medicine Recommendation System

3. Name of the Guide:

Mr. Prashant Chaudhary

4. Teaching experience of the Guide: Years

5.Is this your first submission? Yes ☐ No ☐

Signature of the Student

Date:

Signature of the Guide

Date:

Signature of the Coordinator

Date:

ABSTRACT

CureWise is a groundbreaking medicine recommendation system that harnesses advanced artificial intelligence and machine learning technologies to deliver personalized healthcare insights.

It analyzes user-submitted symptoms along with individual health histories to generate preliminary diagnoses and provide tailored suggestions for medications, dietary plans, and exercise routines.

CureWise incorporates continuous learning and adaptive algorithms that integrate real-time user feedback and the latest medical research, ensuring its healthcare recommendations remain evidence-based and dynamically up-to-date.

The system integrates state-of-the-art Optical Character Recognition (OCR) technology, which converts doctors' handwritten prescriptions into clear digital text, enhancing accessibility and understanding of treatment details.

An intuitive user interface enables real-time tracking of symptoms and health metrics, effectively bridging the gap between immediate healthcare needs and professional medical consultation.

By consolidating diverse data sources and continuously refining its predictive models, CureWise promotes proactive health management and informed decision-making.

Ultimately, the platform transforms everyday health monitoring into an interactive, data-driven experience while emphasizing the importance of consulting qualified healthcare professionals for comprehensive care.

ACKNOWLEDGEMENT

The success and final outcome of this project required a lot of guidance and assistance from many people, and I am extremely privileged to have got this all through the completion of my project. All that I have done is only due to such supervision and assistance and I would not forget to thank them.

I owe my deep gratitude to our Project Guide and B. Sc.IT Coordinator, **Mr. Prashant Chaudhary**, who took keen interest in my project work and guided me all along, till the completion of my project work by providing all the necessary information for developing a good system.

I am thankful to our Principal **Dr. Anju Kapoor** for providing us with the facilities for the smooth working of our project.

I am thankful and fortunate to get constant encouragement, support, and guidance from all the teaching staff of the **BSc. I.T. Department of Usha Pravin Gandhi College**, which helped me successfully completing my project work.

Lastly, I would like to express my appreciation towards my fellow classmates and friends for providing me the moral support and encouragement.

-Harsh Jain

DECLARATION

I hereby declare that the project entitled, “**CureWsie - A Medicine Recommendation System**” done at **SVKM’s Usha Pravin Gandhi College of Arts, Science and Commerce**, has not been in any case duplicated to submit to any other university for the award of any degree. To the best of my knowledge other than me, no one has submitted to any other university.

The project is done in partial fulfilment of the requirements for the award of degree of **BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY)** to be submitted as final semester project as part of our curriculum.

- **Harsh Jain**

TABLE OF CONTENTS

SR.NO		CONTENT		PAGE NO
1	Introduction			
	1.1	Background		13
	1.2	Objectives		13
	1.3	Purpose, Scope, and Applicability		14
		1.3.1	Purpose	14
		1.3.2	Scope	15
		1.3.3	Applicability	15
	1.4	Achievements		17
1.5	Organization of Report		17	
2	Survey of Technologies			
	2.1	Frontend		19
		2.1.1	Comparison With Framework	19
		2.1.2	HTML	19
		2.1.3	CSS	20
		2.1.4	JAVASCRIPT	20
		2.1.5	BOOTSRTAP	20
		2.1.6	Features Of Frontend Stack	21
		2.1.7	Advantages	21
		2.1.8	Disadvantages	21

		2.1.9	Summarization	22
	2.2		Backend	23
3	Requirement Analysis			
	3.1		Problem Definition	27
	3.2		Requirement Specification	27
		3.2.1	Functional Requirements	27
		3.2.2	Non-Functional Requirements	28
	3.3		Planning and Scheduling	28
		3.3.1	Project Planning Process	28
		3.3.2	Scheduling Process	29
		3.3.3	Techniques used for Scheduling (Gantt chart)	31
	3.4		Software Requirements	33
		3.4.1	Developer Side	33
		3.4.2	Client-Side	34
	3.5		Hardware Requirements	35
		3.5.1	Developer Side	35
		3.5.2	Client-Side	35
	3.6		Preliminary Product Description	37
	3.7		Conceptual Models	37
		3.7.1	SDLC	37
		3.7.2	Agile Model	37
		3.7.3	Why Agile Model ?	38

4	System Design		
	4.1	Basic Modules	40
	4.2	Database Design	42
		4.2.1 User Table Design	42
		4.2.2 Data integrity and constraints	42
	4.3	Procedural design (Logic diagrams)	45
		4.3.1 Use Case	45
		4.3.2 Activity	46
		4.3.3 Sequence	47
		4.3.4 Entity Relationship	48
	4.4	User Interface Design	49
	4.5	Security Issues	51
	4.6	Test Case Design	53
5	Implementation and Testing		
	5.1	Implementation Approaches	54
	5.2	Testing Approach	65
		5.2.1 Testing Methodologies	65
		5.2.2 Test Cases	66
6	Results and Discussion		
	6.1	Overview	71

7	6.2	Landing and About us pages		72
		6.2.1	Landing	72
		6.2.2	About	73
	6.3	Login and Signup pages		74
		6.3.1	Login	74
		6.3.2	Signup	75
	6.4	Dashboard		76
	6.5	Disease Prediction Page		77
	6.6	Medicine Recommendation Page		78
	6.7	Nutrition Plan Page		79
	6.8	Workout Plan Page		80
	6.9	Disease Description Page		81
	6.10.	Additional Interfaces		82
		6.10.1	OCR	82
		6.10.2	ChatBot	83
		6.10.3	Medical Text Analyzer	84
7	Conclusion			
	7.1	Conclusion		85
	7.2	Limitations of the system		86
	7.3	Future Scope of the project		87
References				89

LIST OF TABLES			
SR.NO		TABLE NAME	PAGE NO
2.	2.1.1	Comparisons with frameworks	19
	2.1.9	Summarization	22
	2.2.2.4	MySQL V/S MongoDB	26
4.	4.2	Database Design	42-43
	4.6	Test Case Design	53
5.	5.2.1	Testing Methodologies	65

LIST OF FIGURES

SR.NO		FIGURE NAME	PAGE NO
3.	3.1	Gantt Chart	32
	3.2	Agile Model	36
4.	4.1	Use case diagram	45
	4.2	Activity Diagram	46
	4.3	Sequence Diagram	47
	4.4	Entity relationship diagram	48
	4.5	User Interface: CureWise-Landing page	49
	4.6	User Interface: CureWise-Login/signup page	50
6.	6.1	Landing/About us Page	72
	6.2	Login/Signup Page	74
	6.3	Dashboard	76
	6.4	Disease Prediction Page	77
	6.5	Medicine Recommendation Page	78
	6.6	Nutrition plan	79
	6.7	Workout Plan	80
	6.8	Disease Description Page	81
	6.10	Additional Interfaces	82-84

Chapter 1

Introduction

1.1 Background:

In today's fast-paced world, we often seek quick solutions to save time and effort. Just like how streaming services recommend movies based on our tastes, our website uses advanced technology to provide personalized healthcare advice. Our website uses machine learning to predict diseases based on symptoms that users enter. This means users can get an idea of what might be wrong without having to wait for a doctor's appointment. But it doesn't stop at diagnosis. The website also suggests the right medicines, diet plans, exercise routines, and precautions tailored to each user's specific needs. This comprehensive approach ensures users receive all-around care. Additionally, we are combining Optical Character Recognition (OCR) technology to our system to help users understand doctors' handwriting from uploaded medical documents. This feature improves user convenience by making medical information more accessible. Moving ahead, our project is designed to not only enhance personal health management but also to empower users with the tools and knowledge they need to take control of their well-being in an increasingly digital world.

1.2 Objectives:

The objective of our medicine recommendation system is -

1. To provide personalized and accurate healthcare suggestions to users based on their symptoms.
2. To improve user satisfaction, health outcomes, and overall well-being by delivering tailored recommendations for medicines, diets, workouts, and precautions.

- Key objectives for our medicine recommendation system include:

- a) **Accuracy:** Deliver highly accurate and relevant health suggestions to ensure users receive

effective and appropriate recommendations for their conditions.

- b) **User Experience:** Enhance the overall user experience by presenting health recommendations in an intuitive and user-friendly interface, making it easy for users to input symptoms and receive advice.
- c) **Accessibility:** Integrate features like Optical Character Recognition (OCR) to help users understand doctors' handwriting, making medical information more accessible and understandable.
- d) **Comprehensiveness:** Provide a holistic approach to health management by offering recommendations not just for medicines but also for diet, exercise, and precautions.
- e) **Advisory:** Include a clear note or warning message advising users to always consult their doctor, emphasizing that the system's recommendations are supplementary and not a substitute for professional medical advice.

By achieving these objectives, our medicine recommendation system can significantly enhance user satisfaction, improve health outcomes, and contribute to a positive user experience, thereby benefiting both users and the healthcare platforms that implement the recommendation system.

1.2.1 Purpose:

The purpose of a medicine recommendation system is to provide accurate and personalized health suggestions based on symptoms inputted by users. This system is designed to enhance user experience, support informed health decisions, and integrate OCR technology for better readability of doctors' handwritten prescriptions. Key purposes include:

1. **Enhance User Experience:** By offering personalized health recommendations based on user-inputted symptoms, users receive relevant advice on medicines, diets, precautions, and workouts tailored to their specific health conditions, leading to a more satisfying and effective health management experience.
2. **Support Informed Health Decisions:** The system aids users in understanding their health conditions better by converting doctors' handwritten prescriptions into readable text and providing additional information. This enhances comprehension and encourages users to make informed decisions about their health.
3. **Educational and Precautionary Guidance:** Along with medical recommendations, the system provides important notes or warning messages to consult with healthcare professionals, ensuring that users do not solely rely on automated advice and seek professional guidance when necessary.

1.2.2 Scope:

Our medicine recommendation system has vast potential and numerous applications in the healthcare industry. Here are some aspects of its scope:

1. **Research and Data Analysis:** By using machine learning algorithms to predict diseases based on symptoms entered by users, the system provides preliminary diagnoses that can guide users on whether to seek professional medical help.
2. **Comprehensive Care Plans:** The system suggests not only the right medicines but also diet plans, exercise routines, and precautions tailored to each user's specific needs.
3. **OCR Integration:** By incorporating OCR technology, the system can help users understand doctors' handwriting from uploaded medical documents, making medical information more accessible.
4. **Consultation Warnings:** The system provides important notes or warning messages advising users to consult healthcare professionals, emphasizing that the recommendations are supplementary and not a replacement for professional medical advice.

1.2.3 Applicability:

Our medicine recommendation system has broad applicability and can be used in various contexts to enhance patient care and user convenience. Key areas of applicability include:

1. **Home Healthcare:** Users can access healthcare advice from the comfort of their homes, reducing the need for frequent doctor visits for minor health issues. The system provides recommendations for medications, diet, and exercise routines, helping users manage their health proactively. This approach not only saves time but also empowers users to take charge of their health in a familiar environment.
2. **Telemedicine:** The system can support telemedicine platforms by providing remote users with healthcare

recommendations based on their symptoms and health data. By integrating with telemedicine services, the system ensures that patients receive timely and accurate medical advice, even from a distance. This enhances the effectiveness of virtual consultations and helps bridge the gap between patients and healthcare providers.

3. **Pharmacies:** Pharmacies can use the system to offer medication advice to customers and improve medication adherence. The system can suggest optimal medication regimens, enhancing customer satisfaction and improving health outcomes.

4. **Health Monitoring Applications:** The system can be integrated into health monitoring applications, allowing users to receive insights into their health status and make informed decisions regarding their well-being and lifestyle.

1.3 Achievement:

Our system significantly reduces the time users spend seeking medical advice by providing quick and accurate healthcare recommendations based on their symptoms and health data. By delivering personalized medication suggestions, diet plans, exercise routines, and precautions, users can make informed decisions about their health promptly, bypassing the need for immediate doctor's appointments for minor issues. The integration of OCR technology further enhances user convenience by deciphering doctors' handwriting on uploaded medical documents, making critical medical information easily accessible. Moving forward, we aim to collaborate with healthcare providers and pharmaceutical companies to continually refine and expand our system, striving to make healthcare more accessible, efficient, and tailored to individual needs, thereby enhancing the overall quality of life for users.

1.4 Organization of report:

- **1st chapter:**

It gives a brief introduction about the project, its objectives and goals and achievements. It also focuses on the scope and purpose of the project and how it's applicable in real-world problems.

- **2nd chapter:**

This chapter focuses on survey of technologies used in the project, this shows the comparison between various technologies, and it shows which programming language would be better to choose.

- **3rd chapter:**

This chapter defines the problem definition. It will also show the system requirements of the components such as hardware and software requirements. The Gantt Chart in this chapter will show how I planned to complete this project. The conceptual model will consist of a basic flow chart that shows how the website will basically work.

- **4th chapter:**

This chapter will show some logic diagrams like use case, activity, sequence, and ER diagram. System describing the modules or the components we will be using in our application.

- **5th chapter:**

This chapter defines implementation and testing of the project. Some important pieces of code and testing approach are added.

- **6th chapter:**

This chapter will include Test result and User documentation. Step-by-step use of platform is instructed along with screenshots.

- **7th chapter:**

This chapter will show the conclusion, significance, limitation, and future scope of the project.

- **8th chapter:**

This chapter will include all the references needed to build this project.

Chapter 2

Survey of Technologies

2.1 Frontend.

2.1.1 Comparisons with Frameworks

Aspect	Flask	Django	React
Ease of Use	Simple and minimalistic	More complex due to full-stack nature	Requires understanding of JS and related technologies
Scalability	Suitable for small to medium projects	Suitable for small to enterprise-level projects	Suitable for small to enterprise-level projects
Performance	Lightweight but not for heavy loads	Superior performance	Superior performance but dependent on code efficiency
UI Complexity	Best for simple UIs	Suitable for complex UIs	Suitable for complex UIs
Deployment	Easy deployment and hosting	Requires web server setup	Requires build process and server setup

2.1.2 HTML

2.1.2.1 What is html?

HTML (Hypertext Markup Language) is the backbone of your frontend, responsible for structuring the content and layout of your web pages. It provides the foundation upon which the entire frontend is built.

2.1.3 Css

2.1.3.1 What is Css?

CSS (Cascading Style Sheets) is used for styling your web pages, ensuring they look visually appealing. CSS allows for a separation between content and presentation, enabling you to customize fonts, colors, layouts, and other visual elements.

2.1.4 JavaScript

2.1.4.1 What is JavaScript?

JavaScript adds interactivity to your web pages. It handles events, manipulates the DOM (Document Object Model), and allows for communication with your Flask backend through APIs. JavaScript ensures users can interact with the system dynamically, such as submitting symptoms and receiving recommendations in real-time.

2.1.5 Bootstrap

2.1.5.1 What is Bootstrap?

Bootstrap is a popular frontend framework used for responsive design. It provides pre-built components like navigation bars, forms, buttons, and grids, which speed up development and ensure the application is mobile-friendly and accessible on various devices.

2.1.6 Features of the Frontend Stack:

- Responsiveness: Bootstrap ensures that your web pages look great on all devices, whether it's a desktop, tablet, or smartphone.
- Interactivity: JavaScript enables dynamic interactions, such as form submissions and live data updates.
- Styling: CSS allows for sophisticated and consistent styling across all web pages.
- Structure: HTML provides the structural framework that organizes your content.

2.1.7 Advantages:

- Ease of Use: The combination of HTML, CSS, and Bootstrap makes it easy to develop and design responsive web pages quickly.
- Interactivity: JavaScript enhances the user experience by enabling dynamic content and real-time updates.
- Cross-Compatibility: Bootstrap ensures that the application is accessible on various devices with different screen sizes.

2.1.8 Disadvantages:

- Tooling Complexity: React often requires a complex setup with additional tools and libraries (e.g., Webpack, Babel) for optimal performance, which can complicate the development process.
- Single-Threaded Limitations: Node.js operates on a single-threaded event loop, which can lead to performance bottlenecks in CPU-intensive tasks.
- Callback Hell: The heavy reliance on callbacks for asynchronous operations can lead to "callback hell," making the code difficult to read and maintain.

2.1.9 Summarization:

Technology	Purpose	Features	Advantages	Disadvantages
HTML	Structuring the content and layout of web pages	- Provides the structure for web pages using elements like headings, paragraphs, and forms	- Easy to learn and use - Essential for web page structure	- Limited in styling and interactivity without CSS and JavaScript
CSS	Styling the web pages	- Separates content from presentation - Customizes fonts, colors, layouts	- Enhances the visual appeal - Supports responsive design with media queries	- Can become complex to manage in large projects - Cross-browser compatibility issues
JavaScript	Adding interactivity and dynamic content	- Handles events and DOM manipulation - Communicates with the backend via APIs	- Enables real-time updates and dynamic user interactions	- Can lead to complex and hard-to-manage code without proper structure
Bootstrap	Providing responsive design and pre-built components	- Includes components like navigation bars, forms, and grids - Ensures responsiveness	- Speeds up development - Ensures consistent design across devices	- Limits customization - May result in similar-looking websites

2.2 Backend: Primary technologies used in backend is MYSQL

2.2.1 MYSQL:

2.2.1.1 What is SQL Database?

SQL, which stands for Structured Query Language, is a specialized programming language used for managing and manipulating relational databases. A SQL database, also known as a relational database management system (RDBMS), is a software system designed to store, manage, and retrieve structured data. These databases are organized into tables, each consisting of rows and columns, and they follow a predefined schema that defines the structure and relationships of the data.

2.2.1.2 Features of MySQL:

- **Data Storage:** SQL databases are designed to store structured data in tables. They can manage a variety of data types such as numbers, text, dates, and more.
- **Data Retrieval:** SQL allows users to retrieve data from the database using queries. The `SELECT` statement is commonly used to retrieve specific data from one or more tables.
- **Data Manipulation:** SQL databases support data manipulation operations such as `INSERT` (for adding new records), `UPDATE` (for modifying existing records), and `DELETE` (for removing records).
- **Data Integrity:** They enforce data integrity constraints such as primary keys, foreign keys, unique constraints, and check constraints to maintain the consistency and accuracy of the data.
- **Transaction Management:** SQL databases support transactions, which allow a group of SQL statements to be executed as a single unit. Transactions ensure the database remains in a consistent state, and they follow the ACID (Atomicity, Consistency, Isolation, Durability) properties.
- **Concurrency Control:** SQL databases manage multiple concurrent users and ensure that data remains consistent even when multiple users are accessing and modifying it simultaneously.
- **Indexing:** They provide indexing mechanisms to optimize query performance by speeding up data retrieval operations.

- **Security:** SQL databases offer various security features, including user authentication, authorization, and data encryption, to protect data from unauthorized access.
- **Backup and Recovery:** SQL databases support regular backups and provide mechanisms for recovering data in case of hardware failures or other disasters.
- **Scalability:** Many SQL databases support horizontal and vertical scalability options to manage growing data and user loads. This can involve techniques like sharding and replication.

2.2.2 Flask:

2.2.2.1 What is Flask?

Flask is a micro web framework for Python that is used to build web applications. It is called a "micro" framework because it is designed to be lightweight and easy to use, with a minimal set of components for building web applications. Flask is not a full-stack framework like Django; instead, it provides the essential tools and libraries for building web applications and leaves many architectural decisions to the developer.

2.2.2.2 Features of Flask:

- **Routing:** Flask allows you to define URL routes and associate them with Python functions, making it easy to manage different HTTP requests (e.g., GET, POST) and render different views or perform actions based on the URL.
- **Templating:** Flask includes a template engine (Jinja2 by default) that helps you generate dynamic HTML content by inserting data into HTML templates. This makes it easier to create web pages that display data from your application.
- **HTTP Request Handling:** Flask provides simple and convenient ways to manage incoming HTTP requests, access request parameters, and work with request headers.
- **Web Development Extensions:** Flask has a wide range of extensions and libraries available for adding features like user authentication, database integration (e.g., SQLAlchemy), form handling, and more

to your web applications.

- **Development Server:** Flask includes a built-in development server for testing and debugging your applications locally.
- **RESTful Support:** Flask is often used for building RESTful APIs because of its simplicity and flexibility.

2.2.2.3 Why using Flask and no other Framework?

- **Lightweight and Minimalistic:** Flask is a micro-framework, which means it provides the essentials for building web applications without imposing a lot of structure. This allows developers to have more control and choose the components they want to use, making it suitable for small to medium-sized projects.
- **Flexibility:** Flask doesn't enforce a specific way of doing things. It gives you the freedom to structure your application as you see fit. This flexibility can be beneficial when you have unique project requirements.
- **Large Ecosystem:** Despite being lightweight, Flask has a thriving ecosystem of extensions and libraries that can be easily integrated into your project when needed. You can add features like authentication, database integration, and more using Flask extensions.
- **Community and Documentation:** Flask has a strong community of developers, which means there are plenty of resources, tutorials, and documentation available to help you get started and solve problems.
- **Rapid Prototyping:** Flask is excellent for rapid prototyping and building small to medium-sized web applications quickly. It's a popular choice for hackathons and proof-of-concept projects.
- **Pythonic:** If you're already familiar with Python, using Flask allows you to leverage your Python skills in web development. The code you write in Flask typically follows Python's idiomatic style.

● MySQL V/S MongoDB

Aspect	MySQL	MongoDB
Data model	Relational (Table Based)	NoSQL (Document Based)
Schema	Fixed Schema	Dynamic Schema
Transactions	ACID Compliant	Support for multi-document ACID Transactions
Complex Queries	Supports complex joins	Limited support for Complex queries
Flexibility	Limited Schema Flexibility	Schema Less with flexible Data Structure

Chapter 3 Requirement and Analysis

3.1 Problem Definition

The primary challenge is that users often lack access to personalized medical recommendations based on their symptoms without needing an immediate doctor's visit. The existing solutions are limited in scope, often focusing solely on medicine recommendations without taking into account other aspects like diet, exercise, and precautionary measures.

Other identified problems include:

- **Lack of Accessibility:** Users in remote areas with limited internet bandwidth find it difficult to access high-quality healthcare platforms.
- **Limited Integration with Medical Data:** Current systems do not efficiently process medical prescriptions or doctor's notes for further insight.
- **Delayed or Insufficient Recommendations:** Many medical platforms do not offer real-time recommendations or are unable to provide a comprehensive health plan covering medicines, exercise, and diet.

3.2 Requirement Specification

3.2.1 Functional Requirements

- Users must be able to create an account using a valid email address and password.
- Secure login should be provided to users, with an option to reset the password if forgotten.
- The system shall recommend personalized health plans (medicines, diet, exercise) based on the user's symptoms.
- Integration with Optical Character Recognition (OCR) technology for reading and interpreting uploaded medical prescriptions.
- Users can search for health-related content, such as medicines, diets, symptoms, or exercises.
- Search results must include detailed information such as side effects, dosages, and reviews.
- The system should provide real-time health suggestions and recommendations, with response

times under 2 seconds.

3.2.2 Non- Functional Requirements

- The system architecture should support scalability to accommodate growing data and users.
- The user interface must be intuitive and accessible across various devices (PCs, smartphones, tablets).
- The platform must integrate a reliable medical database for accurate, up-to-date health and drug information.
- It should operate efficiently even on low-bandwidth internet connections.

3.3 Planning and Scheduling:

Planning and scheduling are critical steps in ensuring that the medical recommendation system is developed on time, within budget, and meets the project's objectives. This section outlines the detailed process for planning and scheduling the development phases, resources, and timelines.

3.3.1 Project Planning Process:

The project planning process involves defining the project objectives, identifying the necessary tasks, estimating resource requirements, and creating a roadmap for the development lifecycle. Below are the steps for your project planning:

a. Define Objectives:

The key objectives of the medical recommendation system are to provide personalized health advice based on user symptoms, integrate OCR for prescription recognition, and offer real-time health suggestions across multiple platforms.

b. Breakdown of Tasks: Each task in the project is broken down into manageable phases, such as:

- i. Requirements Gathering and Analysis
- ii. System Design and Architecture
- iii. Frontend Development
- iv. Backend Development

- v. Integration with OCR
- vi. Database Integration (MySQL)
- vii. Machine Learning Model Development
- viii. Testing and Quality Assurance
- ix. Deployment and User Feedback
- x. Maintenance and Updates

c. Resource Allocation:

- i. Team Members: Developers, designers, testers, and project managers.
- ii. Software Tools: PyCharm, Jupyter Notebooks, MySQL, Chrome Browser.
- iii. Hardware: Server for backend hosting, devices for testing the user interface across platforms (PCs, smartphones, tablets).

d. Risk Identification:

Risks in this project may include potential delays in the integration of OCR, challenges with machine learning model accuracy, or difficulties in achieving real-time recommendations under 2 seconds. These risks will be identified early and mitigated with proper planning.

e. Timeline Creation:

Using scheduling tools like Gantt charts and CPM (Critical Path Method), timelines are estimated for each task and milestone to ensure project completion within the set deadline.

3.3.2 Scheduling Process:

The goal of scheduling is to estimate the time it will take to complete a project. It informs us. To keep costs and resources under control. It acts as a record. It oversees dealing with changes and uncertainty.

Some commonly tools used for scheduling are GANTT Chart, Schedule Network Analysis, Critical Path Method (CPM), PERT (Program Evaluation and Review Technique).

1. Task Sequencing:

Tasks are arranged in logical order, ensuring that dependent tasks are completed before the next phase begins.

For example:

- i. Before integrating the machine learning model, the data collection and preparation phase must be complete.
- ii. Before frontend development, the system design and architecture must be finalized.

2. Task Estimation:

Each task is assigned an estimated time of completion based on complexity and resource availability.

For instance:

- i. Frontend and backend development may take around 2-3 months.
- ii. Machine learning model development, testing, and integration may take an additional 2 months.
- iii. System testing and quality assurance will take about 1 month.

3. Task Dependencies:

Dependencies are identified to ensure that certain tasks do not block others. For example, system testing can only begin after the entire backend and frontend functionalities are implemented.

4. Review and Adjustment:

The schedule is regularly reviewed, and adjustments are made based on any delays or unforeseen challenges. Agile techniques will be used to continuously adapt the schedule to meet dynamic requirements.

3.3.3 Techniques used for scheduling: Gantt chart (Bar Chart)

Scheduling ensures that tasks are completed on time by organizing the project into clearly defined phases with set timelines. It involves estimating the time required for each task, establishing dependencies between activities, and assigning deadlines. This helps the team stay on track and monitor progress efficiently, ensuring smooth transitions between different stages like frontend development, backend integration, and testing.

The scheduling process also allows for resource optimization, as tasks are mapped to specific team members or tools, ensuring no overlap or bottlenecks. By having a well-structured schedule, the team can easily identify any delays, manage risks, and adjust timelines if necessary, maintaining control over the overall project flow.

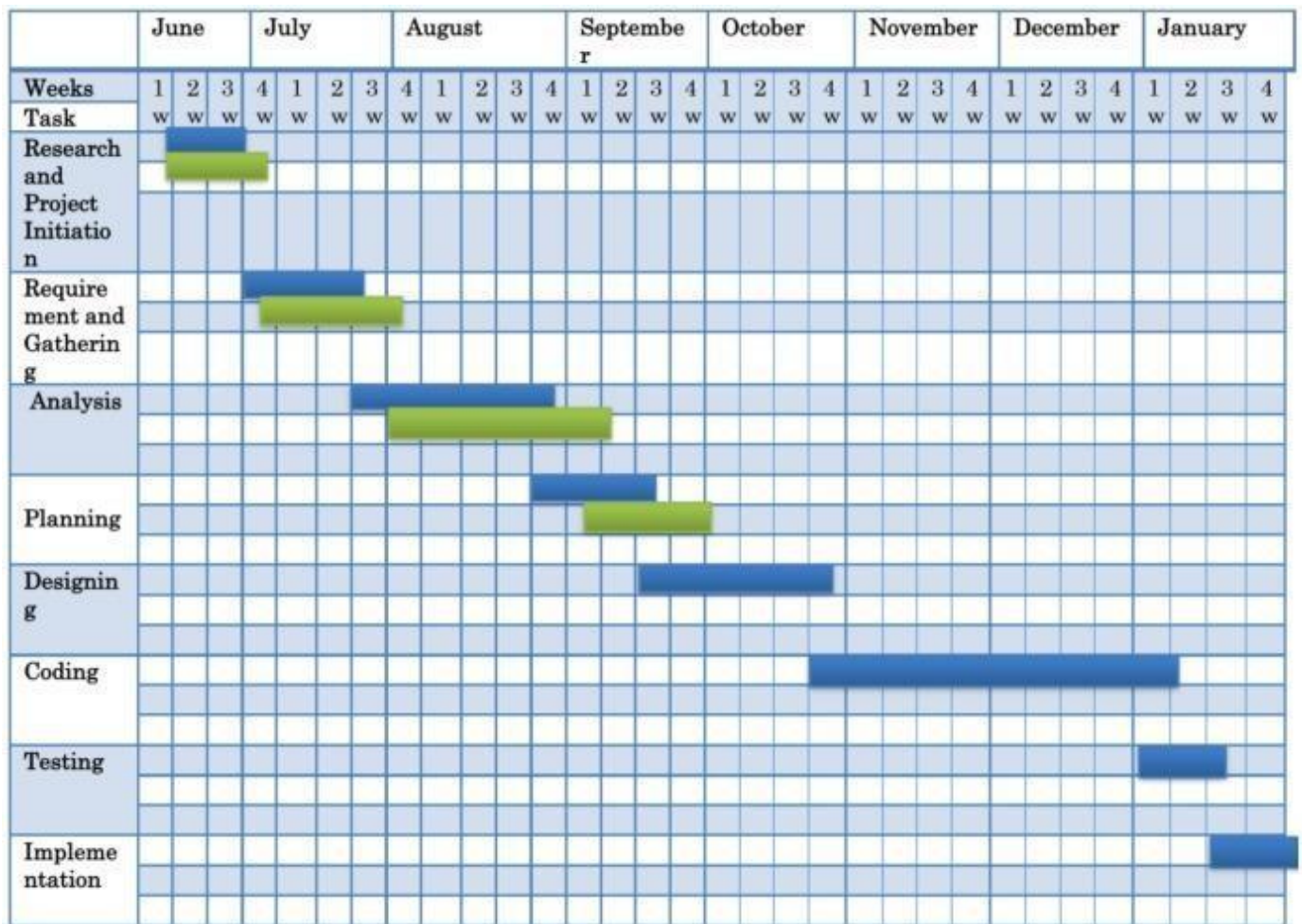
Gantt Chart:

A Gantt chart is a visual representation that helps in scheduling tasks, tracking dependencies, and monitoring deadlines. Each task is represented as a bar, showing the start date, duration, and end date.

Advantages:

Gantt charts make it easy to visualize the project's progress, showing overlapping tasks, dependencies, and how much time is allocated to each stage.

Example: The frontend and backend development tasks are tracked separately, but overlap in the schedule. A delay in backend development will be reflected in the chart, allowing the project manager to shift timelines accordingly.



PLANNING: 

ACTUAL: 

Fig 3.1 : Gantt Chart

3.4 Software Requirement

3.4.1 Developer Side:

3.4.1.1 Development Environment:

- **PyCharm:**

- PyCharm is a widely used integrated development environment (IDE) for Python, which provides tools for developing, testing, and deploying the backend components (machine learning models, APIs).
- **Use:** Backend development (Flask APIs), ML model integration, and debugging.

- **Jupyter:**

Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations, and narrative text. It's a popular tool among data scientists and researchers for developing and presenting Python code.

3.4.1.2 Programming Languages & Frameworks

- **Python:**

- Python is used for developing machine learning algorithms, building the backend (Flask framework), and integrating APIs.
- **Use:** ML models, backend APIs, and system logic.

- **Flask (Python Micro-framework):**

- Flask is a lightweight Python web framework used for building the web application backend.
- **Use:** Web API creation, request handling, and routing.

3.4.1.3 Database Management System

- **MySQL:**

- MySQL is a relational database management system used to store user data, symptoms, health recommendations, prescription history, and OCR results.

- **Use:** Store user information, medical recommendations, and other related data.

3.4.1.4 Testing Tools

- **Postman:**

- Postman is used for API testing, ensuring that the Flask backend and frontend interact seamlessly.
- **Use:** Testing APIs for real-time data exchange between frontend and backend.

3.4.1.5 Operating System

- **Windows/macOS/Linux:**

- The system should be developed on a platform-agnostic environment to ensure compatibility.
- **Use:** Development across multiple operating systems, ensuring flexibility for different developer preferences.

3.4.2 Client side:

These are the software tools needed for end-users to interact with the platform:

1. Web Browser

- **Chrome Browser** (Recommended):

- The platform should be compatible with **Google Chrome**, as it provides high-speed performance and supports modern web standards (HTML5, CSS3, JavaScript). Chrome's developer tools are useful for debugging and performance optimization.

- **Browser Compatibility:**

- The system should also be tested and compatible with **Firefox**, **Safari**, and **Edge** to ensure cross-browser accessibility for all users.

2. Mobile Access

- The platform should be accessible via mobile devices, supporting Android and iOS browsers, especially **Chrome Mobile** and **Safari Mobile**.

3. Internet Connection

- **Minimum Requirement:** A stable internet connection is required for real-time data exchange and fetching recommendations from the system. The platform should work efficiently even on low-bandwidth internet connections (at least 512 kbps).

3.5 Hardware Requirements:

3.5.1 Developer-Side:

To ensure the system is developed efficiently, the following hardware specifications are recommended for developers:

1. Processor

- **Minimum Requirement:** Intel Core i5 or equivalent.
- **Recommended Requirement:** Intel Core i7 or equivalent for better performance during machine learning model training and backend development.

2. RAM

- **Minimum Requirement:** 8 GB.
- **Recommended Requirement:** 16 GB or higher, as machine learning algorithms and data processing require significant memory, especially during model training and testing.

3. Storage

- **Minimum Requirement:** 256 GB SSD.
- **Recommended Requirement:** 512 GB SSD or more, for faster file access, loading datasets, and handling development tools.

4. Graphics Processing Unit (Optional for ML)

- **Optional Requirement:** Dedicated GPU (e.g., NVIDIA GTX 1660 or higher) is recommended for efficient machine learning model training, particularly if large datasets are being processed.

5. Internet Connection

- A stable and fast internet connection is required for accessing online resources, testing APIs, and working collaboratively with GitHub.

6. External Devices

- **Monitor:** A dual-monitor setup is recommended for productivity, enabling developers to view code, documentation, and outputs simultaneously.

Peripherals: Standard keyboard and mouse, with options for ergonomic alternatives to improve long-term productivity.

3.5.1 Client-Side:

For end-users accessing the system, the hardware requirements are relatively lightweight, as the platform will be web-based.

1. Device

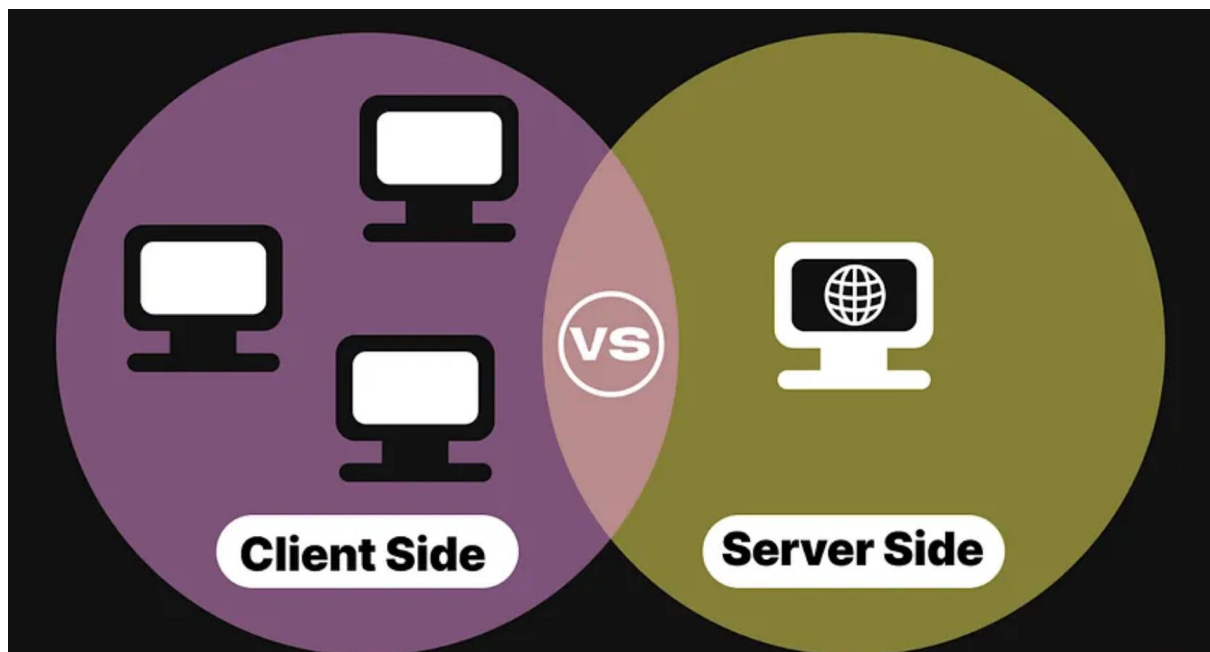
- **PC or Laptop:**
 - Minimum Requirement: A standard laptop or PC with a dual-core processor and 4 GB of RAM.
 - Recommended Requirement: A PC or laptop with a quad-core processor and 8 GB of RAM for a smooth experience.
- **Smartphones/Tablets:**
 - The platform should be responsive and compatible with modern smartphones and tablets.
 - **Minimum Requirement:** Devices running Android 8.0 or iOS 11 or higher.
 - **Screen Size:** The platform should be optimized for mobile devices, with responsiveness for screen sizes ranging from 5-inch smartphones to 10-inch tablets.

2. Operating System

- The platform should be accessible on devices running **Windows, macOS, Linux, Android, and iOS.**

3. Internet Connectivity

Recommended Requirement: A stable internet connection is necessary, with at least 3G/4G for mobile devices and 1 Mbps for desktop access.



3.6 Preliminary Product Description

- **Cover Page:** Features the system logo prominently and provides intuitive navigation to the main platform, ensuring easy access for users.
- **Home Page:** Serves as the central interface where users can input their symptoms. The page displays personalized health recommendations, including suggested medications, dietary adjustments, exercise routines, and precautionary measures, all tailored to individual needs. The layout is designed for user-friendliness, facilitating straightforward interaction and quick access to critical health information.
- **OCR Feature:** Integrates Optical Character Recognition technology to enable users to upload and decode handwritten prescriptions or medical documents. This feature enhances understanding and accessibility of vital health information, ensuring users can easily interpret their medical records.

Advisory Notes: Each recommendation includes a clear note emphasizing the importance of consulting healthcare professionals, reinforcing that the system's suggestions are supplementary and not a substitute for professional medical advice.

3.7 Conceptual Models

3.7.1 Software Development Life Cycle (SDLC)

The Software Development Life Cycle (SDLC) encompasses the entire process of software development, from initial planning through to maintenance and support. The development of the medical recommendation system follows a structured approach to ensure efficient and effective delivery of the final product.

The SDLC includes several key phases:

- **Planning:** Identifying the project scope, objectives, and timelines. This phase involves gathering initial requirements and understanding stakeholder needs.
- **Requirement Analysis:** Detailed analysis of user requirements to ensure that all necessary features are captured and prioritized effectively.
- **Design:** Creating the architecture and design specifications for the system, outlining how components will interact and meet user needs.

- **Development:** Building the system according to the design specifications, focusing on coding and implementing features as defined.
- **Testing:** Conducting various types of testing (unit, integration, system) to ensure the system meets quality standards and functions as intended.
- **Deployment:** Releasing the system to users and ensuring it is operational in the target environment.
- **Maintenance:** Providing ongoing support, updates, and enhancements to address user feedback and changing requirements over time.

3.7.2 Agile Model

The Agile model promotes flexibility and responsiveness by delivering features in short iterations, typically lasting 1 to 3 weeks. This approach is ideal for continuous refinement, enabling the development team to gather user feedback and make necessary adjustments throughout the project lifecycle. The development process in Agile encompasses several key phases:

- **Planning:** Initial discussions with stakeholders to define the project scope, objectives, and timelines.
- **Requirement Analysis:** Gathering and prioritizing user requirements to ensure that the most critical features are addressed first.
- **Design and Development:** Creating a prototype and building the system iteratively, with regular reviews to ensure alignment with user expectations.
- **Unit Testing:** Conducting tests on individual components to verify functionality before integration.
- **Acceptance Testing:** Involving users in testing to validate the system against their needs and expectations, ensuring that the final product meets quality standards.

The Agile approach allows for dynamic adjustments to meet user needs efficiently. By delivering incremental builds with frequent testing, the team can identify and address potential issues early in the process, minimizing the risk of major failures later on.

3.7.3 Why Choose the Agile Model?

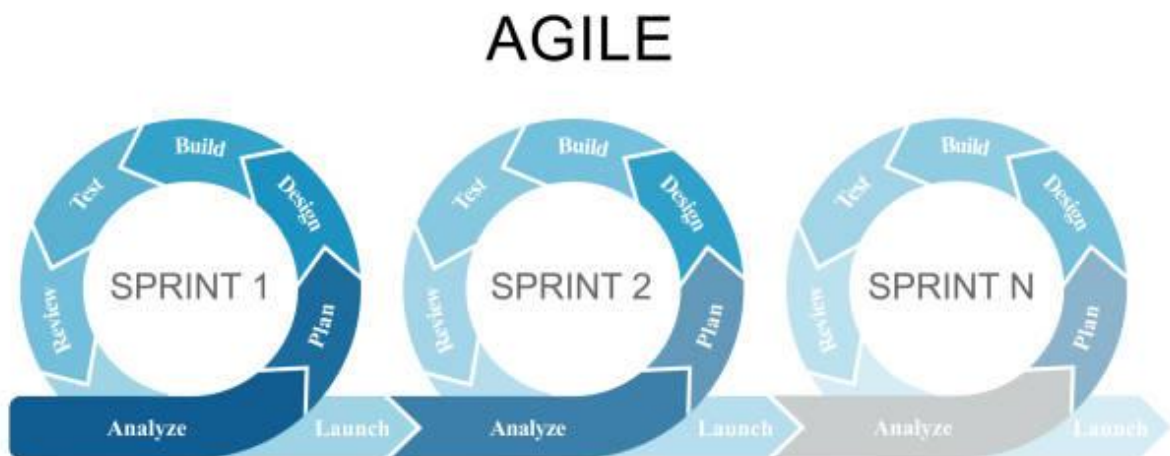
The Agile model provides significant advantages, particularly crucial in healthcare systems where user requirements can evolve rapidly due to emerging needs, regulatory changes, or advancements in

medical practices.

Key benefits of the Agile model include:

- **Flexibility:** Agile allows for quick adaptations in response to user feedback and changing requirements, enabling developers to pivot as necessary without disrupting the overall project timeline.
- **Enhanced Collaboration:** Regular communication and collaboration among stakeholders, developers, and users foster a sense of partnership, ensuring that everyone's insights contribute to the development process.
- **Frequent Deliverables:** By delivering incremental updates, the Agile model provides stakeholders with regular opportunities to review progress, offer feedback, and adjust priorities, ensuring that the final product closely aligns with user needs.

Risk Mitigation: Continuous testing and feedback loops reduce the likelihood of significant failures by identifying issues early, allowing for timely corrections that enhance overall project quality.



Chapter 4 System Design

4.1 Basic Modules

4.1.1 User Authentication Module

- This module manages user registration and login processes, ensuring secure access to the system. New users can create an account by providing their personal details, while existing users can securely log in using their credentials. The module also supports password recovery and account management.
- To ensure that only authorized users can access the system, safeguarding personal health data and maintaining user privacy.

4.1.2 Symptom Input and Analysis Module

- This module is where users input their symptoms through a guided interface. The system prompts users to provide detailed descriptions and selects relevant options from pre-defined symptom categories. Once the symptoms are entered, the system uses machine learning algorithms to analyze the data and generate a preliminary diagnosis.
- To gather and analyze user-reported symptoms accurately, forming the basis for generating personalized healthcare recommendations.

4.1.3 Personalized Recommendation Module

- Based on the symptoms analyzed, this module generates personalized healthcare recommendations, including suitable medications, diet plans, exercise routines, and precautions. The system pulls data from a comprehensive medical database, ensuring the advice is accurate and relevant to each user's condition.
- To deliver tailored health advice that meets the specific needs of users, enhancing their overall health management experience.

4.1.4 Optical Character Recognition (OCR) Module

- This module allows users to upload images of handwritten medical prescriptions or notes. The OCR technology scans these images, converts the handwriting into digital text, and interprets it to extract important medical information, such as prescribed medications and dosages.
- To make medical prescriptions more accessible by converting handwritten notes into readable digital text, thereby aiding users in understanding and managing their treatment plans.

4.1.5 Medication Information and Interaction Module

- This module provides detailed information about the medications recommended by the system.
- To ensure users are well-informed about the medicines they are taking, enabling them to make safer and more effective treatment decisions.

4.1.6 User Dashboard Module

- The User Dashboard serves as the central hub for managing user interactions with the system. It displays the user's health history, symptoms logged, recommendations provided, and any uploaded prescriptions. The dashboard is designed to be user-friendly and customizable, allowing users to access their health data efficiently.
- To provide users with a personalized space where they can easily manage their health records and access recommendations, facilitating continuous and proactive health management.

4.2 Database Design

4.2.1 Users table design

Field Name	Data type	Constraints
User_ID	String	Primary Key, Unique, Not null
Name	String	Not null
Email	Varchar	Unique, Not null
Password	Varbinary	Not null

4.2.2 Symptoms table design

Field Name	Data type	Constraints
Symptom_ID	Int	Primary Key, Unique, Not null
User_ID	String	Foreign Key, Not null
Symptom_Description	Text	Not null
Date_Reported	DateTime	Not null

4.2.3 Recommendations table design

Field Name	Data type	Constraints
Recommendation_ID	Int	Primary Key, Unique, Not null
User_ID	String	Foreign Key, Not null
Medicine_Recommendati on	Text	Not null
Diet_Recommendation	Text	Not null
Exercise_Recommendatio n	Text	Not null
Date_Generated	DateTime	Not null

4.2.4 OCR Data table design

Field Name	Data type	Constraints
OCR_ID	Int	Primary Key, Unique, Not null
User_ID	String	Foreign Key, Not null
Original_Text	Text	Not null
Processed_Text	Text	Not null
Date_Uploaded	DateTime	Not null

4.2.5 Recommendation table design

Field Name	Data type	Constraints
Recommendation_ID	Int	Foreign Key, Not null
Medicine_ID	Int	Foreign Key, Not null

4.2.6 Data Integrity and Constraints

Data integrity refers to ensuring the accuracy, consistency, and reliability of data throughout its life cycle in the medicine recommendation system. This ensures that all user data, symptoms, prescriptions, and recommendations remain accurate and unaltered across various interactions with the system. Maintaining data integrity is critical to the design, implementation, and usage of a system that stores and processes sensitive healthcare data.

Types of Data Integrity:

1. **Entity Integrity:** Ensuring that each table has a primary key that uniquely identifies a record. For instance, every user must have a unique `User_ID`, and each medicine must have a unique `Medicine_ID`.
2. **Referential Integrity:** Ensuring relationships between tables are valid, such as ensuring a valid `User_ID` exists when a symptom or recommendation is entered.
3. **Domain Integrity:** Ensuring that data entered into fields conforms to specific data types and formats, like ensuring `Email` is in the correct format and `Date_Reported` is a valid date.
4. **User Authentication and Access Control:** Implementing account validation for users, ensuring only authorized users access sensitive data such as recommendations and prescriptions.

Validity checks and constraints applied to maintain data integrity :

- **Account Validation:** Users must pass through authentication (e.g., username and password verification) to access their personalized recommendations and medical data. The system ensures the user's identity through secure login credentials.
- **Input Validation:** Whenever a user reports symptoms, uploads prescriptions, or updates personal information, the system checks the validity of the input. For example, symptoms cannot be blank, dates must be valid, and medicine dosages should follow pre-defined formats.
- **Data Type Constraints:** The system enforces specific data types for each field (e.g., `String`, `Int`, `DateTime`) and ensures that only valid data is stored in the database.

Foreign Key Constraints: The system uses foreign key constraints to maintain relationships between tables. For instance, every `Symptom` must be associated with a valid `User_ID`, ensuring that no orphaned records exist in the database.

4.3 Procedural Design

4.3.1 Logic Diagrams

4.3.1.1 Use case diagram.

- Use case Diagram for developer side and user.

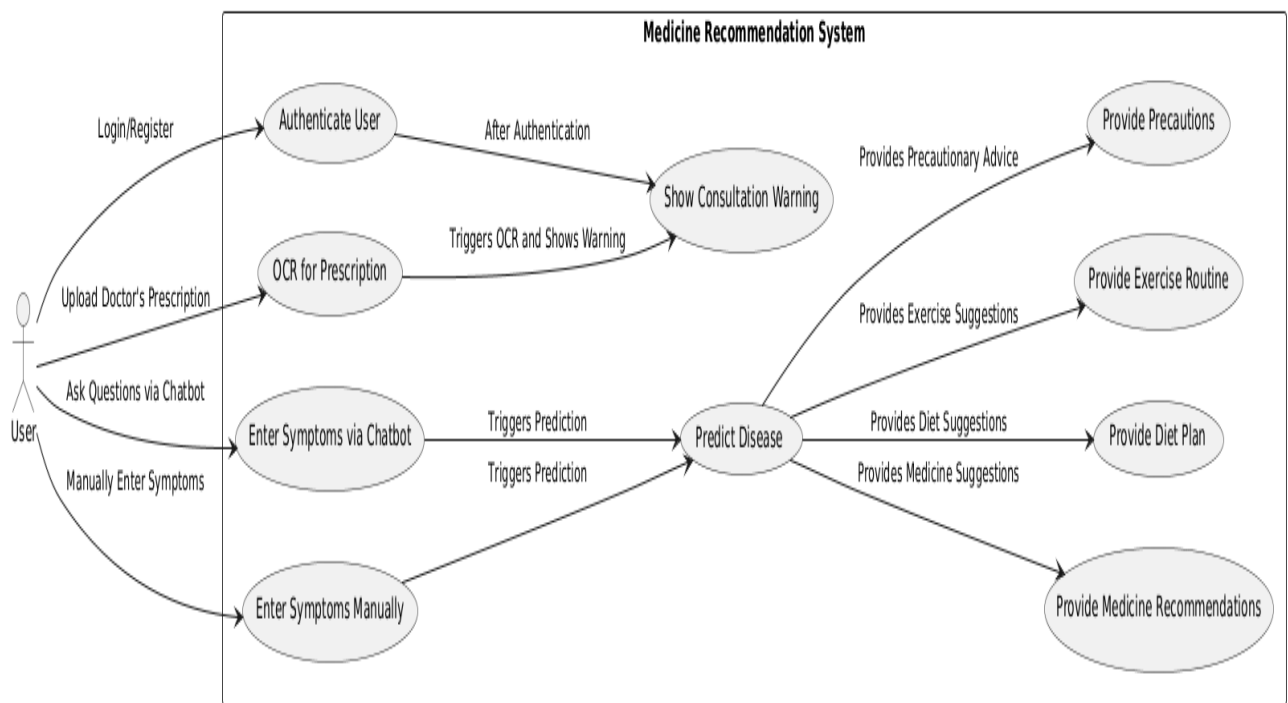


Fig. 4.1: Use case diagram.

4.3.1.2 Activity diagram

- Activity Diagram:

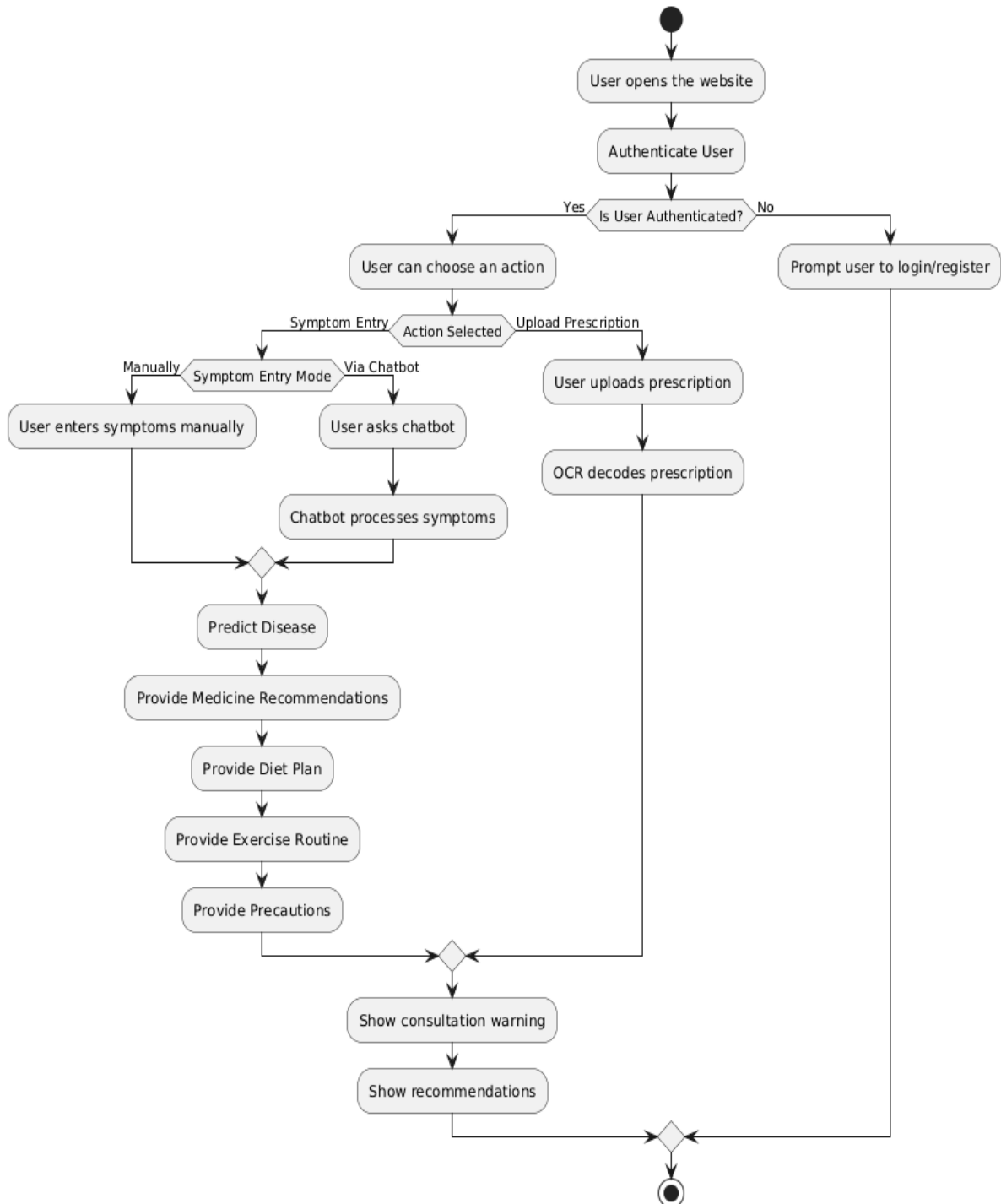


Fig. 4.2: Activity Diagram

4.3.1.3 Sequence diagram

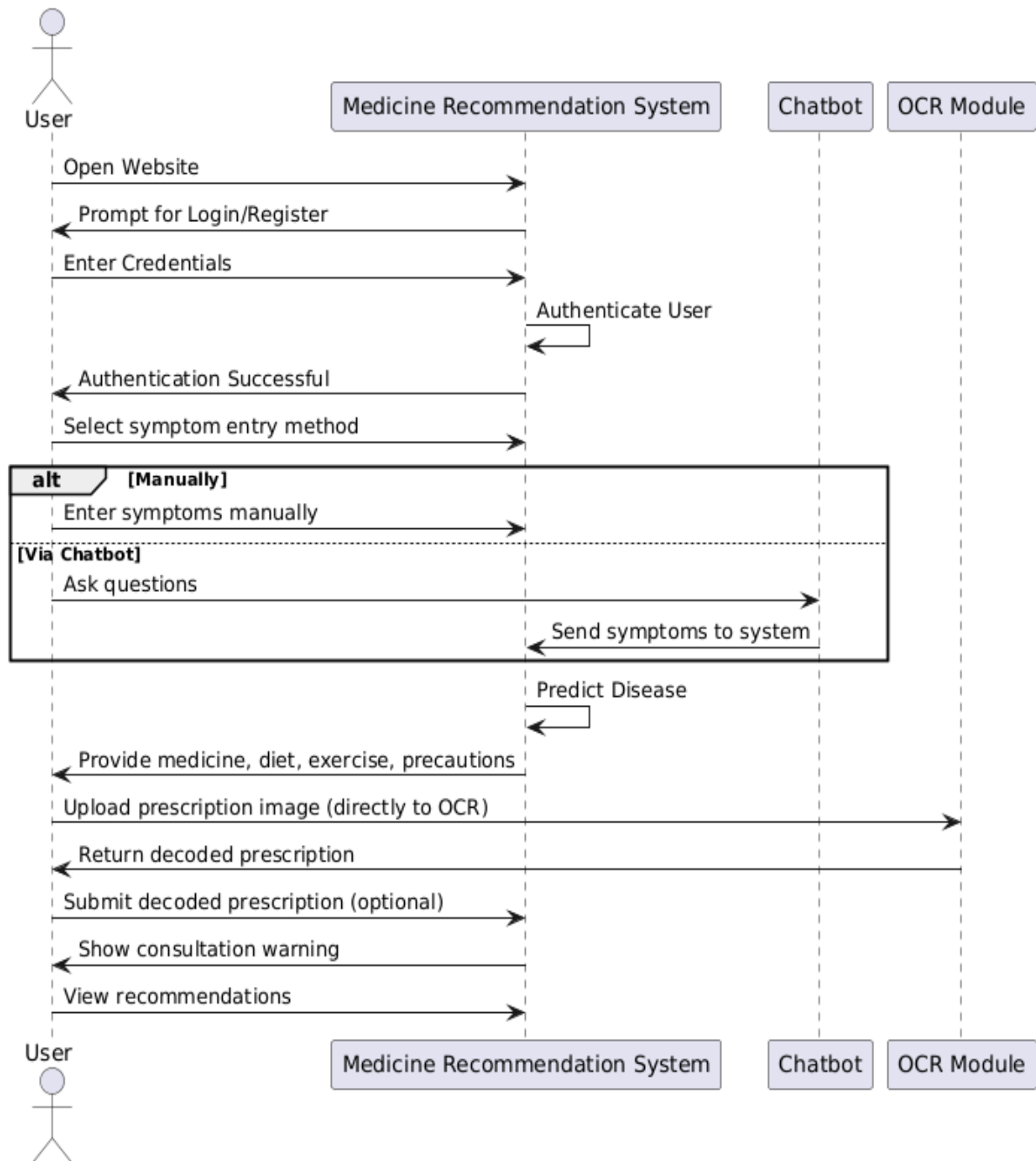


Fig. 4.3: Sequence Diagram

4.3.1.4 Entity Relationship Diagram

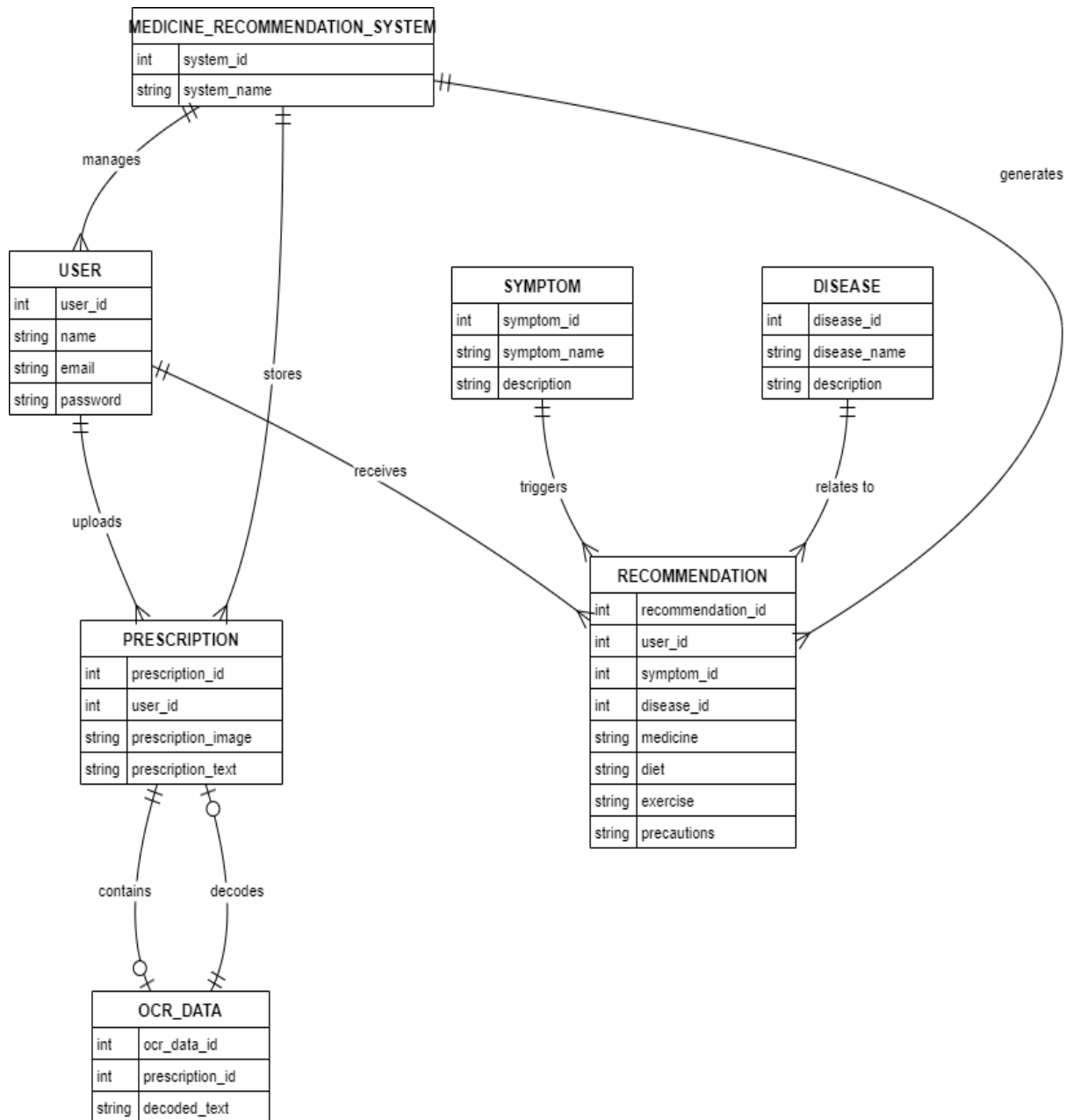


Fig. 4.4: Entity Relationship Diagram

4.4 User interface design

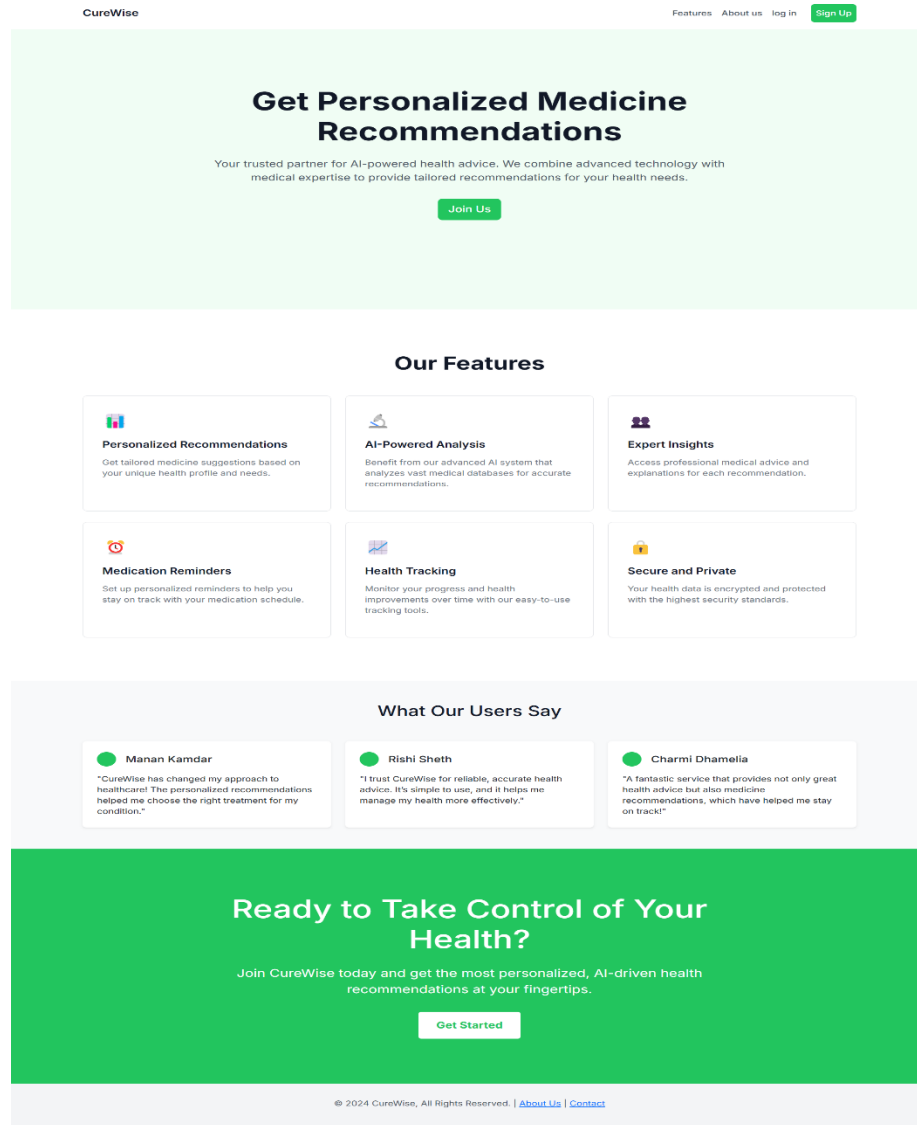


Fig. 4.5: Curewise-Landing Page

CureWise

[Home](#) [About us](#) [Log In](#) [Sign Up](#)

Create a CureWise account

Enter your details to create your CureWise account

Full Name

John Doe

Email

jhash2501@gmail.com

Password

☐ At least 8 characters

☐ One uppercase letter

☐ One lowercase letter

☐ One number

☐ One special character (!@#\$%^&*)

Confirm Password

Sign Up

Already have an account? Log in

© 2024 CureWise. All rights reserved.

[Terms of Service](#) [Privacy](#)

Fig. 4.6: Curewise-Login/SignUp page

4.5 Security Issues

There are some primary methods used by cyber criminals for conducting criminal activity.

Below are some methods:

- 1. Botnets:** A botnet is a network of computers that attackers infect with malware, compromised, and connected to a central command & control center. The attackers enlist more and more devices into their botnet, and use them to send spam emails, conduct DDoS attacks, click fraud, and crypto mining.
- 2. Ransomware and other malware:** Ransomware is malware that encrypts data on a local machine and demands a ransom to unlock it. There are hundreds of millions of other types of malwares that can cause damage to end-user devices and result in data exfiltration.
- 3. Phishing and other social engineering attacks:** Phishing involves sending misleading messages via email or other channels, which causes internet users to provide personal information, access malicious websites or download malicious payloads.

- 4. Fraud and identity theft:** Fraud is the theft of funds by an attacker pretending to be the owner of an account or using stolen cards or credentials. Identity theft is a related concept and involves compromising a user's online accounts to enable an attacker to perform actions in their name.
- 5. Flood attacks-** Most modern flood attacks are DDoS attacks, which leverage a botnet to hit a website or organization with massive amounts of fake traffic. Flood attacks can be targeted at the network layer, choking an organization's bandwidth and server resources, or at the application layer, bringing down a database or email server for example.
- 6. Browser hijacking-** Attacks like cross site scripting (XSS) can cause malicious code to run in a user's browser. This can result in session hijacking, drive-by downloads and other illicit activity carried out in the user's browser without their consent.

4.6 Test Cases Design

Table 4.2 Test cases

No	Name	Test Case	Input	Expected Output
1.	User Authentication Page	Registered email address should be stored in database	-	Registered Successfully
		Authentication of password is done according to the email ID		Home page to be displayed

2.	Home	Disease or	Title,	Disease
	page	Symptom is	Description,	displayed
		entered by	Image URL,	according
		user in search	Goal,	to the
		Drop	deadline	entered text
		down	date.	Symptom
		Box		Medicines
				according to
				your input is
				Displayed.

Chapter 5

Implementation and Testing

5.1 Implementation Approaches:

1. 1. System Initialization & Global Configuration

```
# Initialize the Flask app, enable CORS, and configure logging.
app = Flask(__name__)
CORS(app) # Enable cross-origin requests globally
app.config['SECRET_KEY'] = os.urandom(24)
logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s -
%(message)s')
```

Description:

This snippet sets up the core Flask application by generating a dynamic secret key for enhanced security. CORS is enabled to allow cross-origin requests, and logging is configured to capture and format runtime events for debugging purposes.

2. Database and Mail Service Setup

```
# Configure MySQL database and email settings.
app.config['MYSQL_HOST'] = '127.0.0.1'
app.config['MYSQL_USER'] = 'root'
app.config['MYSQL_PASSWORD'] = 'admin'
app.config['MYSQL_DB'] = 'curewise'
mysql = MySQL(app)

app.config['MAIL_SERVER'] = 'smtp.gmail.com'
app.config['MAIL_PORT'] = 587
app.config['MAIL_USE_TLS'] = True
app.config['MAIL_USERNAME'] = 'your_email@example.com'
app.config['MAIL_PASSWORD'] = 'your_email_password'
mail = Mail(app)
```

Description:

This code connects the application to a MySQL database for persistent storage and sets up SMTP mail services

to send confirmation and password reset emails, ensuring reliable user communication.

3. Third-Party Integration and Model Loading

```
# Initialize Firebase and load machine learning models for prediction.
cred = credentials.Certificate("path/to/firebase-adminsdk.json")
firebase_admin.initialize_app(cred)

mild_model = load_model("path/to/mild_model.h5")
moderate_model = load_model("path/to/moderate_model.h5")
severe_model = load_model("path/to/severe_model.h5")
```

Description:

This snippet integrates Firebase for additional backend functionality and loads three pre-trained ML models. Each model is dedicated to predicting diseases at different severity levels based on user input.

4. Data Preparation: Reading and Merging CSV Files

```
# Load CSV files, normalize disease names, and merge into a unified dataset.
desc_df = pd.read_csv("Disease_description.csv")
diet_df = pd.read_csv("Diseases_with_Diet_Plans.csv")
workout_df = pd.read_csv("Diseases_with_Workout_Plans.csv")
med_df = pd.read_csv("medicine.csv", encoding='latin1')

for df in [desc_df, diet_df, workout_df, med_df]:
    df['disease'] = df['disease'].str.strip().str.lower()

disease_data = desc_df.merge(diet_df, on='disease')\
                    .merge(workout_df, on='disease')\
                    .merge(med_df, on='disease')
```

Description:

Multiple CSV files containing various disease-related data are read into DataFrames, normalized by cleaning the disease names, and merged to create a comprehensive mapping. This unified dataset is later used to correlate prediction outputs with detailed disease information.

5. Unified Signup and Login Endpoints

```
# Define forms for user signup and login, and implement their endpoints.
class SignupForm(FlaskForm):
    name = StringField('Name', validators=[DataRequired()])
    email = StringField('Email', validators=[DataRequired(), Email()])
    password = PasswordField('Password', validators=[DataRequired()])
    confirm_password = PasswordField('Confirm Password', validators=[DataRequired(),
    EqualTo('password')])
    submit = SubmitField('Sign Up')

class LoginForm(FlaskForm):
    email = StringField('Email', validators=[DataRequired(), Email()])
    password = PasswordField('Password', validators=[DataRequired()])
    submit = SubmitField('Log In')

@app.route('/signup', methods=['GET', 'POST'])
def signup():
    form = SignupForm()
    if form.validate_on_submit():
        # Process registration: insert user into DB, hash password, generate & send
confirmation token.
        pass
    return render_template('signup.html', form=form)

@app.route('/login', methods=['GET', 'POST'])
def login():
    form = LoginForm()
    if form.validate_on_submit():
        # Authenticate user: validate credentials and establish session.
        pass
    return render_template('login.html', form=form)
```

Description:

This combined snippet handles both user registration and login. It employs Flask-WTF forms to capture and validate user input. During signup, the process includes password hashing and sending an email confirmation token. The login endpoint validates credentials and manages user sessions.

6. JWT Token Generation and Confirmation


```
def generate_confirmation_token(email: str) -> str:
    payload = {'email': email, 'exp': datetime.utcnow() + timedelta(hours=24)}
    token = jwt.encode(payload, app.config['SECRET_KEY'], algorithm='HS256')
    return token if isinstance(token, str) else token.decode('utf-8')

def confirm_token(token: str) -> str:
    try:
        payload = jwt.decode(token, app.config['SECRET_KEY'], algorithms=['HS256'])
        return payload.get('email')
    except (jwt.ExpiredSignatureError, jwt.InvalidTokenError):
        return None
```

Description:

These helper functions create and validate JWT tokens for email confirmation. Tokens are set to expire after 24 hours, thereby adding an extra layer of security by ensuring that only users with valid tokens can confirm their accounts.

7. Input Preprocessing for Symptom Analysis

```
def preprocess_input(symptoms_list, symptom_columns):
    clean_map = {col.replace("Symptom_", ""): col for col in symptom_columns}
    input_vector = np.zeros(len(symptom_columns))
    for symptom in symptoms_list:
        if symptom in clean_map:
            index = symptom_columns.get_loc(clean_map[symptom])
            input_vector[index] = 1
    return input_vector
```

Description:

This function processes the list of symptoms provided by the user by converting it into a binary vector. The mapping of simplified symptom names to their corresponding columns ensures that the machine learning model receives input in the expected format.

8. Disease Prediction Endpoint (Snippet)

```
@app.route('/submit_symptoms', methods=['POST'])
def submit_symptoms():
    if 'id' not in session:
```

```

        return jsonify({"success": False, "error": "Log in required."}), 403
    data = request.get_json()
    symptoms = [s.strip() for s in data.get('symptoms', []) if s.strip()]
    severity = data.get('severity', '1')
    if severity == '1':
        vector = preprocess_input(symptoms, mild_symptom_columns)
        # Scale input and predict using the mild model (internal details abstracted)
        # Map predictions to detailed disease info using the unified dataset
    return jsonify({"success": True, "data": "mapped_disease_details"})

```

Description:

This endpoint handles symptom submission from logged-in users. It validates the session, preprocesses the symptom input, chooses the appropriate machine learning model based on the severity level, and returns mapped disease details from the merged dataset.

9. Chatbot Nutrition Plan Integration

```

def get_chatbot_nutrition_plan(disease):
    prompt = f"Provide a concise nutrition plan for {disease} in 40-50 words without extra commentary."
    response = requests.post(f"{OLLAMA_URL}/v1/completions", json={
        "model": "deepseek-r1:7b", "prompt": prompt, "max_tokens": 100, "temperature":
0.7
    }, timeout=200)
    generated = response.json().get("response", "").strip()
    return {"disease": disease, "diet": generated or "No response provided."}

```

Description:

This function creates a prompt and interacts with an external NLP model (chatbot) to generate a short nutrition plan for a specified disease. The output is returned in a structured JSON format.

10. OCR Endpoint (Simplified)

```

@app.route('/ocr', methods=['GET', 'POST'])
def ocr():
    if request.method == 'GET':
        # Process OCR from a captured file (implementation details abstracted)
        pass

```

```

else:
    # Handle uploaded file (image, PDF, DOC) and extract text
    return jsonify({"success": True, "text": "extracted_text"})

```

Description:

This endpoint supports OCR functionality, allowing the extraction of text from captured or uploaded files using an external OCR API. The logic is abstracted to protect full implementation details.

11. Video Streaming Endpoint

```

def gen_frames():
    cap = cv2.VideoCapture(0)
    while video_feed_active:
        ret, frame = cap.read()
        # Overlay dynamic status information on the frame.
        ret, buffer = cv2.imencode('.jpg', frame)
        yield (b'--frame\r\nContent-Type: image/jpeg\r\n\r\n' + buffer.tobytes() +
b'\r\n\r\n')
    cap.release()

@app.route('/video_feed')
def video_feed():
    global video_feed_active
    video_feed_active = True
    return Response(gen_frames(), mimetype='multipart/x-mixed-replace; boundary=frame')

```

Description:

This code streams live video from the webcam. Frames are captured, optionally overlaid with dynamic information (e.g., document coverage status), and sent as an MJPEG stream to the client.

12. Image Capture Endpoint

```

@app.route('/capture_image', methods=['POST'])
def capture_image():
    cap = cv2.VideoCapture(0)
    ret, frame = cap.read()
    cap.release()
    ret2, buffer = cv2.imencode('.png', frame)

```

```
image_b64 = base64.b64encode(buffer).decode('utf-8')
return jsonify({"success": True, "image": image_b64})
```

Description:

This endpoint allows users to capture a still image from the webcam. The image is encoded as a PNG, converted to a base64 string, and returned in JSON format for use in the client interface.

13. Text Analysis – Spell Correction

```
def correct_text(text: str) -> str:
    spell = SpellChecker()
    corrected = []
    for word in text.split():
        corrected.append(word if word.lower() in spell or len(word) < 3 else
spell.correction(word))
    return " ".join(corrected)
```

Description:

This function processes input text to correct misspelled words using a spell-checker. It helps ensure that subsequent text analysis or summary generation works with clean, accurate text.

14. Text Analysis – Summary Generation

```
def generate_summary(text: str) -> str:
    prompt = "Provide a concise summary for the following medical text: " + text
    summary_response = requests.post(f"{OLLAMA_URL}/v1/completions", json={
        "model": "deepseek-r1:7b", "prompt": prompt, "max_tokens": 150, "temperature":
0.3
    }, timeout=200)
    summary = summary_response.json().get("response", "").strip()
    return summary
```

Description:

This function generates a concise summary of medical text by sending a prompt to an external NLP model. The resulting summary is extracted from the response and returned for display.

15. HTML Template – Mild Symptoms Page

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Mild Symptoms - CureWise</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
  <link
href="https://fonts.googleapis.com/css2?family=Inter:wght@400;500;600&display=swap"
rel="stylesheet">
  <link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-
beta3/css/all.min.css" rel="stylesheet">
  <style>
    .footer { background-color: #F3F4F6; padding: 2rem 1rem; text-align: center; color:
#4B5563; }
    body { font-family: 'Inter', sans-serif; background-color: #F0FDF4; }
    .header { background-color: white; box-shadow: 0 2px 4px rgba(0,0,0,0.1); }
    .logo { color: #22C55E; font-weight: bold; }
    .main-content { padding: 2rem; }
    .card { background-color: white; border-radius: 0.5rem; box-shadow: 0 2px 4px
rgba(0,0,0,0.1); }
  </style>
</head>
<body>
  <header class="header py-3">
    <div class="container">
      <div class="d-flex justify-content-between align-items-center">
        <a href="{{ url_for('index') }}" class="logo text-decoration-none">
          <i class="fas fa-heartbeat me-2"></i>CureWise
        </a>
        <nav>
          <a href="{{ url_for('severity_selection') }}" class="btn btn-outline-primary
me-2">Back to Severity Selection</a>
          <a href="{{ url_for('index') }}" class="btn btn-outline-secondary">Back to
Home</a>
        </nav>
      </div>
    </div>
  </header>
  <main class="main-content">
```

```

<div class="container">
  <h1 class="mb-4 text-center">Mild Symptoms</h1>
  <div class="card p-4">
    <form id="symptom-form">
      <div id="symptom-fields">
        <!-- Dynamically generated symptom fields -->
      </div>
      <button type="submit" class="btn btn-primary mt-3">Submit</button>
    </form>
    <div id="prediction-result" class="mt-4"></div>
    <div id="flash-response" style="display: none;"></div>
  </div>
</div>
</main>
<footer class="footer">
  <div class="container">
    <p>&copy; 2024 CureWise. All rights reserved.</p>
    <p>
      <a href="static/CureWise_Terms_of_Service.pdf">Terms of Service</a> |
      <a href="static/CureWise_Privacy_Policy.pdf">Privacy Policy</a>
    </p>
  </div>
</footer>
<script>
  // JavaScript for dynamic symptom field addition and form submission.
  const symptomOptions = [/* List of symptom options */];
</script>
</body>
</html>

```

Description:

This HTML template is for the Mild Symptoms page. It employs Bootstrap for styling, incorporates custom CSS, and uses Flask template syntax to integrate dynamic content. The page allows users to input their symptoms, which will then be processed by the backend.

16. Landing Page (landing.html)

```

<!DOCTYPE html>
<html lang="en">
<head>

```

```

<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Welcome to CureWise</title>
<link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">
</head>
<body>
  <header class="bg-primary text-white p-4 text-center">
    <h1>Welcome to CureWise</h1>
  </header>
  <main class="container my-5">
    <p>Embark on your journey to better health with our innovative solutions.</p>
    <a href="{{ url_for('signup') }}" class="btn btn-success">Sign Up</a>
    <a href="{{ url_for('login') }}" class="btn btn-outline-primary">Log In</a>
  </main>
  <footer class="text-center p-3 bg-light">
    <small>&copy; 2024 CureWise. All rights reserved.</small>
  </footer>
</body>
</html>

```

Description:

The Landing page offers a warm introduction to CureWise with clear call-to-action buttons for signup and login. It is styled using Bootstrap and serves as the primary entry point for new users.

17. Chatbot Page (chatbot.html)

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Chat with CureWise</title>
  <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">
</head>
<body>
  <header class="bg-info text-white p-3 text-center">
    <h2>CureWise Chatbot</h2>
  </header>
  <div class="container mt-4">

```

```

    <div id="chat-window" class="border rounded p-3" style="height:400px; overflow-
y:scroll;">
        <!-- Chat messages appear here -->
    </div>
    <form id="chat-form" class="mt-3">
        <input type="text" id="chat-input" class="form-control" placeholder="Type your
message...">
        <button type="submit" class="btn btn-primary mt-2">Send</button>
    </form>
</div>
</body>
</html>

```

Description:

This dedicated Chatbot page provides an interactive interface where users can communicate with the CureWise assistant. It includes a scrollable chat window and a simple form for sending messages, all styled with Bootstrap.

18. About Page (about.html)

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>About CureWise</title>
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">
</head>
<body>
    <header class="bg-secondary text-white p-4 text-center">
        <h1>About CureWise</h1>
    </header>
    <main class="container my-5">
        <p>CureWise is dedicated to providing innovative healthcare solutions by integrating
advanced machine learning with user-friendly interfaces. Our platform is designed to
offer personalized disease predictions, nutritional plans, and more.</p>
    </main>
    <footer class="text-center p-3 bg-light">
        <small>&copy; 2024 CureWise. All rights reserved.</small>
    </footer>

```



```
</footer>
</body>
</html>
```

Description:

The About page provides users with information about the CureWise project, its mission, and the technologies behind it. It is designed to be informative yet simple, giving users context about the platform.

5.2 TESTING APPROACH

To ensure the quality, reliability, and performance of the **CureWise** platform, we employed a comprehensive testing approach that includes **unit testing** and **integration testing**. These methodologies were applied throughout the development lifecycle to detect issues early, validate individual components, and ensure seamless interaction among different system modules.

5.2.1 Testing Methodologies

Table 6.1: Test Report Summary

EXECUTED	PASSED	3	
	FAILED	1	
	Total Tests Executed (Passed + Failed)	4	
PENDING			0
IN PROGRESS			0
BLOCKED			0
TEST PLANNED (PENDING + IN PROGRESS + BLOCKED +TEST EXECUTED)			4
DEFERRED			0

Unit Testing:

Unit tests are designed to validate the functionality of individual components or functions in isolation. Each function is tested with various inputs to ensure it behaves as expected. For example, we verify that the

symptom preprocessing function converts input strings into the correct binary vector and that text correction accurately fixes common misspellings.

Integration Testing:

Integration tests focus on the interactions between different modules and endpoints. These tests simulate complete workflows—such as user registration, login, symptom submission, and disease prediction—to confirm that data flows correctly through the system and that components work together seamlessly. Additionally, endpoints that interact with external services (OCR, chatbot, etc.) are tested to ensure proper integration.

5.2.2 Test Cases

Below are nine representative test cases that cover critical functionalities of the CureWise platform. Each test case is presented in a tabular format, detailing the **Test ID**, **Module/Endpoint**, **Test Type**, **Objective**, **Pre-Conditions**, **Test Steps**, **Expected Outcome**, and a placeholder for **Actual Outcome/Status**.

Test Case 1: Disease Prediction Endpoint

Test ID	IT-DP-001
Module/Endpoint	POST /submit_symptoms
Test Type	Integration
Objective	Verify that symptom submission correctly triggers disease prediction using the appropriate ML model and maps predictions to detailed disease data.
Pre-Conditions	- User must be logged in.- mild_symptom_columns, scaler, and mild model are properly configured.- Unified disease_data exists.
Test Steps	1. Log in as a valid user.2. Submit a POST request with payload: {"symptoms": ["fever", "cough"], "severity": "1"}.3. Inspect the JSON response for predicted disease details.
Expected Outcome	The response should indicate success and return a JSON object with disease details (e.g., disease name and description) mapped from the dataset using the mild model.
Actual	<i>Pass</i>

Outcome/Status	
-----------------------	--

Test Case 2: Diet Prediction Endpoint

Test ID	IT-DIET-002
Module/Endpoint	POST /disease/diet
Test Type	Integration
Objective	Ensure that the diet endpoint returns the correct diet plan for a given disease.
Pre-Conditions	- User must be logged in.- Unified <code>disease_data</code> includes diet plan details.
Test Steps	1. Log in as a valid user.2. Submit a POST request with payload: {"diseases": ["diabetes"]}.3. Inspect the JSON response for diet details and any additional information.
Expected Outcome	The response should include a valid diet plan and additional information extracted from the dataset (or a chatbot-generated plan if necessary).
Actual Outcome/Status	<i>Pass</i>

Test Case 3: Workout Endpoint

Test ID	IT-WO-003
Module/Endpoint	POST /disease/workout
Test Type	Integration
Objective	Verify that the workout endpoint returns appropriate workout recommendations for the provided disease.
Pre-Conditions	- User must be logged in.- The <code>disease_data</code> contains workout plan details for the disease.
Test Steps	1. Log in as a valid user.2. Submit a POST request with payload: {"disease": "hypertension"}.3. Inspect the JSON response for workout recommendations.
Expected Outcome	The endpoint should return a JSON response with a valid workout plan or a message indicating that no suggestions are available.
Actual Outcome/Status	<i>Pass</i>

Test Case 4: Medicine Endpoint

Test ID	IT-MED-004
Module/Endpoint	POST /disease/medicine
Test Type	Integration
Objective	Verify that the medicine endpoint returns appropriate medicine recommendations based on the input disease.
Pre-Conditions	- User must be logged in.- The <code>disease_data</code> includes medicine recommendations.
Test Steps	1. Log in as a valid user.2. Submit a POST request with payload: {"disease": "asthma"}.3. Inspect the JSON response for medicine details.
Expected Outcome	The response should include medicine recommendations from the dataset or indicate that no recommendations are available.
Actual Outcome/Status	<i>Pass</i>

Test Case 5: Description Endpoint

Test ID	IT-DESC-005
Module/Endpoint	POST /disease/description
Test Type	Integration
Objective	Ensure that the description endpoint returns detailed information (description, cause, management) for a given disease.
Pre-Conditions	- User must be logged in.- The unified <code>disease_data</code> includes comprehensive descriptions.
Test Steps	1. Log in as a valid user.2. Submit a POST request with payload: {"diseases": ["hypertension"]}.3. Verify that the JSON response includes description, cause, and management details.
Expected Outcome	The response should return a JSON object with accurate and complete details for the requested disease.
Actual Outcome/Status	<i>Pass</i>

Test Case 6: Chatbot Endpoint

Test ID	IT-CHAT-006
Module/Endpoint	POST /generate
Test Type	Integration
Objective	Verify that the chatbot endpoint returns a concise and accurate response (e.g., a nutrition plan) for a given query.
Pre-Conditions	- User must be logged in.- External chatbot API is available.
Test Steps	1. Log in as a valid user.2. Submit a POST request with payload: {"prompt": "nutrition plan for diabetes"}.3. Inspect the JSON response for a clear, coherent answer.
Expected Outcome	The response should provide a direct answer (approximately 40–50 words) relevant to the query.
Actual Outcome/Status	Pass

Test Case 7: OCR Endpoint

Test ID	IT-OCR-007
Module/Endpoint	POST /ocr
Test Type	Integration
Objective	Verify that the OCR endpoint accurately extracts text from an uploaded file (image, PDF, or DOC).
Pre-Conditions	- User must be logged in.- A valid file is available for upload.
Test Steps	1. Submit a POST request with a valid file (e.g., a scanned document image).2. Inspect the JSON response for the extracted text.3. Ensure the text is relevant and non-empty.
Expected Outcome	The response should return "success": True and non-empty "text" containing the extracted content.
Actual Outcome/Status	Pass

Test Case 8: OpenCV Image Capture Endpoint

Test ID	IT-OPCV-008
Module/Endpoint	POST /capture_image

Test Type	Integration
Objective	Confirm that the image capture endpoint uses OpenCV to capture a webcam image and returns a valid base64-encoded image.
Pre-Conditions	- User must be logged in.- A webcam is connected and functional.
Test Steps	1. Attempt a POST request to <code>/capture_image</code> without a valid session (should be rejected).2. Log in as a valid user and reattempt the request.3. Inspect the JSON response for base64-encoded image data.
Expected Outcome	Unauthorized requests are rejected (e.g., with a 403 error).Authorized requests return a JSON object containing the base64 image data.
Actual Outcome/Status	Pass

Test Case 9: User Authentication – Signup/Login Flow

Test ID	IT-AUTH-009
Module/Endpoint	POST <code>/signup</code> and POST <code>/login</code>
Test Type	Integration
Objective	Validate that the full user authentication flow (registration and login) functions correctly.
Pre-Conditions	- Unique user data.- Proper database configuration.
Test Steps	1. Submit a POST request to <code>/signup</code> with valid user details (name, email, password, confirm_password).2. Verify the response indicates account creation and that the new user has <code>is_verified = 0</code> .3. Simulate email confirmation or bypass for testing.4. Submit a POST request to <code>/login</code> with the correct credentials.5. Verify that the session is established and a success message is returned.
Expected Outcome	- Successful signup creates a new user entry and generates a confirmation token.- Successful login returns a message like "Logged in successfully." and sets the session correctly.
Actual Outcome/Status	Pass

Chapter 6

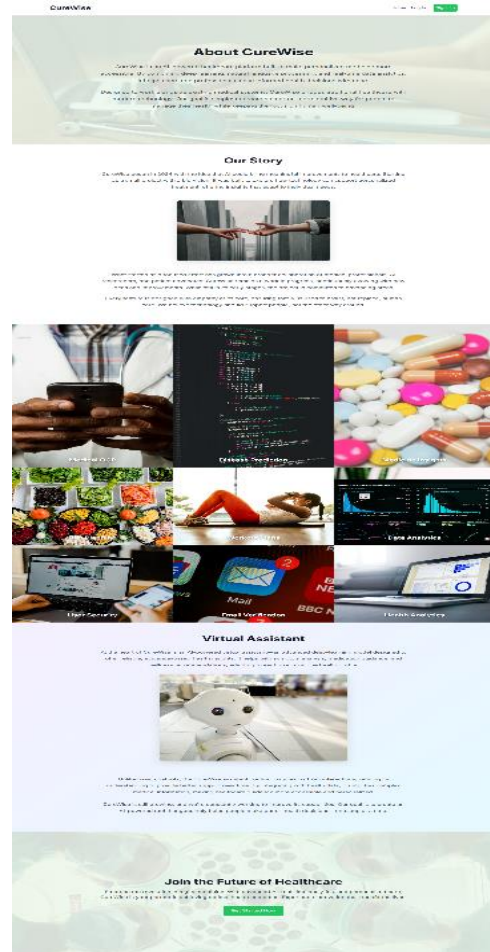
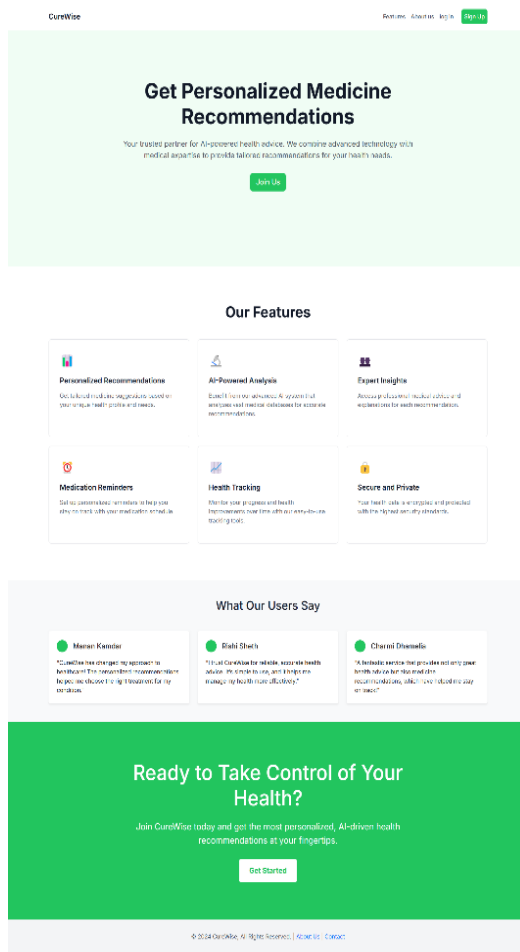
Implementation and Interface Design

6.1 Overview

The CureWise platform is meticulously designed to provide a seamless and intuitive user experience. Each page has been carefully crafted with an emphasis on clarity, ease of navigation, and consistency in branding. The design ensures that users—from first-time visitors to returning members—can quickly locate the features they need. This chapter describes key interfaces, including the Landing & About Us pages, Login & Signup pages, Dashboard, Disease Prediction, Medicine Recommendation, Nutrition Plan, Workout Plan, Disease Description, OCR & Document Analysis, Chatbot, and Medical Text Analyzer. The following sections detail each component, referencing corresponding images as Figures.

6.2 Landing & About Us Pages

Figure 6.1:



6.2.1 Landing Page

- **Purpose:**

The Landing page is the first point of engagement for new users. It is designed to make an immediate impression by clearly presenting CureWise’s value proposition and guiding visitors to take action—either by signing up or logging in.

- **Detailed Design Elements:**

- **Headline & Tagline:** A bold headline captures attention, succinctly conveying the platform’s promise (e.g., “Personalized Health Insights at Your Fingertips”). The tagline further clarifies the benefits, such as the use of AI for personalized recommendations.

- **Call-to-Action Buttons:** Strategically placed buttons (e.g., “Sign Up” and “Log In”) are designed with contrasting colors to draw immediate attention and invite user interaction.
 - **Feature Overview:** Brief sections highlight CureWise’s core functionalities such as disease prediction, personalized medicine, nutrition and workout plans, and an interactive chatbot. These summaries are supported by icons or subtle graphics.
 - **Visual Appeal:** The design employs a balanced color scheme that reflects the platform’s healthcare focus. High-quality images and graphics are used to evoke a sense of trust and professionalism.
- **User Interaction and Experience:**
Users arriving at the Landing page can quickly understand what CureWise offers. Clear navigation and prominent CTAs facilitate a smooth transition to the next steps, ensuring that users are not overwhelmed and can easily move forward to registration or exploration.

6.2.2 About Us Page

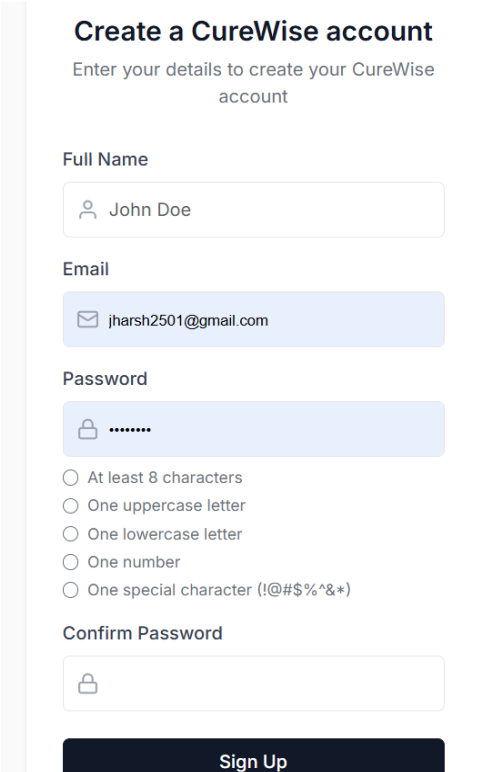
- **Purpose:**
The About Us page provides an in-depth introduction to CureWise. It explains the platform’s mission, vision, and technological foundation, thereby establishing credibility and trust among users.
- **Detailed Design Elements:**
 - **Hero Banner:** A visually appealing banner featuring the CureWise logo and the page title “About CureWise” sets a professional tone. This section may also include a brief tagline that summarizes the platform’s core value.
 - **Informative Content:** The main content is organized into well-defined sections that describe the platform’s origins, the problem it addresses, and its unique approach to personalized healthcare. Detailed narratives provide context about the team’s expertise, the technology stack, and the future vision.
 - **Supporting Imagery:** Carefully selected images or icons complement the text, reinforcing key messages such as innovation in healthcare and user-centric design.
 - **Navigation and Links:** Intuitive navigation elements allow users to quickly jump to related sections, such as team bios or the mission statement, offering a deeper dive into the platform’s background.

User Interaction and Experience:

The About Us page is designed for clarity and engagement. It invites users to learn more about

CureWise’s story, emphasizing transparency and reliability. The layout encourages readers to scroll through the detailed content, gaining a full understanding of the platform’s values and aspirations.

6.3 Login & Signup Pages



Create a CureWise account
Enter your details to create your CureWise account

Full Name

Email

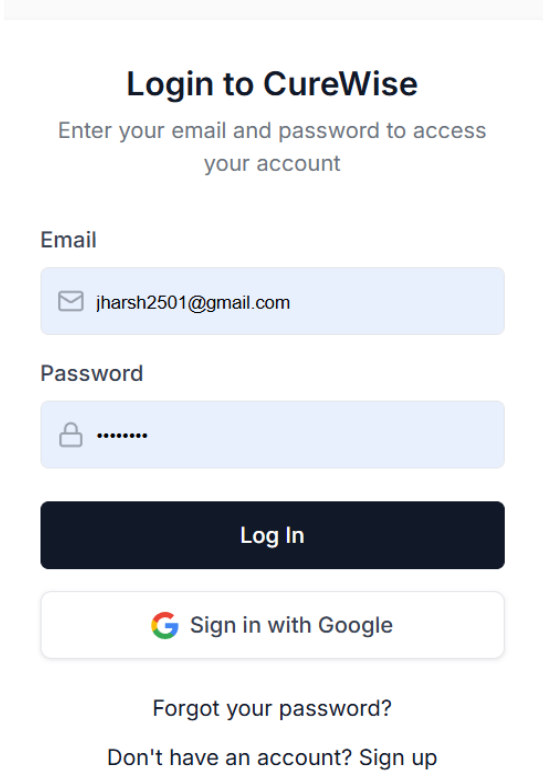
Password

☐ At least 8 characters
☐ One uppercase letter
☐ One lowercase letter
☐ One number
☐ One special character (!@#\$\$%^&*)

Confirm Password

Sign Up

Figure 6.2:




Login to CureWise
Enter your email and password to access your account

Email

Password

Log In

 Sign in with Google

Forgot your password?

Don't have an account? Sign up

6.3.1 Login Page

- **Purpose:**
Provides a secure access point for returning users. The Login page is essential for establishing a secure session, allowing users to access their personalized dashboard and other features.
- **Detailed Design Elements:**

- **Credential Input:** Prominent fields for entering email and password. The design ensures these fields are easily identifiable and accessible.
- **Support Features:** Includes a “Forgot Password” link, enabling users to recover their accounts easily. Optionally, there may be a Google Sign-In option to simplify authentication.
- **Feedback Mechanism:** Clear error messages and validation indicators are provided to guide users in correcting any mistakes, ensuring a secure and user-friendly login process.
- **User Interaction and Experience:**
Upon entering their credentials, users receive immediate feedback on the status of their login attempt. Successful authentication redirects them to the Dashboard, while unsuccessful attempts provide clear, actionable error messages.

6.3.2 Signup Page

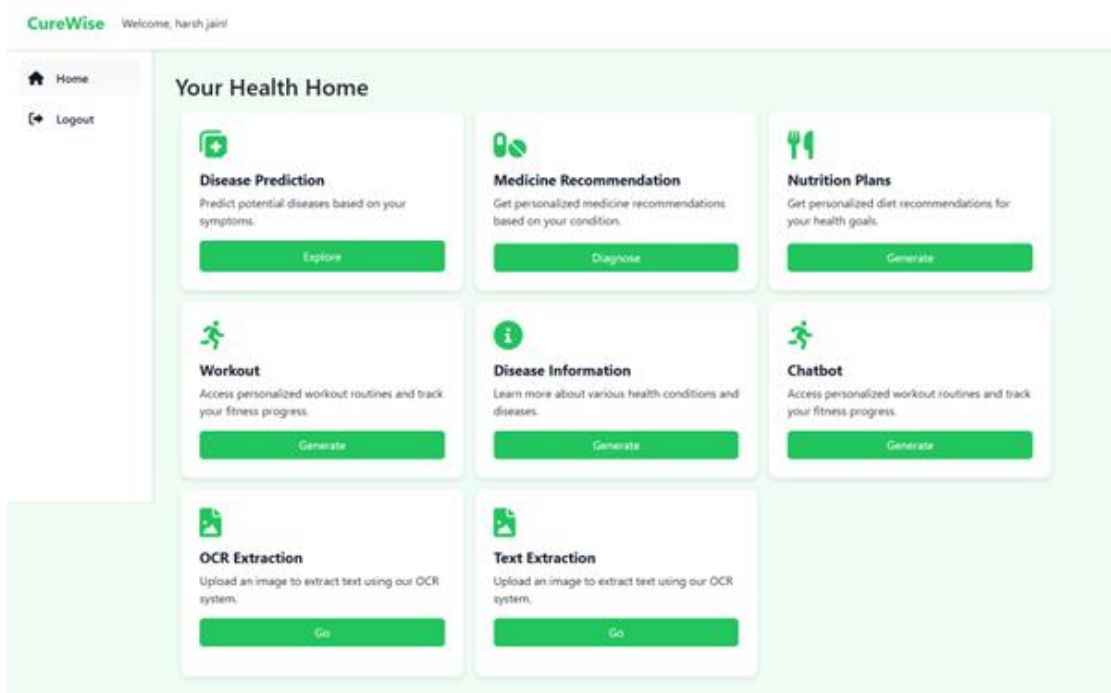
- **Purpose:**
Facilitates the registration of new users by collecting essential information and initiating the account creation process.
- **Detailed Design Elements:**
 - **Registration Form:** Consists of fields for Full Name, Email, Password, and Confirm Password. The layout emphasizes clarity, with labels and placeholder text to guide input.
 - **Security Guidance:** Provides real-time validation feedback (e.g., password strength indicators) and hints to help users create secure passwords.
 - **Submission and Confirmation:** A prominent “Sign Up” button triggers the registration process. Upon submission, the system generates a confirmation token and sends an email to the user to verify their account.
 - **Policy Information:** Footer links to the Terms of Service and Privacy Policy ensure transparency about user data handling.

User Interaction and Experience:

New users experience a straightforward registration process with real-time validation, clear instructions, and immediate feedback. The design reduces friction and builds trust by showing a professional, secure registration flow.

6.4 Dashboard

Figure 6.3:



6.4.1 Dashboard Overview

- **Purpose:**
Acts as the central hub for accessing all functionalities of the CureWise platform. It is the primary interface after user authentication.
- **Detailed Design Elements:**
 - **Feature Tiles:** Organized cards representing key functionalities such as Disease Prediction, Medicine Recommendations, Nutrition Plans, Workout Suggestions, Chatbot Access, and Medical Text Analysis.
 - **Personalized Greeting:** Displays the logged-in user's name and provides quick links to account settings.
 - **Navigation Menu:** A consistent navigation bar that allows users to quickly move between modules.
 - **Dynamic Data:** Real-time updates and summary information for each feature, providing users with a snapshot of their health data and recommendations.

User Interaction and Experience:

The Dashboard is designed for efficiency and clarity. Users can quickly scan the available features and select the module they wish to explore, all while receiving personalized insights and updates that enhance the overall experience.

6.5 Disease Prediction Page

Figure 6.4:

The screenshot shows the CureWise Disease Prediction interface. On the left, a sidebar lists various symptoms under categories like Inflammation, Scarring, and Whiteheads. The main area is titled 'Mild Symptoms' and contains three input fields for Symptom 1, Symptom 2, and Symptom 3. Symptom 1 is set to 'Pimples', Symptom 2 is set to 'Blackheads', and Symptom 3 is empty. A 'Submit' button is located below the input fields. At the bottom, a 'Disease Prediction' section displays the result 'acne' with a brief description: 'A skin condition characterized by pimples, blackheads, and cysts, often occurring in adolescence but can affect people of all ages.' Navigation links 'Back to Severity Selection' and 'Back to Home' are in the top right corner.

6.5.1 Disease Prediction Functionality

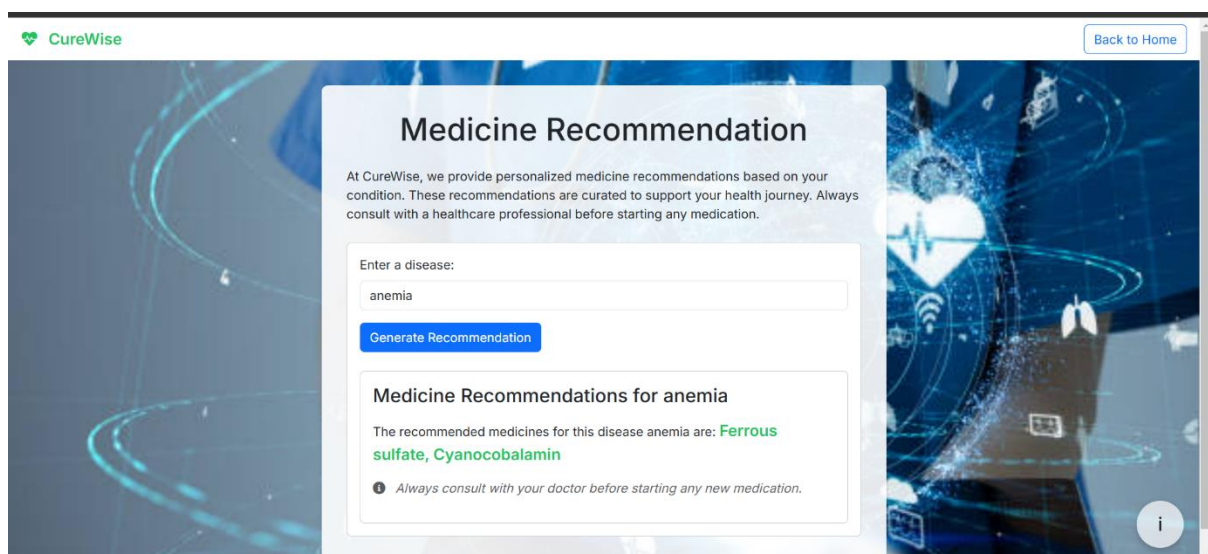
- **Purpose:**
Enables users to input their symptoms and receive predictions about potential diseases. The system processes the input through machine learning models to generate predictions.
- **Detailed Design Elements:**
 - **Symptom Input Form:** Multiple input fields or dropdowns allow users to select or type symptoms. A severity selector (Mild, Moderate, Severe) helps determine the appropriate model for prediction.
 - **Submission Process:** A clearly labeled “Submit” button triggers the model inference.
 - **Result Display:** Predicted disease information is presented in a structured format, including the disease name, a brief description, and recommended actions.

User Interaction and Experience:

Users input their symptoms and choose a severity level. After submission, the system processes the data and presents a prediction in an easily digestible format. This immediate feedback helps users understand their potential health issues quickly.

6.6 Medicine Recommendation Page

Figure 6.5:



6.6.1 Medicine Recommendation Functionality

- **Purpose:**

Provides personalized medicine recommendations based on the disease entered by the user.

- **Detailed Design Elements:**

- **Input Field:** A simple text box where users enter the disease for which they need medicine recommendations.
- **Generate Recommendation Button:** On clicking, the system queries the database or leverages an AI module to suggest relevant medications.
- **Result Panel:** Displays a list of recommended medicines along with brief descriptions or dosage guidelines, including any necessary disclaimers.

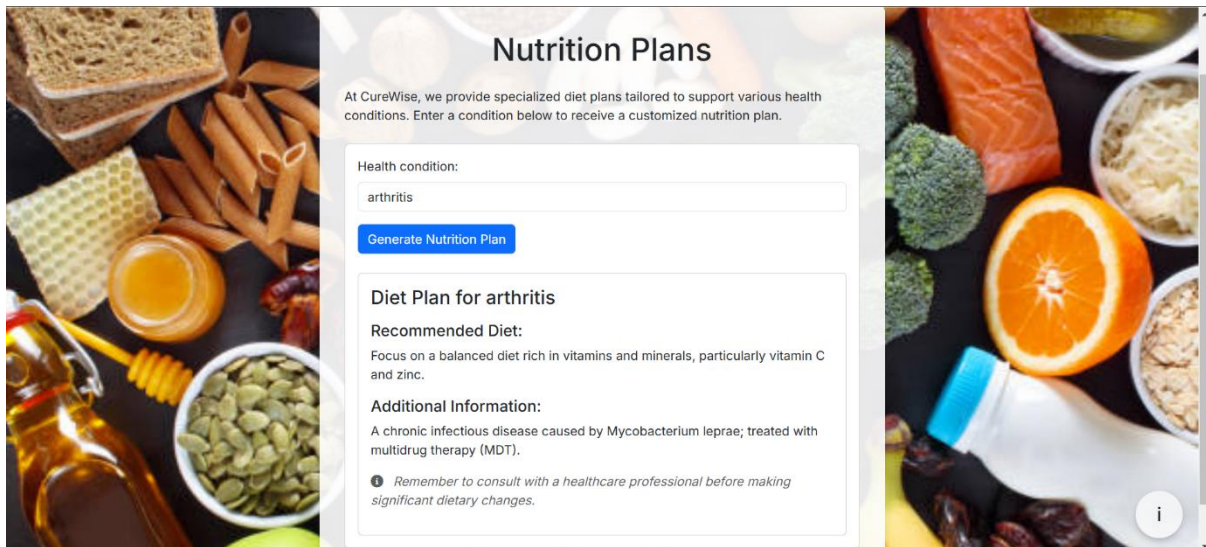
User Interaction and Experience:

Users quickly see tailored medicine suggestions after inputting the disease name. The interface is designed to clearly separate different pieces of information, making it easy for users to understand their

options.

6.7 Nutrition Plan Page

Figure 6.6:



6.7.1 Nutrition Recommendation Functionality

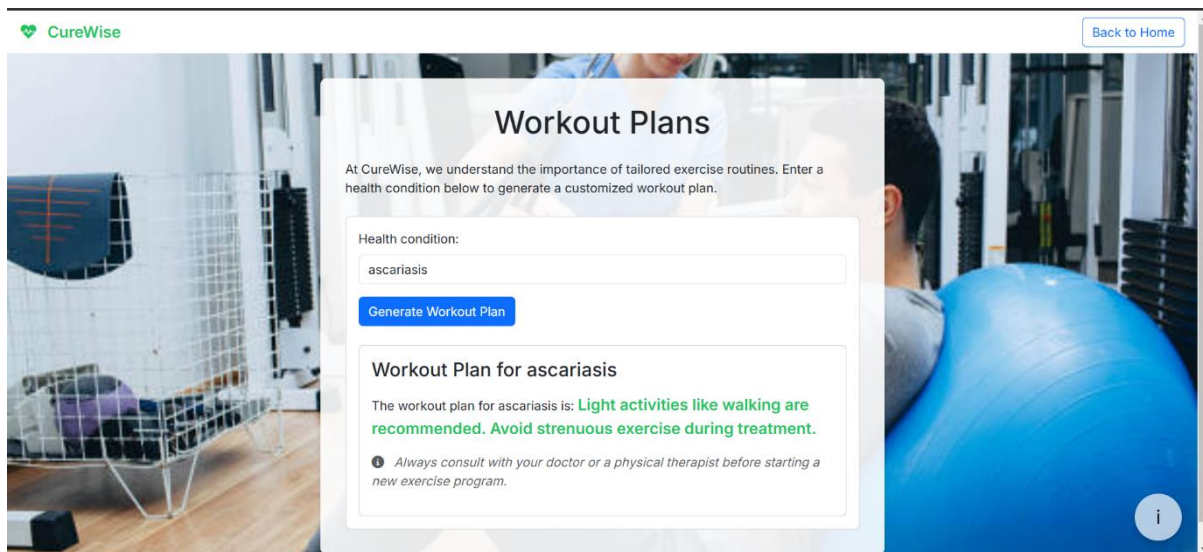
- **Purpose:**
Suggests personalized diet plans that address specific health conditions.
- **Detailed Design Elements:**
 - **Condition Input Field:** Users type in a health condition (e.g., “diabetes”).
 - **Generate Plan Button:** Triggers the system to compile a diet plan based on curated data and/or AI recommendations.
 - **Output Section:** Presents a structured diet plan that includes recommended foods, portion sizes, and additional dietary tips.

User Interaction and Experience:

Users are provided with actionable dietary advice immediately upon request. The design ensures that the recommendations are presented in a clear and concise manner, helping users make informed nutritional choices.

6.8 Workout Plan Page

Figure 6.7:



6.8.1 Workout Recommendation Functionality

- **Purpose:**

Offers personalized workout plans tailored to the user's health condition and fitness level.

- **Detailed Design Elements:**

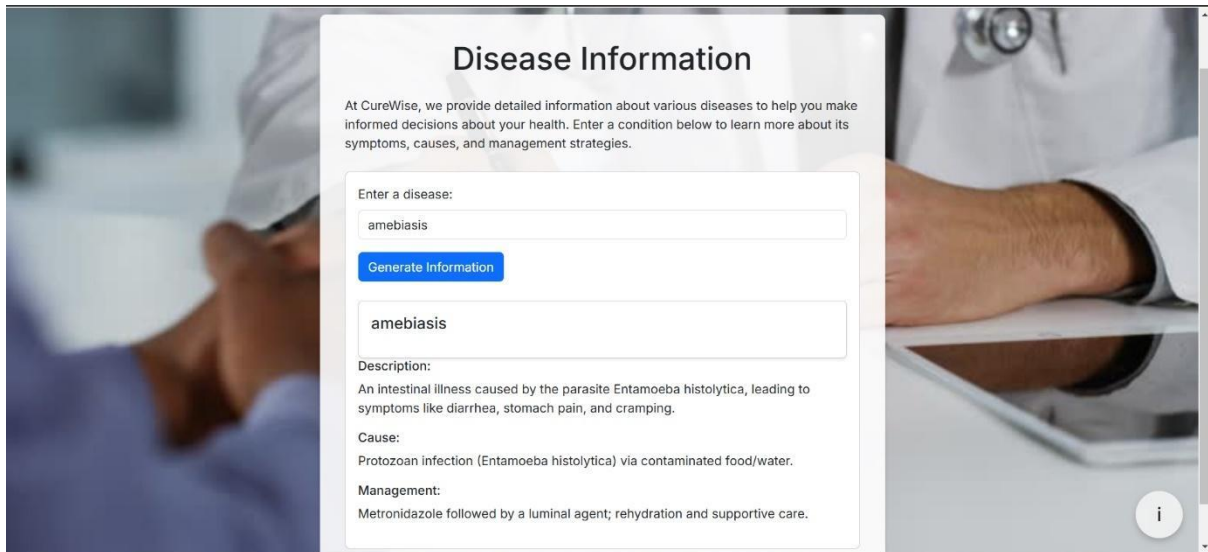
- **Condition/Preference Input:** A field or set of options allows users to specify a health condition or fitness goal.
- **Generate Workout Button:** On submission, the system generates a recommended exercise routine.
- **Plan Display:** Provides details on the recommended exercises, including duration, intensity, and frequency, along with any necessary safety warnings.

User Interaction and Experience:

Users receive a customized workout plan that they can follow, complete with clear instructions and visual cues. The interface supports easy navigation and quick adjustments if needed.

6.9 Disease Description Prediction Page

Figure 6.8:



6.9.1 Disease Information Functionality

- **Purpose:**

Allows users to search for a disease and view detailed information, including its description, causes, and management guidelines.

- **Detailed Design Elements:**

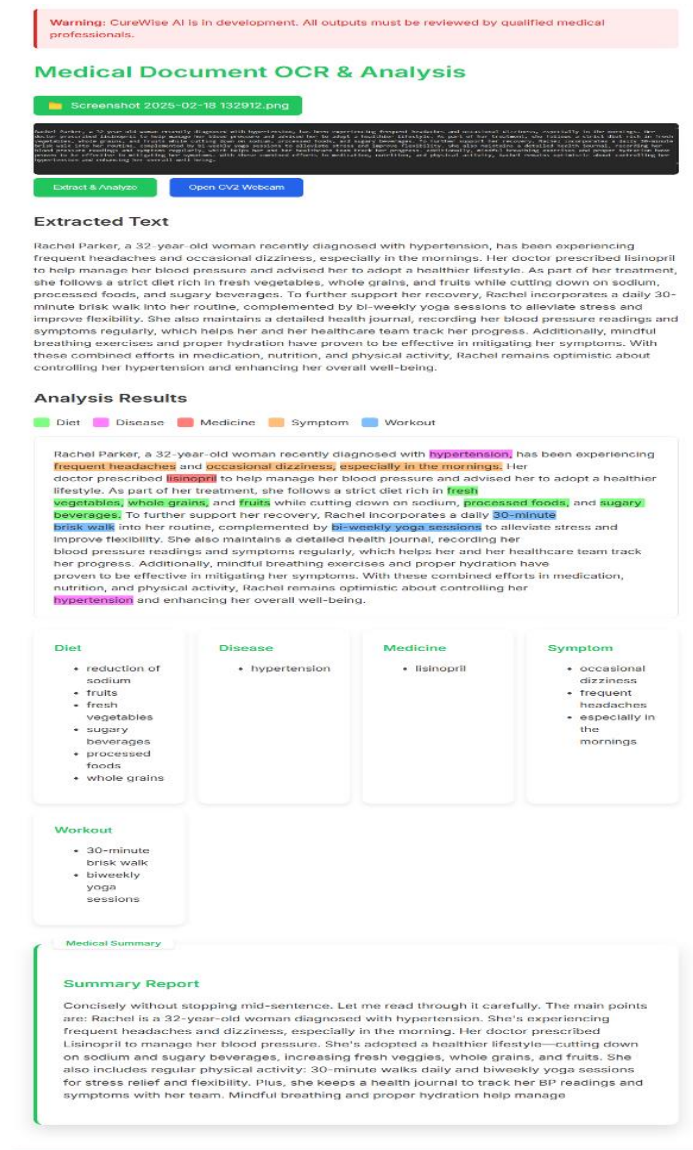
- **Search Field:** Users enter a disease name, which the system uses to query its unified dataset.
- **Information Retrieval:** Upon submission, the system returns a detailed report organized into sections (e.g., description, cause, management).
- **Structured Layout:** The information is displayed in a clear, segmented format for easy reading.

User Interaction and Experience:

The search process is straightforward, and the results are presented in an easy-to-read manner. This enables users to quickly gain insights into the disease and understand potential treatment options.

6.10 Additional Interfaces

6.10.1 OCR & Medical Document Analysis



Purpose:

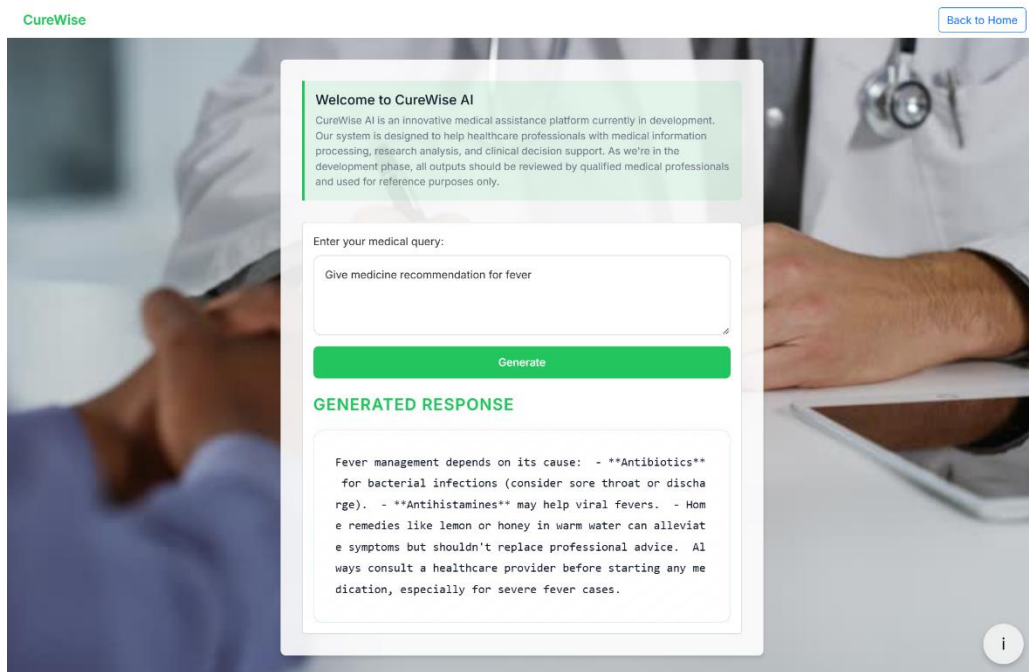
Converts uploaded documents (images, PDFs, etc.) into editable text and analyzes the content to highlight key medical terms.

• Detailed Design Elements:

- **File Upload Section:** Users can upload documents via drag-and-drop or file selection.
- **Text Extraction Display:** The raw text extracted via OCR is shown in a dedicated area.

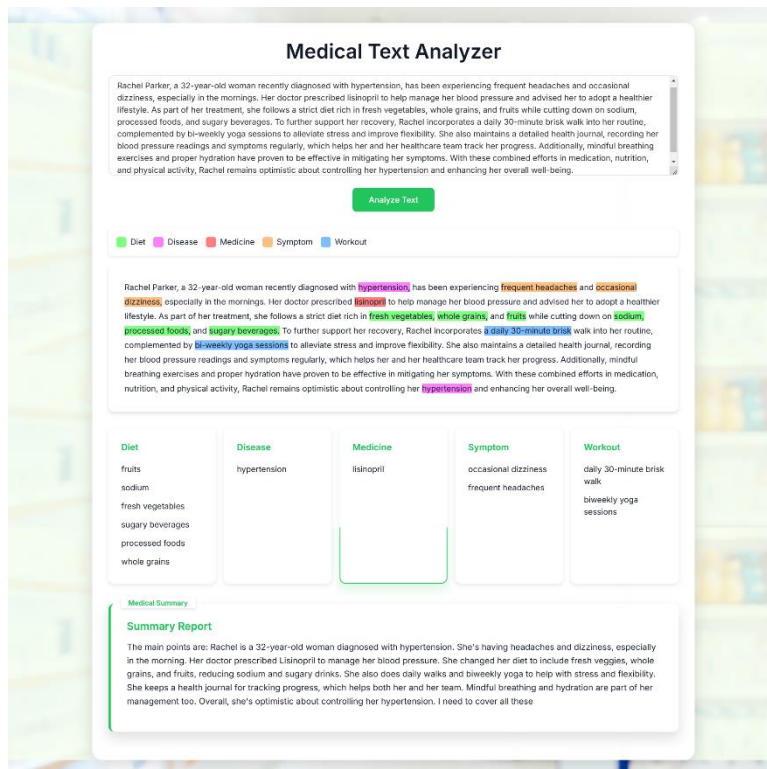
- **Highlighting and Summarization:** The extracted text is processed to highlight medical terms using color-coding, and a concise summary is generated.
- **User Interaction and Experience:**
Users see immediate results after uploading a document. The interface supports interactive review of the highlighted terms and provides a summary for quick comprehension.

6.10.2 Chatbot Interface



- **Purpose:**
Facilitates real-time communication with an AI-driven assistant that answers medical queries and offers recommendations.
- **Detailed Design Elements:**
 - **Chat Window:** A scrollable area displays the conversation history.
 - **Input Mechanism:** A simple input field is provided for users to type their queries.
 - **Response Display:** Instant AI-generated responses appear in the conversation area, maintaining a friendly, informative tone.
- **User Interaction and Experience:**
Users experience a natural, conversational flow while interacting with the chatbot. The interface is designed for clarity, ensuring that both queries and responses are easy to read and understand.

6.10.3 Medical Text Analyzer



- **Purpose:**
Processes and analyzes medical text to extract key terms, correct spelling errors, and generate a summary.
- **Detailed Design Elements:**
 - **Text Input Area:** Users can paste text or upload documents.
 - **Processing Panel:** The system highlights terms by category (e.g., diseases, medicines) using distinct colors.
 - **Summary Output:** A concise summary is displayed below the highlighted text.

User Interaction and Experience:

The interface provides immediate visual feedback on the text analysis, allowing users to quickly identify important information and gain an overview through the summary.

Chapter 7

Conclusion

7.1 Conclusion:

In summary, the implementation of the CureWise Medicine Recommendation System represents a significant advancement in personalized healthcare delivery. By leveraging advanced machine learning techniques alongside Optical Character Recognition (OCR) technology, the system analyzes patient symptoms, medical documents, and prescription details to provide tailored recommendations. This approach enables timely, relevant health advice—including medication suggestions, diet plans, exercise routines, and precautionary measures—while safeguarding user privacy and minimizing dependency on extensive personal data. Ultimately, CureWise not only enhances user engagement and convenience but also empowers individuals to make more informed decisions about their health, serving as a valuable supplementary tool to professional medical consultation.

7.1.1 Significance of the system:

1. **Personalized Healthcare Delivery:**

By leveraging both patient-reported symptoms and OCR-extracted data from handwritten prescriptions, CureWise is capable of constructing a comprehensive health profile for each user. The system's ability to process diverse data sources ensures that recommendations are not only personalized but also dynamically adjusted based on new inputs. This results in a more nuanced understanding of an individual's health needs and a tailored set of actionable suggestions.

2. **Enhanced Diagnostic Support:**

CureWise functions as an immediate first point of contact in healthcare by providing rapid, preliminary diagnostic insights. It reduces the waiting time for initial assessments and helps users identify potentially serious conditions early on. This can lead to quicker medical interventions and ultimately improve patient outcomes.

3. **User-Centric and Intuitive Interface:**

The system is designed with a focus on usability, incorporating features such as real-time symptom analysis, user-friendly dashboards, and seamless integration of

multimedia inputs. Patients can easily upload images of their prescriptions, input detailed symptom descriptions, and receive comprehensive recommendations—all in a few simple steps. This intuitive design not only increases user engagement but also enhances overall satisfaction with the healthcare process.

4. **Privacy and Data Security:**

A core principle behind CureWise is the minimization of data collection. The system is engineered to use only essential patient information, thereby reducing the risks associated with data breaches. Secure encryption protocols and minimal data retention policies ensure that sensitive health data is handled with the highest level of security and transparency.

5. **Holistic Health Management:**

Unlike traditional systems that might focus solely on medication, CureWise adopts a holistic approach. It offers integrated health advice by coupling medication recommendations with diet, exercise, and lifestyle modifications. This comprehensive strategy is crucial in managing chronic conditions and promoting overall well-being, making the system a one-stop solution for personalized healthcare management.

7.2 Limitation of the system:

Despite its many strengths, CureWise also faces several limitations that need to be addressed in future iterations:

1. **Scope of Data and Diagnostic Nuance:**

The system's diagnostic accuracy is closely tied to the quality and comprehensiveness of the input data. In instances where patients provide incomplete or ambiguous symptom descriptions, the recommendations may not fully capture the complexity of their conditions. Additionally, while OCR technology is a powerful tool, it may struggle with illegible handwriting or poor-quality document scans, potentially leading to incomplete data extraction.

2. **Over-Reliance on Automated Assessments:**

Although CureWise is designed to empower users with immediate insights, there is an inherent risk that users might rely too heavily on the system's automated recommendations. Without professional oversight, patients could misinterpret the advice provided, possibly delaying critical medical consultations. It is essential to emphasize that the system is intended as an adjunct to, not a replacement for, professional medical evaluation.

3. **Limited Contextual Awareness:**

Current implementations of CureWise primarily focus on static inputs such as symptoms and medical documents. They do not yet fully integrate dynamic contextual factors such as real-time biometric data, environmental conditions, or stress levels—all of which can play a crucial role in accurate health assessment and recommendation.

4. **Integration Challenges with Broader Health Systems:**

While CureWise can serve as a valuable standalone tool, its effectiveness would be greatly enhanced by integration with existing electronic health records (EHR) and telemedicine platforms. The absence of such integrations limits the system's ability to provide a more complete and continuous picture of a patient's health history, potentially affecting the long-term accuracy and relevance of its recommendations.

7.3 Future scope of the project:

Looking ahead, there are numerous avenues for enhancing and expanding the capabilities of CureWise:

1. **Advanced Personalization Through Deep Data Integration:**

Future versions of CureWise could incorporate additional layers of patient data, such as genetic information, continuous biometric monitoring from wearable devices, and historical health records. This would enable the system to deliver even more personalized and precise recommendations, reflecting a broader spectrum of an individual's health profile.

2. **Multi-Modal Data Fusion:**

Expanding the system's ability to integrate data from multiple modalities—including audio, video, and sensor data—can provide a more holistic view of patient health. For example, combining visual analysis of medical imaging with textual data from prescriptions could improve diagnostic precision and offer richer, more detailed health insights.

3. **Context-Aware and Adaptive Recommendations:**

By incorporating contextual information such as geographic location, current weather conditions, and even patient activity levels, future iterations of CureWise could tailor recommendations to the immediate environment and lifestyle of the user. This adaptive approach would ensure that the health advice is both timely and contextually relevant.

4. **Enhanced Feedback and Interactivity:**

Building interactive feedback loops into the system will allow users to provide real-time responses to the recommendations they receive. This continuous feedback can be used to refine

the machine learning models, making the system more accurate and responsive over time. Additionally, interactive features like chatbots or virtual health assistants could offer immediate clarification and further support for users.

5. **Seamless Integration with Healthcare Infrastructure:**

Future enhancements should focus on creating interoperability with existing healthcare systems such as EHRs, pharmacy databases, and telemedicine platforms. Such integration would facilitate a more comprehensive view of a patient's health history, enabling better-coordinated care and more informed decision-making by healthcare providers.

6. **Ethical and Regulatory Considerations:**

As the system evolves, it will be critical to continuously update its data governance and ethical frameworks to comply with evolving healthcare regulations. This includes ensuring transparency in how recommendations are generated, safeguarding patient privacy, and maintaining robust data security protocols. Future developments should also focus on minimizing algorithmic biases and ensuring that the system's advice is equitable and accessible to all users.

7. **Research and Development of New Machine Learning Models:**

Ongoing research into more advanced natural language processing, computer vision, and deep learning techniques will further enhance the system's ability to interpret complex medical data. By staying at the forefront of AI research, CureWise can continuously improve its diagnostic accuracy and recommendation quality.

8. **Expansion into Preventive Health and Wellness:**

Beyond immediate diagnostic and treatment recommendations, future versions of CureWise could emphasize preventive healthcare. By analyzing trends in user data over time, the system could provide early warnings and personalized advice to help users avoid the onset of chronic diseases. This proactive approach would further empower individuals to manage their health proactively, potentially reducing the overall burden on healthcare systems.

Overall, the future scope of personalized medicine recommendation systems like CureWise is expansive, with opportunities for innovation in personalization, multi-modal analysis, contextualization, interactivity, cross-platform integration, and ethical considerations. By embracing these avenues for growth, such systems can continue to evolve and enhance the user experience, ultimately shaping the way we access, understand, and manage our health in the digital age.

References

Flask Documentation. Available at: <https://flask.palletsprojects.com/>

MySQL Documentation. Available at: <https://dev.mysql.com/doc/>

Jupyter Notebook. Available at: <https://jupyter.org/>

Agile Alliance – Agile 101. Available at: <https://www.agilealliance.org/agile101/>

Pandas Documentation. Available at: <https://pandas.pydata.org/docs/>

NumPy Documentation. Available at: <https://numpy.org/doc/>

Scikit-learn Documentation. Available at: <https://scikit-learn.org/stable/>

Python Official Website. Available at: <https://www.python.org/>

Deep-seek Chatbot Documentation. Available at: <https://deepseek.ai/>

Requests Library Documentation. Available at: <https://docs.python-requests.org/en/latest/>