**SGP REPORT**

*On*

# PredictiX - A Multi Disease Predictor

*Submitted by*

**Dhrudeepsinh Jadeja (IU2341221251)**

**Vishvjitsinh Chauhan (IU2241220117)**

**Roshni Panchal (IU2241220096)**

**Priya Viroja (IU2341221246)**

**Maher Savaj (IU2341221248)**

*In partial fulfillment for the award of the degree*

*Of*

*BACHELOR OF TECHNOLOGY*

*In*

*Information Technology*


**INDUS INSTITUTE OF TECHNOLOGY AND ENGINEERING**

**INDUS UNIVERSITY CAMPUS, RANCHARDA, VIA-THALTEJ**

**AHMEDABAD-382115, GUJARAT, INDIA,**

WEB: www.indusuni.ac.in

OCTOBER 2025-26

**SGP REPORT**

ON

# PredictiX - A Multi Disease Predictor

AT


*In the partial fulfillment of the requirement*
*for the degree of*
*Bachelor of Technology*
*in*
*Information Technology*

**PREPARED BY**

Dhrudeepsinh Jadeja (IU2341221251)

Vishvjitsinh chauhan (IU2241220117)

Roshni Panchal (IU2241220096)

Priya Viroja (IU2341221246)

Maher Savaj (IU2341221248)

**UNDER GUIDANCE OF**

**Internal Guide**
Dr. Shruti Yagnik
Head of the Department
Department of Information Technology,
IITE. Indus University,
Ahmedabad

**SUBMITTED TO**

INDUS INSTITUTE OF TECHNOLOGY AND ENGINEERING
INDUS UNIVERSITY CAMPUS, RANCHARDA, VIA-THALTEJ
AHMEDABAD-382115, GUJARAT, INDIA,
WEB: www.indusuni.ac.in

OCTOBER 2025-26

# CANDIDATE'S DECLARATION

We declare that final semester report entitled "**PredictiX - A Multi Disease Predictor**" is our own work conducted under the supervision of the guide **Dr**. **Shruti Yagnik.**

We further declare that to the best of our knowledge, the report for B. Tech 7th semester does not contain part of the work which has been submitted for the award of B. Tech Degree either in this university or any other university without proper citation.

Candidate's Signatures
Dhrudeepsinh Jadeja (IU2341221251)

Candidate's Signatures
Vishvjitsinh Chauhan (IU2241220117)

Candidate's Signatures
Priya Viroja (IU2341221246)

Candidate's Signatures
Roshni Panchal (IU2241220096)

Candidate's Signatures
Maher Savaj (IU2341221248)

Guide: Dr. Shruti Yagnik

Head of the Department
Department of Information Technology,
Indus Institute of Technology and Engineering
INDUS UNIVERSITY– Ahmedabad,
State: Gujarat

**INDUS INSTITUTE OF TECHNOLOGY AND ENGINEERING**
**INFORMATION TECHNOLOGY**
**2025 -2026**


# CERTIFICATE

**Date:27/10/2025**

This is to certify that the project work entitled "**PredictiX - A Multi Disease Predictor"** has been carried out by **Our team** under my guidance in partial fulfillment of degree of Bachelor of Technology in **Information Technology (Final Year)** of Indus University, Ahmedabad during the academic year 2025 – 2026.

| | |
|---|---|
| Dr. Shruti Yagnik<br>Head of the Department,<br>Department of Information Technology,<br>I.I.T.E. , Indus University<br>Ahmedabad | Dr. Shruti Yagnik<br>Head of the Department,<br>Department of Information Technology,<br>I.I.T.E. , Indus University<br>Ahmedabad |

## ACKNOWLEDGEMENT

We would like to express my sincere gratitude to all those who have contributed to the successful completion of our Bachelor of Technology Project.

Firstly, We would like to thank our project guide, **Dr. Shruti Yagnik** , for her guidance and support throughout the project. Her valuable insights and suggestions have been instrumental in shaping our work.

We are also grateful to the faculty members of the department of information technology engineering for their support and encouragement. Their feedback and suggestions have been invaluable in improving the quality of our work.

We would like to express our gratitude to our family and friends for their unwavering support and encouragement throughout the project. Their constant motivation has been a source of inspiration for us.

Lastly, We would like to thank the university for providing us with the necessary resources and facilities to complete our project successfully.

Thank you all for your support and guidance.

Dhrudeepsinh Jadeja (IU2341221251)
Vishvjitsinh Chauhan (IU2241220117)
Roshni Panchal (IU2241220096)
Priya Viroja (IU2341221246)
Maher Savaj (IU2341221248)

Department of Information Technology

# TABLE OF CONTENT

# ABSTRACT

*PredictiX - A Multi Disease Predictor is an online health prediction system designed to help users assess the early detection of four serious medical conditions: Breast Cancer, Lung Cancer, Diabetes, and Heart Disease. The system allows individuals to either enter their health information manually or upload a structured medical report. Based on the input, it provides an immediate risk of suffering from the mentioned disease.*

*The purpose of PredictiX is to support early awareness and encourage timely medical consultation. It simplifies complex medical evaluations into easy-to-understand results, making it accessible for both healthcare professionals and the general public. The system is designed to be user-friendly, fast, and informative, helping bridge the gap between initial health concerns and professional diagnosis.*

*By focusing on accuracy, ease of use, and accessibility, PredictiX aims to promote preventive healthcare and empower users to take control of their health in a convenient and reliable way..*

# LIST OF FIGURES

# LIST OF TABLES



# ABBREVIATION

Abbreviations used throughout this whole document for Survey Application are:

| | |
|---|---|
| HTML | Hypertext Markup Language |
| HOD | Head of the Department |
| UML | Unified Modeling Language |
| CSS | Cascading Style Sheet |
| DBMS | Database management |
| MongoDB | Document Oriented Database |
| Tailwind CSS | Utility-first CSS Framework |
| React.js | Javascript Library |
| Node.js | Javascript run-time environment |
| Mongoose | Object Data Modeling (ODM) library for MongoDB and JavaScript |

# CHAPTER 1
# INTRODUCTION

- **PROJECT SUMMARY**
- **PROJECT PURPOSE**
- **PROJECT SCOPE**
- **OBJECTIVES**
- **TECHNOLOGY AND TOOLS**
- **SYNOPSIS**

## 1.1 PROJECT SUMMARY

With the rising number of people affected by chronic and life-threatening diseases, early diagnosis has become an important step in improving treatment outcomes and reducing health risks. Many health conditions, such as breast cancer, lung cancer, diabetes, and heart disease, can be better managed or even prevented when they are identified at an early stage. However, people often miss early signs due to lack of awareness or limited access to regular medical checkups. To support early detection in a simple and accessible way, **PredictiX** was developed as a health prediction system.

PredictiX focuses on four major health conditions that are common and have a significant impact on public health. The system helps users check for possible signs of breast cancer, lung cancer, diabetes, or heart disease based on the information they provide. It is designed to give quick results and serve as an initial step that encourages users to follow up with medical professionals if any risk is found.

The system provides two input options for users. One option is to manually fill in their health-related details, such as basic information and symptoms. The other option is to upload a structured medical report that contains relevant data. Once the information is submitted, the system gives a simple output indicating whether there is a possible risk of the disease. This helps users get a basic understanding of their health status without needing advanced medical knowledge.

PredictiX is built to be user-friendly and easy to navigate. It does not give a final diagnosis but is meant to assist with early detection and raise awareness. It encourages people to take the next step by consulting a doctor or going for further medical tests if needed. The goal is to help users become more informed about their health and take timely action when necessary.

In summary, PredictiX is a supportive health prediction tool that focuses on four key diseases. It allows users to input their information, get a quick result, and use that result as a starting point for further medical consultation. The system is meant to complement medical checkups and support early awareness, making health monitoring more accessible.

## 1.2 PROJECT PURPOSE

The purpose of the PredictiX project is to develop an intelligent and accessible health prediction system that enables early detection of critical diseases such as Kidney disease, Brain tumor, Pneumonia, and heart disease. With the rising prevalence of these conditions and the importance of timely intervention, the project aims to provide individuals with a simple yet reliable platform to assess their health risks.

PredictiX is designed to bridge the gap between advanced medical diagnostics and general awareness by offering quick, user-friendly, and interpretable results. It empowers users to make informed decisions about seeking professional medical advice, thereby promoting preventive healthcare and reducing the chances of delayed diagnosis.

By leveraging machine learning and deep learning techniques, the project not only supports individuals in taking proactive steps toward their health but also assists healthcare professionals with a preliminary decision-support tool. Ultimately, the purpose of PredictiX is to simplify health monitoring, encourage early medical consultation, and contribute toward building a more aware and health-conscious society.

## 1.3 PROJECT SCOPE

The PredictiX project aims to provide a powerful, intelligent, and user-friendly platform that leverages machine learning and deep learning to predict four major health conditions: Kidney disease, Brain tumor, Pneumonia, and heart disease. Its scope includes data preprocessing, model development, web application integration, and user accessibility, while also considering current limitations and future enhancements.

**1. Disease Coverage**

PredictiX supports prediction of:

- **Pneumonia** using CNNs trained on histopathological image data.
- **Brain tumor** using ResNet trained on CT scan images.
- **Kidney disease** using SVM applied to structured patient data.
- **Heart Disease** using a tuned Decision Tree model with clinical parameters.

**2. Input Mechanisms**

- **Manual Entry**: Users can input numerical values for features such as glucose level, age, cholesterol, etc.
- **Report Upload**: Users can upload structured health reports (CSV/txt) following a specific format.
- **Planned Feature**: OCR integration to automatically read values from scanned or printed medical reports in PDF/image format.

**3. Target Users**

PredictiX is designed with a wide range of potential users in mind:

- **General Public / Patients**: Individuals seeking early awareness or risk assessment for chronic diseases based on their health metrics. Especially helpful in areas with limited access to advanced diagnostic facilities.
- **Healthcare Professionals**: Doctors and medical practitioners can use it as a **pre-screening tool** to support decision-making and prioritize diagnostic tests.

- **Medical Students and Researchers**: As a learning and experimentation platform to understand how AI/ML can be applied in healthcare.
- **Hospitals and Clinics**: Integration into existing health management systems for streamlined prediction and record keeping.

The interface and predictive reports are designed to be easy to understand, enabling non-technical users to make sense of their results while still offering valuable insights to professionals.

**4. System Architecture and Technologies**

- **Frontend**: React.js
- **Backend**: Node.js with Express
- **Database**: MongoDB for user and system data
- **ML Models**: Developed using Python, saved as .pkl files, and integrated using Node.js threads
- **Image-based models (CNN, ResNet)** require GPU but are currently not deployed due to resource constraints

**5. Performance & Accuracy**

- **Pneumonia** : ~89%
- **Brain tumor** : ~83%
- **Kidney disease** : ~85%
- **Heart Disease** : ~87%

All models are trained from scratch and optimized through feature selection and hyperparameter tuning.

**6. Deployment & Accessibility**

- Web-based system for easy accessibility via browser.
- No installation required.
- User-friendly forms and interactive interfaces.
- Currently hosted locally with potential for cloud deployment in the future.

**7. Limitations**

- Deployment of image-based models is not feasible in real-time due to GPU limitations.
- Fixed report format needed for file uploads.
- No multilingual support currently.
- Not a replacement for actual medical diagnosis—meant to be a decision-support tool.

**8. Future Scope**

- Integration of **OCR** for unstructured reports.
- Adding more disease predictors.
- Adding user accounts and prediction history tracking.

## 1.4 OBJECTIVES

The primary objective of PredictiX is to develop a comprehensive and intelligent healthcare prediction system capable of early detection and diagnosis of critical medical conditions, including, Kidney disease, Brain tumor, Pneumonia, and heart disease. By leveraging machine learning and deep learning algorithms, the project aims to provide accurate, fast, and accessible disease prediction to support both patients and healthcare professionals in making informed decisions.

PredictiX is designed to:

- Help identify signs of serious health conditions at an early stage.
- Make health checks more accessible through a simple and easy-to-use system.
- Allow users to enter their health information or upload reports for quick analysis.
- Provide fast and clear results without needing heavy resources.
- Support early health awareness in areas with limited medical facilities.

By achieving these objectives, PredictiX strives to bridge the gap between cutting-edge AI technologies and practical, impactful healthcare applications, ultimately promoting preventive healthcare and early diagnosis.

## 1.5 TECHNOLOGY AND TOOLS

### 1.5.1 Front-End Technology

- React.js
  React.js is used for building the web-based user interface. It provides responsive and interactive forms for manual health data entry as well as report uploads. The clean and modular design ensures that even non-technical users can easily navigate the system. It communicates with the backend through REST APIs using HTTP/HTTPS protocols.

### 1.5.2 Back-End Technology

- Node.js with Express.js
  The backend server is developed using Node.js along with the Express.js framework. This handles API requests, processes data from the frontend, integrates machine learning (ML) and deep learning (DL) models, and returns prediction results. It is also responsible for security, validation, and scalability of the system.

### 1.5.3 Database

- MongoDB
  A NoSQL database is used to store user inputs, structured reports, prediction logs, and system metadata. MongoDB was chosen for its flexibility, scalability, and ability to handle JSON-like data, which makes it suitable for web-based healthcare applications.

1.5.4 Programming Languages

- JavaScript – For both frontend (React.js) and backend (Node.js) development.
- Python – For developing, training, and serializing ML/DL models (SVM, Decision Tree, CNN, ResNet).

1.5.5 Frameworks and Libraries

- scikit-learn – For traditional ML models like SVM and Decision Tree.
- TensorFlow and Keras – For building and training deep learning models (CNN and ResNet).
- Mongoose ODM – For connecting Node.js backend with MongoDB.

1.5.6. Other Tools

- Pickle (.pkl) and H5 (.h5) – For model serialization and integration into the backend.
- Postman – For API testing during development.
- Git/GitHub – For version control and collaborative source code management.

1.5.7. Development Environment

- Operating Systems: Windows 10 / Ubuntu Linux (used during development).
- IDE/Code Editor: Visual Studio Code with extensions for JavaScript and Python.
- Python Environment: Anaconda or Python 3.x with required libraries.
- Browser: Google Chrome for frontend testing and debugging.

## 1.6 SYNOPSIS

PredictiX is a web-based health prediction system designed to assist in the early detection of four major diseases: **Kidney disease, Brain tumor, Pneumonia, and eart disease**. Early identification of these conditions can improve treatment outcomes and reduce risks, yet many people miss warning signs due to lack of awareness or access to regular medical checkups.

The purpose of this project is to provide an **intelligent, accessible, and user-friendly platform** that simplifies health risk assessment. PredictiX enables users to either manually enter their health data or upload structured medical reports, which are then analyzed using **machine learning (ML) and deep learning (DL) models**. The system provides quick, accurate, and interpretable results, encouraging timely medical consultation.

The objectives of the project include:

- Early detection of serious diseases through AI-based prediction.
- Providing a simple web interface for general users and healthcare professionals.
- Supporting multiple input mechanisms (manual entry and report uploads).
- Promoting preventive healthcare and awareness.

Technologies used include the **MERN stack** (MongoDB, Express.js, React.js, Node.js) for system development, and Python with libraries such as **TensorFlow, Keras, and Scikit-learn** for building predictive models. Models like **CNN, ResNet, SVM, and Decision Tree** were trained and optimized for accuracy.

PredictiX is not intended as a replacement for medical diagnosis but as a **decision-support tool** that bridges the gap between initial health concerns and professional consultation. By combining accessibility, speed, and accuracy, the project contributes to making preventive healthcare more practical and empowering individuals to take proactive steps toward their well-being.

# CHAPTER 2
# LITERATURE SURVEY

- **INTRODUCTION OF SURVEY**
- **WHY SURVEY ?**

## 2.1 INTRODUCTION OF SURVEY

The development of PredictiX - A Multi Disease Predictor was preceded by an extensive literature survey to understand the current state of healthcare prediction systems, machine learning applications in medical diagnostics, and existing disease detection methodologies. This survey examined academic research papers, published studies, and existing healthcare platforms to identify gaps in current systems and establish a foundation for our project's unique contributions.

The literature review focused on four primary areas corresponding to the diseases addressed by PredictiX: breast cancer detection using deep learning, lung cancer prediction through image analysis, diabetes prediction using clinical parameters, and heart disease assessment through machine learning algorithms. Additionally, we examined web-based healthcare systems, user interface design for medical applications, and the integration of ML/DL models into production environments.

## 2.2 WHY SURVEY ?

The comprehensive literature survey served multiple critical purposes in the development of PredictiX:

**1. Validation of Approach**

The survey validated our choice of machine learning and deep learning algorithms by examining peer-reviewed research demonstrating their effectiveness in medical prediction tasks. Understanding which algorithms performed best for specific disease types enabled informed decision-making during model selection.

**2. Learning from Established Methodologies**

Examining existing research allowed us to adopt proven techniques while avoiding approaches that had demonstrated limitations. For example, the survey revealed that ResNet's skip connections addressed gradient vanishing problems that plagued earlier deep network architectures, directly influencing our lung cancer prediction model design.

**3. Understanding Dataset Characteristics**

The survey provided insights into appropriate datasets for training each disease prediction model. Understanding dataset properties—such as class balance, feature distributions, and data quality issues—enabled better preprocessing strategies and realistic accuracy expectations.

**4. Identifying Technical Challenges**

Literature review highlighted common technical challenges in deploying ML/DL models for healthcare applications:

- Model serialization and integration with web backends
- Real-time prediction performance optimization
- Handling missing or incomplete medical data

- Managing model updates and versioning
- Ensuring security and privacy of health information

Understanding these challenges in advance allowed us to plan mitigation strategies during the design phase.

**5. Establishing Performance Benchmarks**

Published research provided performance benchmarks against which PredictiX could be evaluated. Understanding that Pneumonia classification typically achieves 85-92% accuracy on BreakHis dataset, or that Kidney Disease prediction on Pima dataset generally ranges from 75-85% accuracy, helped establish realistic goals for our models.

**6. Recognizing System Limitations**

The survey revealed inherent limitations in current prediction systems:

- Deep learning models require substantial computational resources for training and inference
- Image-based predictions are sensitive to image quality and preprocessing
- All prediction systems require careful validation and should complement rather than replace professional medical diagnosis
- Model performance may vary across different population demographics

Acknowledging these limitations informed our system design decisions and helpe establish appropriate disclaimers and user guidance

.

**7. Informing User Interface Design**

Research on existing healthcare platforms revealed critical UX principles:

- Medical terminology should be explained in accessible language
- Prediction results should include confidence levels and interpretation guidance
- The system should clearly communicate that it provides risk assessment, not definitive diagnosis
- Multiple input options accommodate users with varying technical proficiency

**8. Guiding Feature Engineering**

Literature examining successful medical prediction systems revealed important feature engineering insights:

- Certain medical parameters are more predictive than others for specific diseases
- Feature scaling and normalization significantly impact model performance
- Handling missing values requires domain-appropriate strategies
- Some features exhibit non-linear relationships with disease outcomes

**9. Understanding Deployment Constraints**

The survey revealed practical constraints of deploying healthcare prediction systems:

- GPU requirements for real-time image analysis
- Network bandwidth considerations for image upload
- Cross-browser compatibility for web applications
- Mobile responsiveness for accessibility

**10. Identifying Future Enhancement Opportunities**

Examining cutting-edge research identified potential future enhancements for PredictiX:

- OCR integration for unstructured medical reports
- Multi-modal fusion combining image and clinical data
- Explainable AI techniques for prediction interpretation
- Integration with electronic health record systems
- Continuous learning from new data

**11. Ensuring Ethical Considerations**

Literature on medical AI applications highlighted important ethical considerations:

- Informed consent for data usage
- Transparency about model limitations and accuracy
- Avoiding over-reliance on automated predictions
- Ensuring equitable performance across diverse populations
- Privacy protection for sensitive health information

**12. Building on Academic Foundation**

The survey provided a solid academic foundation for PredictiX, enabling us to:

- Cite authoritative sources for design decisions
- Position our work within the broader context of medical AI research
- Acknowledge limitations based on established knowledge
- Propose innovations grounded in current understanding

**13. Avoiding Redundant Efforts**

Understanding existing solutions prevented redundant development efforts. Rather than recreating solutions to solved problems, we could focus on genuine innovations:

- Unified multi-disease prediction platform
- Flexible input mechanisms (manual and report upload)
- User-friendly interface for non-technical users
- Accessible web-based deployment

**Conclusion of Survey Analysis**

The literature survey was instrumental in shaping PredictiX's development. It informed our choice of algorithms, guided system architecture decisions, established performance expectations, and identified opportunities for innovation. By building on established research while addressing identified gaps, PredictiX aims to contribute meaningfully to accessible healthcare prediction technology.

The survey confirmed that while individual disease prediction systems exist, there is significant value in creating a unified, accessible, user-friendly platform that combines multiple prediction capabilities. This conclusion validated the core concept behind PredictiX and provided the knowledge foundation necessary for successful implementation.

# Chapter 3
# Project Management

- **PROJECT PLANNING OBJECTIVES**
- **PROJECT SCHEDULING**
- **RISK MANAGEMENT**

## 3.1 PROJECT PLANNING OBJECTIVES

Project management is a critical component of successful software development, encompassing planning, resource allocation, scheduling, risk assessment, and continuous monitoring throughout the project lifecycle. For PredictiX, effective project management ensured that the multi-disease prediction system was developed systematically, meeting quality standards while adhering to time and resource constraints.

This chapter details the comprehensive project management approach adopted for PredictiX, including project planning objectives, resource identification and allocation, development methodology selection, project scheduling with timeline charts, and risk management strategies. The systematic approach enabled the team to navigate technical complexities, coordinate collaborative efforts, and deliver a functional, reliable healthcare prediction system.

Project planning forms the foundation of successful software development. For PredictiX, planning objectives focused on defining clear goals, identifying required resources, selecting appropriate technologies, and establishing a structured development approach. Effective planning enabled the team to anticipate challenges, allocate resources efficiently, and maintain project momentum throughout development.

The project planning objectives include selecting software, defining its scope, and establishing a project development approach using the Software Development Life Cycle (SDLC).

### 3.1.1 SOFTWARE SCOPE

The software scope for PredictiX encompasses the selection of development tools, programming environments, and supporting technologies necessary for building a web-based multi-disease prediction system integrated with machine learning models.

**Visual Studio Code (VS Code)**:

- Visual studio was selected as the primary integrated development environment for the project. VS Code is a lightweight, powerful, and highly customizable code editor that supports multiple programming languages and offers extensive extension capabilities.

**MongoDB Compass:**

- Graphical User Interface: MongoDB Compass provides a user-friendly GUI for interacting with MongoDB databases, enabling developers to visualize data, create and modify collections, and execute queries.
- Schema Visualization: Compass allows for the graphical representation of database schemas, making it easier to understand and manage data structures.

**Node Package Manager (npm):**

- Dependency Management: npm simplifies the process of managing project dependencies, allowing developers to install, update, and remove packages with ease.

- Package Repository: npm provides access to a vast repository of JavaScript packages, enabling developers to leverage existing libraries and frameworks.

**Technological Scope:**

- The MERN stack provides a comprehensive set of technologies for building full-stack web applications. React.js enables the development of dynamic and interactive user interfaces, while Node.js and Express.js handle server-side logic and API development. MongoDB provides a flexible and scalable database solution. The combination of these technologies offers a powerful platform for developing the doctor appointment booking system.

### 3.1.2 RESOURCE

Effective resource management encompasses both human resources (team members, their skills, and role allocation) and environmental resources (infrastructure, tools, and sustainability considerations).

#### 3.1.2.1 Human Resource

Human resources are the most valuable asset in software development. For PredictiX, a team of four members collaborated to develop the multi-disease prediction system, with each member contributing specific expertise while maintaining flexibility in task allocation.

#### 3.1.2.2 Environment Resource

Environmental resource considerations for PredictiX encompassed both technological infrastructure requirements and sustainability aspects.

**Technological Infrastructure:** The project required standard computing devices and reliable internet connectivity for smooth development and collaboration.

**Development Hardware:** Team members utilized personal computing equipment with the following specifications:

**Standard Configuration:**

- **Processor:** Intel Core i5 (8th gen) or AMD Ryzen 5 equivalent
- **RAM:** 8GB DDR4 (minimum), 16GB recommended for ML training
- **Storage:** 256GB SSD for fast operations
- **GPU:** NVIDIA GeForce GTX 1650 or better (for team members training deep learning models)
- **Display:** Full HD (1920x1080) for comfortable development

**Network Infrastructure:**

- Stable broadband internet (minimum 10 Mbps) for collaborative development

- Cloud service access for version control and documentation
- Video conferencing capability for remote team meetings

**Development Software:**

- **Operating Systems:** Windows 10/11, Ubuntu 20.04 LTS, macOS
- **Version Control:** Git and GitHub for code management
- **Communication:** WhatsApp, Discord, Zoom for team collaboration
- **Documentation:** Google Docs, Microsoft Word, Markdown editors

**Cloud Services (Future Consideration):**

- **Hosting:** Heroku, AWS, or Google Cloud Platform
- **GPU Training:** Google Colab for deep learning model training
- **Database:** MongoDB Atlas for cloud database hosting

**Sustainability Considerations:**

PredictiX contributes to environmental sustainability through:

- **Paperless Operations:** All documentation, reports, and communications maintained digitally, eliminating paper consumption
- **Energy Efficiency:** Optimized code and efficient algorithms minimize computational resource requirements and energy consumption
- **Reduced Healthcare Travel:** Preliminary health screening from home reduces unnecessary hospital visits, decreasing carbon emissions from transportation
- **Long-term Viability:** Modular architecture and well-documented code ensure system longevity, reducing the need for complete system rewrites
- **Resource Optimization:** Efficient database queries, caching strategies, and asynchronous operations prevent resource wastage

### 3.1.3 Project Development Approach

The **Iterative Model** was selected as the development methodology for PredictiX. This model involves developing the system through repeated cycles (iterations), with each iteration producing an incrementally refined version.

**Software Development Life Cycle (SDLC) Phases:**

1. **Requirement Gathering and Analysis:**

- Identifying stakeholder needs (patients, doctors, researchers)
- Defining functional requirements (prediction features, input methods)
- Establishing non-functional requirements (performance, security, usability)
- Conducting feasibility analysis

2. **System Design:**
- Architectural design (system components and interactions)
- Database schema design
- User interface wireframingq
- ML/DL algorithm selection

3. **Implementation (Coding):**
- Frontend development (React.js)
- Backend development (Node.js/Express.js)
- ML model development (Python)
- Component integration

4. **Testing and Quality Assurance:**
- Unit testing (individual components)
- Integration testing (component interactions)
- System testing (end-to-end functionality)
- Performance and security testing

5. **Deployment:**
- Environment setup
- Application deployment
- Configuration and initialization

6. **Maintenance and Support:**
- Bug fixes and issue resolution
- Performance optimization
- Feature enhancements
- Model updates and improvements

**Why Iterative Model for PredictiX:**
- **Complexity Management:** Multiple complex components (four ML/DL models, web application, database) developed incrementally
- **Risk Mitigation:** Early iterations identify technical challenges before significant resource investment
- **Flexibility:** Requirements and design refined based on testing outcomes and feedback
- **Continuous Testing:** Each iteration includes testing for early defect detection
- **Stakeholder Feedback:** Regular demonstrations to supervisors provide valuable refinement insights

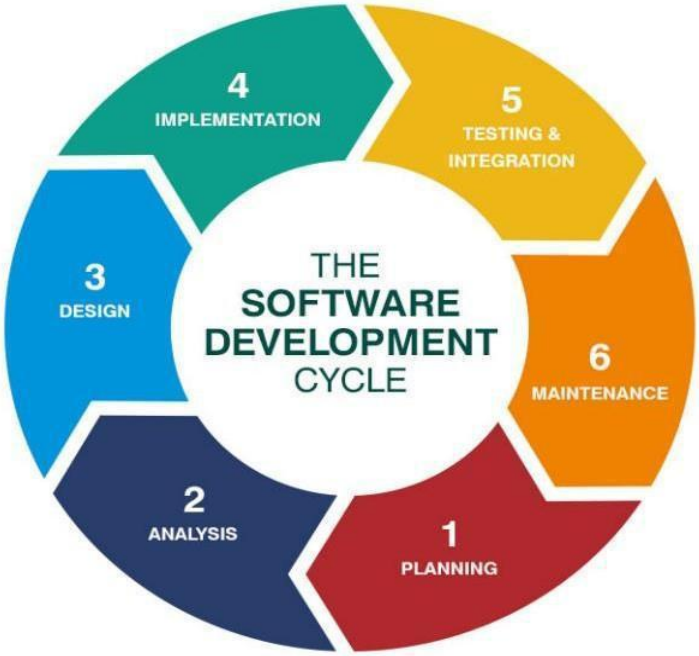


Fig. 3.1.3.1: SDLC (Software Development Life Cycle)

The diagram represents the six key phases of the SDLC — Planning, Analysis, Design, Implementation, Testing & Integration, and Maintenance — illustrating the step-by-step process of developing and managing software efficiently.

## 3.2 PROJECT SCHEDULING

Project scheduling involves planning timelines, allocating tasks, defining milestones, and establishing deadlines to ensure systematic progress and timely completion.

### 3.2.1 Basic Principle Scope

The PredictiX project followed structured scheduling principles:

**1. Phased Development:** The 18-week project was divided into nine distinct phases:

- Phase 1: Requirement Analysis & Design (2 weeks)
- Phase 2: Dataset Collection & Preprocessing (2 weeks)
- Phase 3: ML Model Development & Training (4 weeks)
- Phase 4: Frontend Development (3 weeks)
- Phase 5: Backend Development (3 weeks)
- Phase 6: Model Integration (2 weeks)
- Phase 7: Testing & Debugging (2 weeks)
- Phase 8: Documentation (2 weeks)
- Phase 9: Final Review & Presentation (2 weeks)

**2. Milestone Definition:** Clear milestones established to mark significant achievements:

- M1: Requirements and design approved (Week 2)
- M2: Datasets prepared (Week 4)
- M3: All models trained successfully (Week 8)
- M4: Frontend prototype complete (Week 9)
- M5: Backend APIs functional (Week 10)
- M6: Models integrated (Week 12)
- M7: Testing complete (Week 14)
- M8: Documentation finalized (Week 16)
- M9: Project ready for submission (Week 18)

**3. Task Dependencies:** Tasks sequenced based on dependencies:

- Model training required completed dataset preprocessing
- Backend API development required finalized model interfaces
- Integration testing required completed frontend and backend
- Documentation required functional system

**4. Parallel Development:** Independent tasks executed concurrently:

- Frontend and backend development proceeded simultaneously
- Different team members trained different models in parallel
- Documentation drafted alongside development
- Testing of completed modules while others in development

**5. Buffer Allocation:** Time buffers incorporated for:

- Unexpected technical challenges (10% buffer)
- Model retraining if accuracy targets not met
- Supervisor feedback implementation
- Emergency contingencies (1-week general buffer)

### 3.2.2 Project Organization

The PredictiX team adopted a **collaborative flat structure** with equal participation and shared responsibility.

**Organizational Structure:**

- **No Rigid Hierarchy:** All five team members held equal standing
- **Specialized Responsibilities:** Each member had primary focus areas
- **Cross-functional Collaboration:** Regular knowledge sharing across domains
- **Collective Decision-Making:** Major decisions discussed and agreed upon by team

**Communication Mechanisms:**

1. **Regular Team Meetings:**
   - Weekly progress meetings (every Monday, 2:00-3:30 PM)
   - Technical discussion sessions (2-3 times per week as needed)
   - Supervisor meetings (every Friday for 1 hour)

2. **Online Collaboration:**
   - WhatsApp group for quick questions and updates
   - Discord server for technical discussions and screen sharing
   - Email for formal communications

3. **Code Review Process:**
   - Feature branches for new development
   - Pull requests reviewed by at least one team member
   - Feedback addressed before merging
   - Maintains code quality and knowledge sharing

4. **Version Control:**
   - Git and GitHub for source code management
   - main branch for stable code
   - develop branch for integration
   - Feature branches for individual developments

**Coordination Strategies:**

1. **Sprint-Based Organization:**
   - Two-week sprints with defined goals
   - Sprint planning, daily stand-ups, sprint review, retrospective

2. **Task Tracking:**
   - GitHub Issues for task management
   - Project Board (Kanban): To Do, In Progress, Review, Done
   - Clear ownership and progress visibility

3. **Pair Programming:**
   - Used for complex components (model integration, authentication)
   - Knowledge transfer and skill development
   - Screen sharing for remote collaboration

**Progress Monitoring:**

- Weekly supervisor check-ins
- Peer reviews for quality assurance
- Demo sessions of completed features
- Metrics tracking (model accuracy, development velocity)

### 3.2.3 Timeline Chart

The timeline provides visual representation of project phases, durations, and overall schedule.

#### 3.2.3.1 Time Allocation

| TASK | July | | August | | | September | | October | |
|---|---|---|---|---|---|---|---|---|---|
| | Week 1 | Week 3 | Week 1 | Week 3 | Week 4 | Week 1 | Week 3 | Week 1 | Week 4 |
| Definition , Functionalities, Timeline Chart | | | | | | | | | |
| Problem Formulation, Objectives, Motivation | | | | | | | | | |
| Purpose of the Progress Report | | | | | | | | | |
| Design & Diagrams | | | | | | | | | |
| Implementation | | | | | | | | | |
| Testing | | | | | | | | | |
| Documentation | | | | | | | | | |

Fig. 3.2.3.1: Time Allocation Chart

#### 3.2.3.2 Task Sets

| PROJECT CRITERIA | PRESENTATION |
|---|---|
| Pre Project Discussion | Project definition, Functionalities of project and Timeline chart of implementation. |
| Project Problem Investigation Phase | 1. Explore the Project Problem<br>2. Analyze the Problem<br>3. Define the Problem<br>4. Problem Formulation/Objectives of the Project |
| First and Second Progress Report | Report to summarise the status of project. |
| Initial Project File Checking | Introduction about the project, Design phase, Working Methodology of the project, Results, Conclusion, and Future Directions. |
| Internal Presentation and Viva | Project execution, show skill set on the presented technology along with brief summary about all the phases and modules. |
| Final Documentation | Final hard copy of the file after the changes in the previous phases. |

Table 3.2.3.2: Task Sets

This table outlines the key project stages from Pre-Project Discussion to Final Documentation describing tasks such as problem investigation, progress reporting, initial checks, internal presentation, and final submission, ensuring a structured approach to project development and evaluation.

## 3.3 RISK MANAGEMENT

Risk management involves identifying potential risks, assessing their likelihood and impact, and developing mitigation strategies.

### 3.3.1 Risk Identification

Risk identification recognizes potential threats that could affect project success. For PredictiX, risks were categorized into technical, project management, and operational categories.

**Technical Risks:**

1. **Model Accuracy Below Threshold**
- Models might not achieve sufficient accuracy for reliable predictions
- Impact: System credibility compromised

2. **Model Integration Challenges**
- Difficulty integrating Python ML models with Node.js backend
- Impact: Project delays, potential architecture redesign

3. **Performance Issues**
- Slow prediction response times, especially for image-based models
- Impact: Poor user experience

4. **GPU Resource Constraints**
- Lack of GPU access for training deep learning models
- Impact: Prolonged training times, deployment limitations

5. **Security Vulnerabilities**
- Security flaws exposing sensitive health data
- Impact: Privacy breaches, legal implications

**Project Management Risks:**

1. **Timeline Delays**
- Tasks taking longer than estimated
- Impact: Missed deadlines, rushed final phases

2. **Scope Creep**
- Uncontrolled expansion beyond original objectives
- Impact: Incomplete core features

3. **Team Member Unavailability**
- Members unavailable due to illness or conflicts
- Impact: Work disruption, delayed tasks

4. **Communication Breakdowns**
- Miscommunication leading to duplicated work
- Impact: Wasted effort, integration issues

**Operational Risks:**

1. **Dataset Access Issues**
- Legal restrictions or access problems with datasets
- Impact: Need for alternative datasets, delays

2. **Infrastructure Failures**
- Development machine failures, data loss
- Impact: Work disruption, recovery efforts

#### 3.3.1.1 Risk Identification Artifacts

Risk identification artifacts include documentation of identified risks and their assessment.

**Risk Register:** A comprehensive risk register was maintained documenting:

- Risk description and category
- Potential impact and likelihood
- Risk owner (responsible team member)
- Mitigation strategy
- Current status (active, mitigated, resolved)

**Risk Assessment Meetings:**

- Weekly team meetings included risk review segment
- Emerging risks identified and documented promptly
- Status of existing risks evaluated regularly
- Mitigation strategies adjusted based on project evolution

### 3.3.2 Risk Projection

Risk projection estimates the probability and consequences of identified risks

# CHAPTER 4
# SYSTEM REQUIREMENTS

- **USER CHARACTERISTICS**
- **FUNCTIONAL REQUIREMENT**
- **NON FUNCTIONAL REQUIREMENT**
- **HARDWARE AND SOFTWARE REQUIREMENT**

## 4.1 USER CHARACTERISTICS

The primary users of the **PredictiX – A Multi-Disease Predictor** system will include patients, healthcare professionals, and administrators. Patients will use the system to input their health data and symptoms to receive early disease predictions and preventive recommendations. Healthcare professionals will use the system to analyze patient health trends, validate predictions, and provide accurate diagnoses. Administrators will oversee user management, system updates, and data integrity.

## 4.2 FUNCTIONAL REQUIREMENT

Functional requirements define the essential features, operations, and interactions of the **PredictiX – A Multi-Disease Predictor** system to ensure it meets the needs of users effectively. The key functional requirements include:

- **User Registration and Login:** Users (patients and healthcare professionals) should be able to create accounts, log in securely, and manage their profiles. The system should support password recovery and authentication mechanisms.
- **Health Data Input:** Patients should be able to enter personal health details such as age, gender, weight, height, symptoms, and medical history. The system should validate input data to ensure accuracy and completeness.
- **Disease Prediction Module:** The system should analyze the provided health data using machine learning models to predict the likelihood of multiple diseases such as diabetes, heart disease, and liver disorders.
- **Report Generation:** After analysis, the system should generate a detailed report displaying prediction results, probability percentages, and health recommendations.
- **Doctor Consultation Recommendation:** Based on prediction results, the system should suggest relevant specialists (e.g., cardiologist, endocrinologist) for further consultation.
- **Data Storage and Management:** User data and medical history should be securely stored in the database. The system should allow users to view previous predictions and reports.
- **Admin Dashboard:** Administrators should be able to manage user accounts, update disease prediction models, monitor system performance, and ensure data integrity.

### 4.2.1 ACTIVITY AND PROPOSED SYSTEM

The proposed PredictiX – A Multi-Disease Predictor system aims to provide an intelligent, user-friendly, and efficient platform for predicting multiple diseases based on user-provided health data.

## 4.3 NON FUNCTIONAL REQUIREMENT

Non-functional requirements define the overall quality attributes and constraints of **PredictiX – A Multi-Disease Predictor** system to ensure performance, security, and usability. The non-functional requirements of the system may include:

- **Performance:** The system should deliver accurate predictions quickly, even when processing large datasets or multiple simultaneous user requests. Model inference and report generation should occur within a few seconds.
- **Scalability:** The system should be capable of handling an increasing number of users, datasets, and disease models without performance degradation. It should support model expansion as new diseases are added.
- **Reliability:** The system should maintain high availability with minimal downtime, ensuring consistent and accurate disease prediction results. Regular backups should safeguard model data and user records.
- **Security:** The system should securely store and process sensitive user data, including medical and personal details, using encryption and secure authentication mechanisms to prevent unauthorized access or data breaches.
- **Usability:** The system should provide an intuitive, user-friendly interface suitable for users with varying levels of technical knowledge, ensuring ease of data entry, navigation, and result interpretation.
- **Compatibility:** The system should function seamlessly across multiple devices (desktops, tablets, smartphones) and platforms (Windows, macOS, Android, iOS) using modern web browsers.
- **Accessibility:** The system should be accessible to users with disabilities, adhering to accessibility standards such as WCAG, including features like screen reader compatibility and high-contrast design.
- **Maintainability:** The system should be designed with clean, modular, and well-documented code to facilitate easy updates, debugging, and integration of new predictive models or health features in the future.

## 4.4 HARDWARE AND SOFTWARE REQUIREMENT

### 4.4.1 HARDWARE REQUIREMENT

- Memory: 8 GB RAM (recommended)
- CPU: Intel Core i5 or equivalent (recommended)
- Storage: SSD with sufficient space for database and application files.

### 4.4.2 SOFTWARE REQUIREMENT

- Operating System: Windows, Linux, macOS, Android, iOS (for mobile access)
- Database System: MongoDB

- Backend Environment: Node.js, Express.js
- Frontend Environment: React.js, Tailwind CSS
- Development Environment: VS Code
- Browser: Modern web browsers (Chrome, Firefox, Safari, Edge, Brave)

# Chapter 5
# System Analysis

- **STUDY OF CURRENT SYSTEM**
- **PROBLEMS IN CURRENT SYSTEM**
- **REQUIREMENT OF NEW SYSTEM**
- **PROCESS MODEL**
- **FEASIBILITY STUDY**

## 5.1 STUDY OF CURRENT SYSTEM

In the current healthcare environment, disease prediction and early diagnosis largely depend on manual medical checkups and the expertise of healthcare professionals. Patients typically visit hospitals or clinics for consultations and diagnostic tests, which can be both time-consuming and costly. This traditional approach often delays the detection of diseases, especially for individuals who do not undergo regular health screenings.

In many cases, people rely on self-diagnosis based on symptoms, which can lead to inaccurate conclusions and delayed treatment. Additionally, doctors face challenges in analyzing large amounts of patient data manually, which increases the risk of human error.

Therefore, the current system lacks automation, accuracy, and efficiency — emphasizing the need for a unified, intelligent platform like **PredictiX**, which can analyze multiple health parameters and predict various diseases simultaneously with precision and reliability.

## 5.2 PROBLEMS IN CURRENT SYSTEM

The current disease diagnosis and prediction methods face numerous challenges that impact both patients and healthcare professionals. These problems often lead to delayed treatment, higher medical costs, and reduced accuracy in disease detection. The key issues can be summarized as follows:

**Manual Diagnosis and Human Dependency:**

- Disease detection largely depends on doctors' interpretation of test results, which can vary from one professional to another.
- Manual data analysis increases the risk of human error, leading to incorrect or delayed diagnosis.
- Doctors often need to review large datasets, which can be time-consuming and prone to oversight.

**Limited Accessibility and Awareness:**

- Patients in rural or remote areas lack easy access to advanced diagnostic tools or expert consultations.
- Existing systems often require in-person medical visits, which are time-consuming and expensive.
- Non-technical users find it difficult to use complex medical prediction platforms without guidance.

**Fragmented and Disease-Specific Tools:**

- Current online systems focus on predicting a single disease (e.g., only diabetes or only heart disease).
- Patients have to use multiple platforms to check different conditions, which is inefficient and confusing.
- There is no unified solution that provides a holistic health overview or multi-disease prediction.

**Data Handling and Privacy Concerns:**

- Manual storage of patient data increases the risk of data loss, duplication, or unauthorized access.
- Existing digital platforms may not use secure encryption or authentication mechanisms, risking patient confidentiality.
- Users are often hesitant to input personal medical information due to privacy concerns.

**Lack of Real-Time Analysis and Feedback:**

- Traditional healthcare systems cannot instantly analyze user data to predict health risks.
- Patients must wait for lab results and follow-up appointments to get a complete health assessment.
- This delay can be critical for diseases requiring immediate attention or preventive measures.

By addressing these issues through the implementation of an AI-powered, multi-disease prediction system like PredictiX, healthcare can become more proactive, data-driven, and accessible. Such a system would enable early detection, improve diagnostic accuracy, safeguard patient data, and empower users to make informed health decisions efficiently.

## 5.3 REQUIREMENT OF NEW SYSTEM

The new **PredictiX – A Multi-Disease Predictor** system should offer a smart, reliable, and accessible platform that leverages artificial intelligence and machine learning to predict multiple diseases accurately. It should provide real-time health risk analysis and empower users to take preventive actions before critical conditions develop. The system must be **fast, accurate, and user-friendly**, allowing patients to input medical parameters such as blood pressure, sugar levels, BMI, cholesterol, and other vital signs. It should instantly process this data through predictive models and display detailed health risk reports along with recommendations for medical consultation. Additionally, the system should maintain **secure storage** of user data with proper encryption and authentication to prevent unauthorized access. Doctors and administrators should be able to update disease prediction models, monitor system performance, and analyze aggregated health data for research or healthcare improvement.

## 5.4 PROCESS MODEL

The **Spiral Model**, a risk-driven software development process, is most suitable for the **PredictiX – A Multi-Disease Predictor** system, as it involves complex functionalities like data processing, AI-based prediction, and user data security. The Spiral Model allows the development team to iteratively refine requirements and address potential risks, such as inaccurate predictions or data breaches, at each stage of development. **Detailed Explanation of the Spiral Model's Phases:**

1. **Objectives Determination and Alternative Solutions:**
    - This is the **core phase** of the Spiral Model, emphasizing proactive risk analysis and mitigation.
    - The first step involves defining the **core goals** of the system — such as online patient scheduling, doctor management, secure payment handling, and data protection.
    - **User feedback** plays a vital role. Patients, doctors, and administrators share their preferences and expectations, which helps shape the system's requirements and user experience.
2. **Risk Identification and Resolution:**
    - This is the core of the Spiral model. We identify potential risks, such as security vulnerabilities, scalability issues, or integration challenges.
    - **Security measures** such as encryption, authentication, and role-based access control are planned to safeguard patient data.
    - **Prototypes** are created to test high-risk components. For example, developing a payment gateway prototype to ensure secure transaction flow.
    - This helps in detecting potential problems early, thereby preventing costly issues during later stages of development.
3. **Development and Verification:**
    - The actual **coding and implementation** take place in this phase, starting with core features such as patient registration and appointment scheduling.
    - **Testing and integration** are carried out continuously to maintain high software quality. Unit testing, integration testing, and user acceptance testing ensure functionality and performance.
    - Each completed module is **verified** against requirements to confirm that it meets user expectations and complies with standards.
4. **Review and Planning:**
    - The developed version is **reviewed** by key stakeholders — including patients, doctors, and administrators — to assess usability and performance.
    - **Feedback** is collected and analyzed to refine features and enhance user experience in the next development loop.
    - The team then **plans the next iteration**, addressing newly identified requirements or issues.
    - For instance, if users report difficulty navigating the appointment calendar, the design will be improved in the subsequent cycle.

**Why the Spiral Model is Suitable for a Doctor Appointment Booking System:**

- **Risk Management:** The system deals with **sensitive patient data**, including personal details and medical information, which must be protected from breaches and unauthorized access.
- **Flexibility:** The healthcare domain is dynamic — new technologies, policies, and user expectations may emerge during development.

- **User Feedback:** The success of an appointment booking system heavily depends on its **ease of use and user satisfaction**.
- **Complex Integrations:** The system requires integration with multiple external services such as **payment gateways**, **electronic health record (EHR) systems**, and **patient management platforms**.

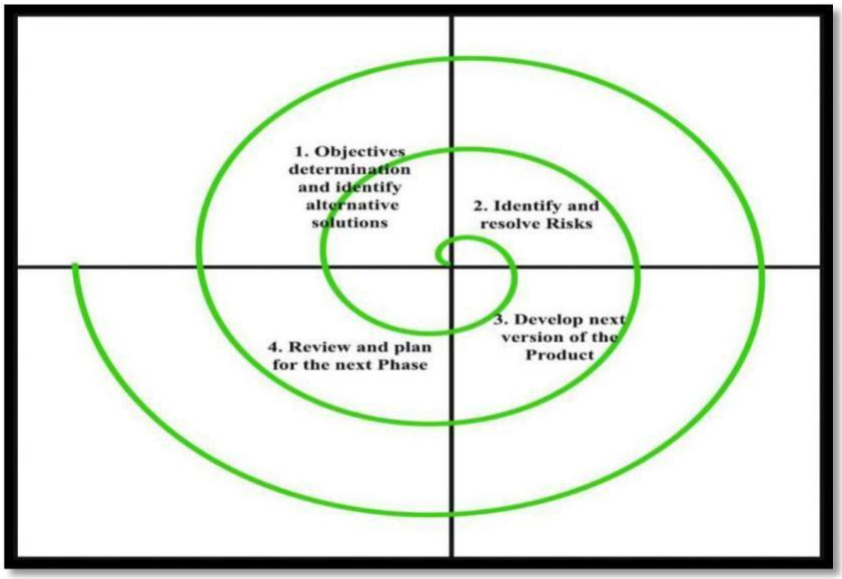


Fig 5.4.1: Spiral Model

## 5.5 FEASIBILITY STUDY

### 5.5.1 TECHNICAL FEASIBILITY

The PredictiX – A Multi-Disease Predictor system will be developed using modern, reliable, and widely supported technologies to ensure robustness and scalability. The backend will be built using Node.js with Express.js to handle data processing and API requests efficiently. The frontend will use React.js combined with Tailwind CSS for a responsive and user-friendly interface, allowing smooth interactions across devices.

### 5.5.2 OPERATIONAL FEASIBILITY

The PredictiX – A Multi-Disease Predictor system will be designed with an intuitive and user-friendly interface, ensuring ease of use for patients, doctors, and healthcare administrators. Users will be able to input symptoms or medical data seamlessly, and the system will provide real-time predictions for multiple diseases.

### 5.5.3 ECONOMICAL FEASIBILITY

The development and deployment costs of PredictiX – A Multi-Disease Predictor will be carefully analyzed against the potential benefits it offers. These benefits

include improved patient care through timely disease prediction, reduced workload for healthcare professionals, and enhanced operational efficiency in clinics and hospitals.

### 5.5.4 SCHEDULE FEASIBILITY

Schedule feasibility assesses whether PredictiX – A Multi-Disease Predictor can be developed and deployed within a realistic and achievable timeline. The project schedule will be carefully planned based on task breakdowns, resource allocation, and interdependencies between modules.
Potential constraints considered in establishing the timeline include:

**Internal Project Constraints:**

- Ensuring developers, AI/ML specialists, and testers are available and have the necessary skills to complete tasks on time.
- Integration of AI/ML models, database systems, and front-end frameworks may introduce delays if unforeseen technical challenges arise.
- Resource allocation must balance development speed with financial constraints to avoid cost overruns.
- Project milestones, including prototype development, model training, system testing, and deployment, must be realistically scheduled to ensure timely completion.

# CHAPTER 6
# DETAIL DESCRIPTION

- **CLIENT AND DOCTOR MODULE**

## 6.1 CLIENT AND DOCTOR MODULE

### 6.1.1 Client Module :

The Client/Patient Module is an essential part of any doctor appointment booking system. This module enables users to place orders and make payment through the platform.

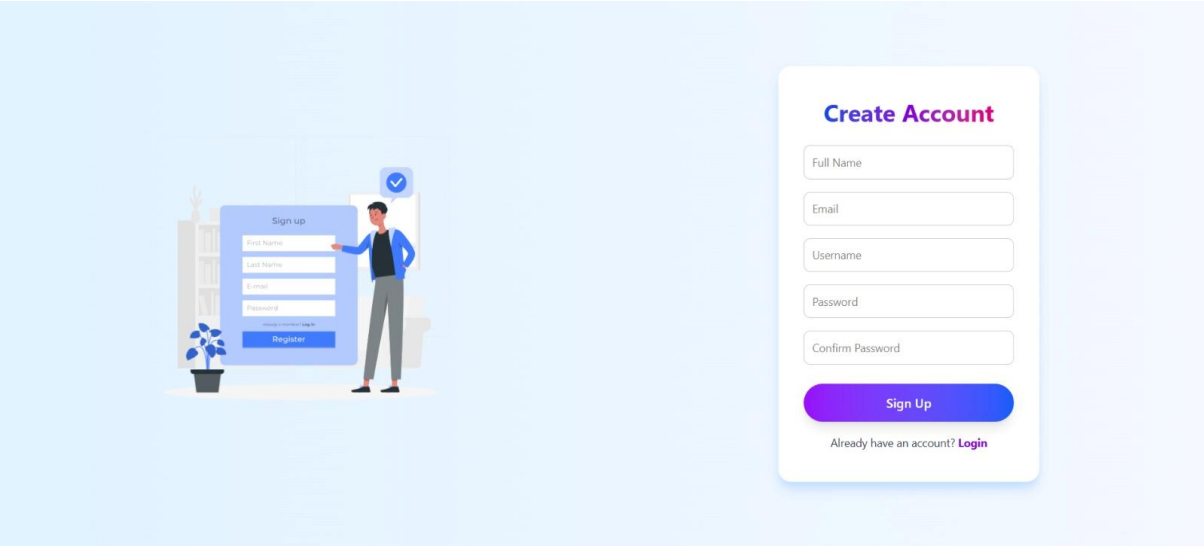


Fig. 6.1.1.1 Create Account

User fills the form with name, email, username, and password to create a new account.
System validates and stores data securely in the database, then allows login access.

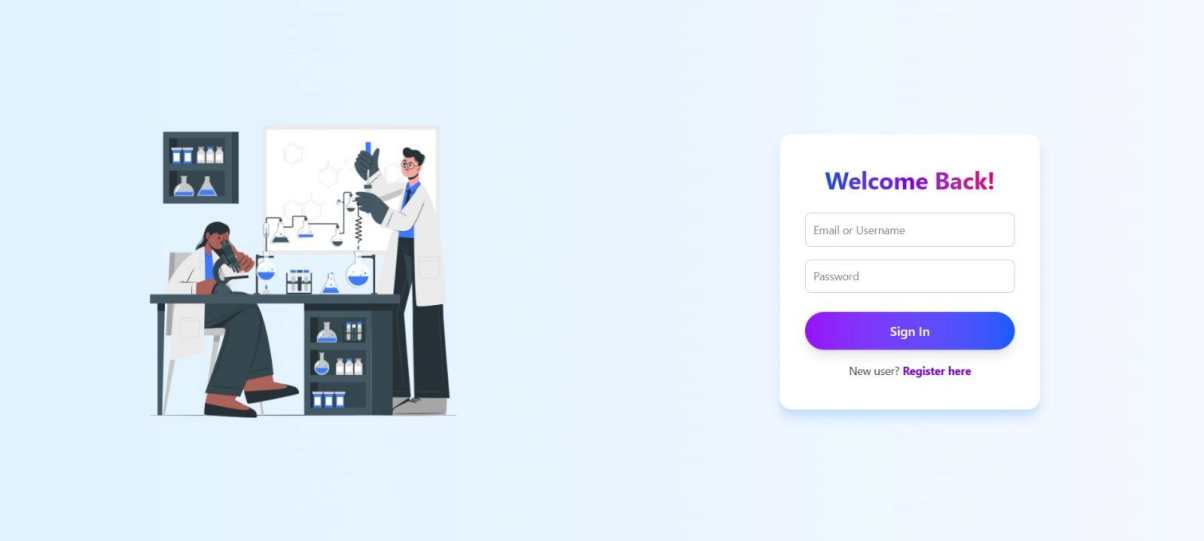


Fig. 6.1.1.2 Login Page

User enters email/username and password to access their existing account.
System verifies login details and grants access to the user’s dashboard or main page.

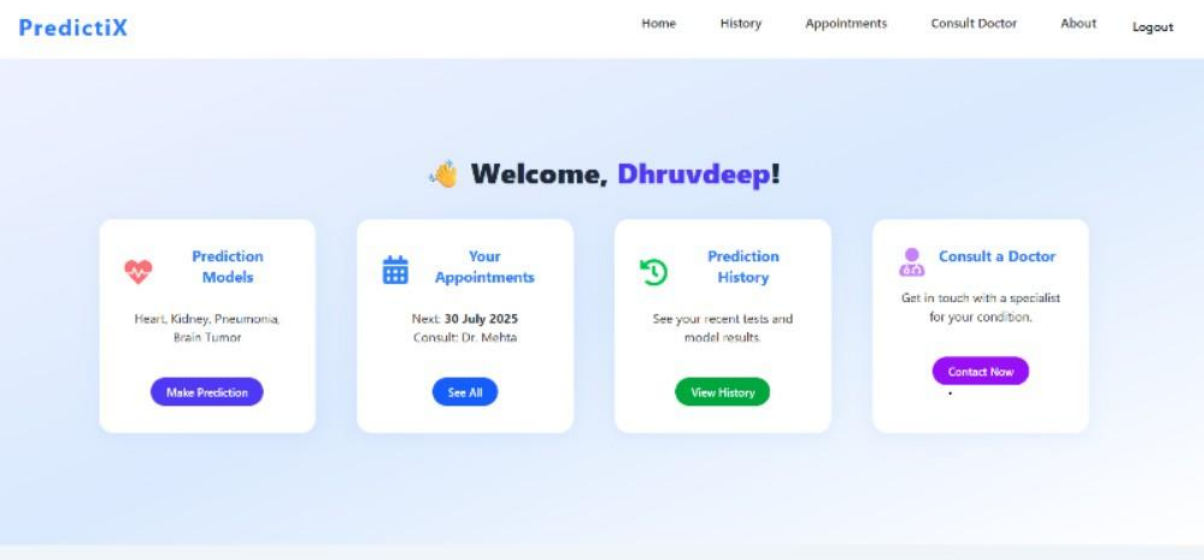


Fig. 6.1.1.3 Home Page

Dashboard displays user details and options for predictions, appointments, history, and doctor consultation.
User can manage health activities like making predictions, viewing history, or contacting a doctor easily.


Fig. 6.1.1.4 Navigation Bar Options

Navigation bar provides quick access to main sections like Home, History, Appointments, Consult Doctor, About, and Logout.

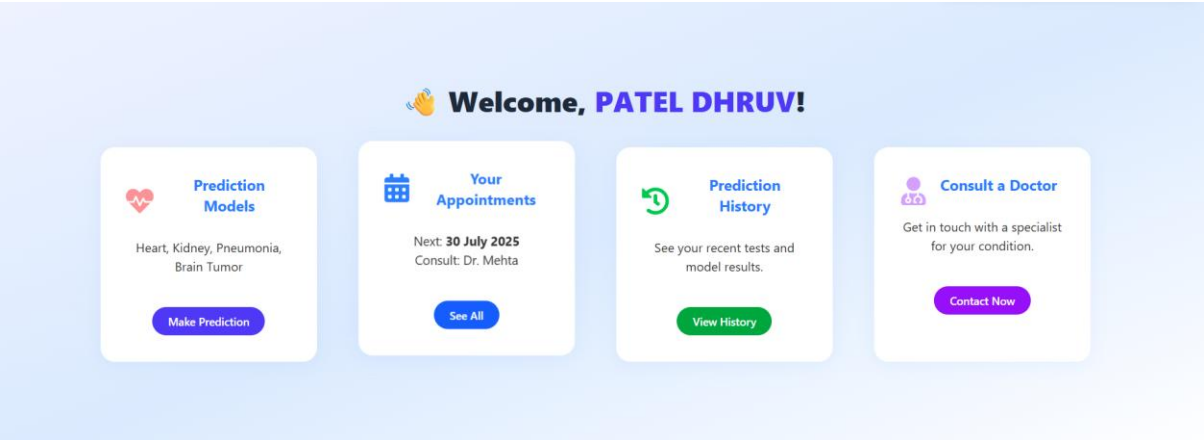


Fig. 6.1.1.5 My Profile Page

Dashboard welcomes the user and displays quick options for predictions, appointments, history, and doctor consultations.

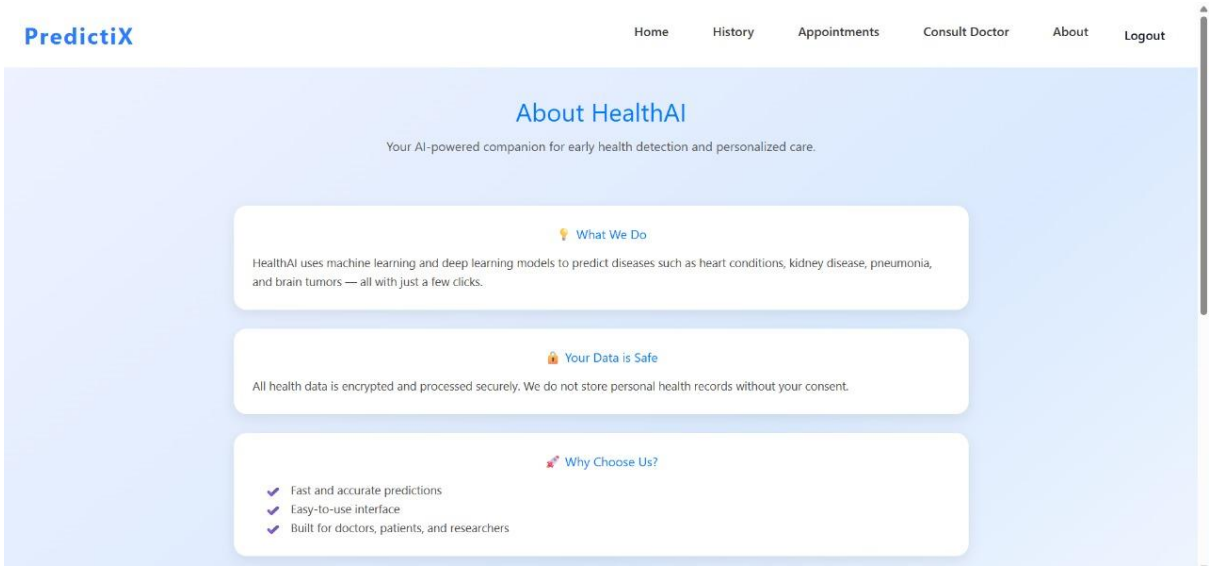


Fig. 6.1.1.6 About Us Page

About page explains HealthAI, describing its AI-based disease prediction and secure data handling.
Highlights features like accuracy, ease of use, and reliability for patients, doctors, and researchers.

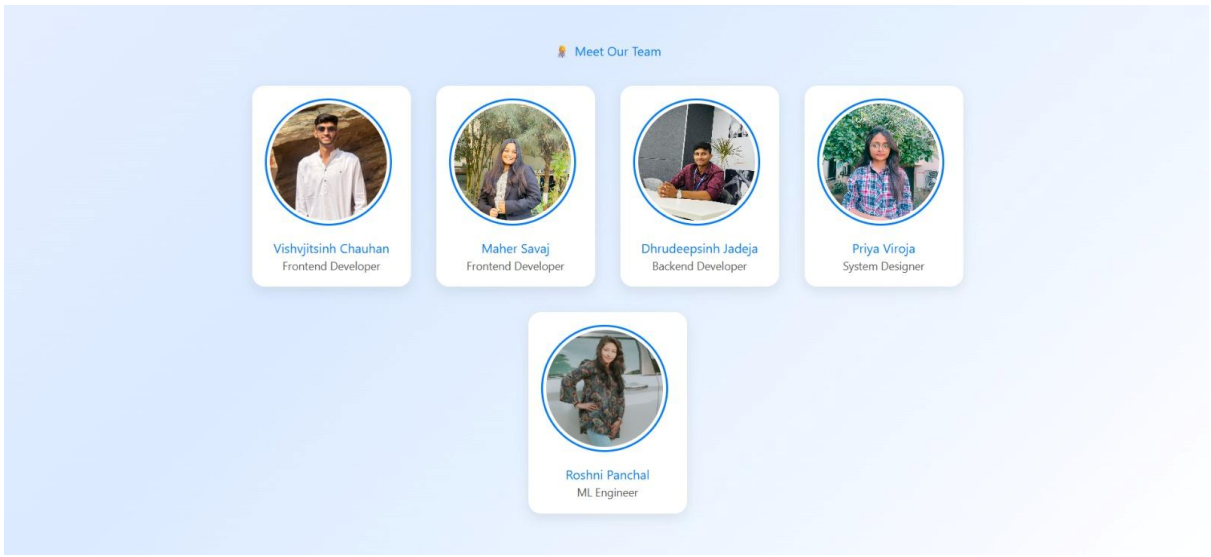


Fig. 6.1.1.7 Contact Us Page

Team section introduces members with their names, photos, and roles in the project

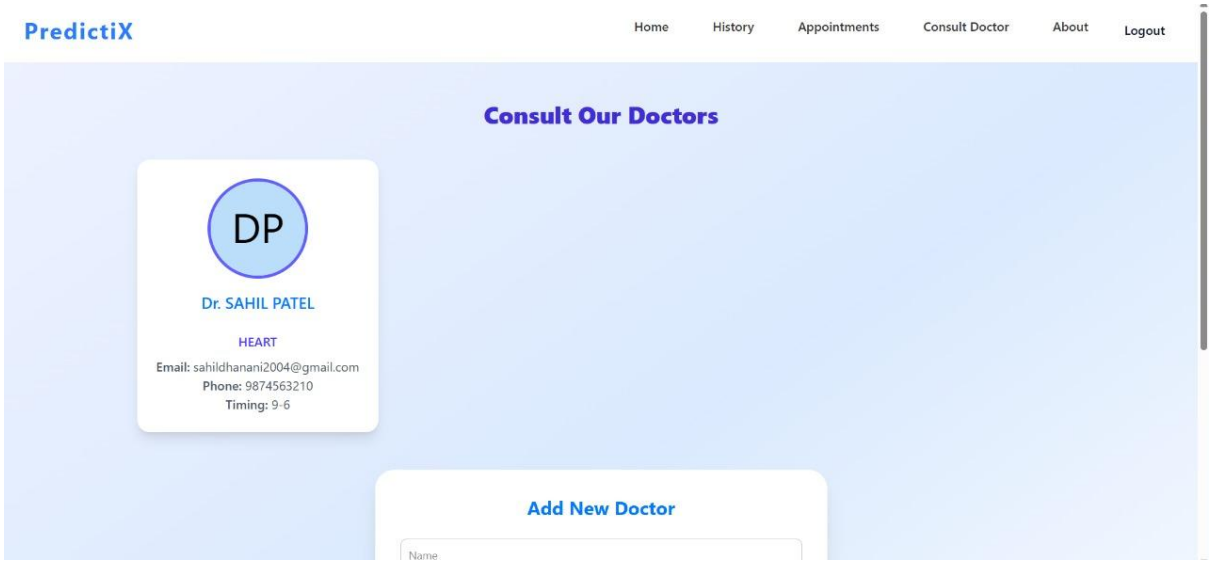


Fig. 6.1.1.8 All Doctors Page

Consult Doctor page displays doctor details like name, specialization, contact info, and available timings.


Fig. 6.1.1.9 Doctor Info with Time Slots

Appointment section lists booked consultations with doctor name, date, and time.
Helps users manage schedules and keep track of upcoming medical appointments easily.

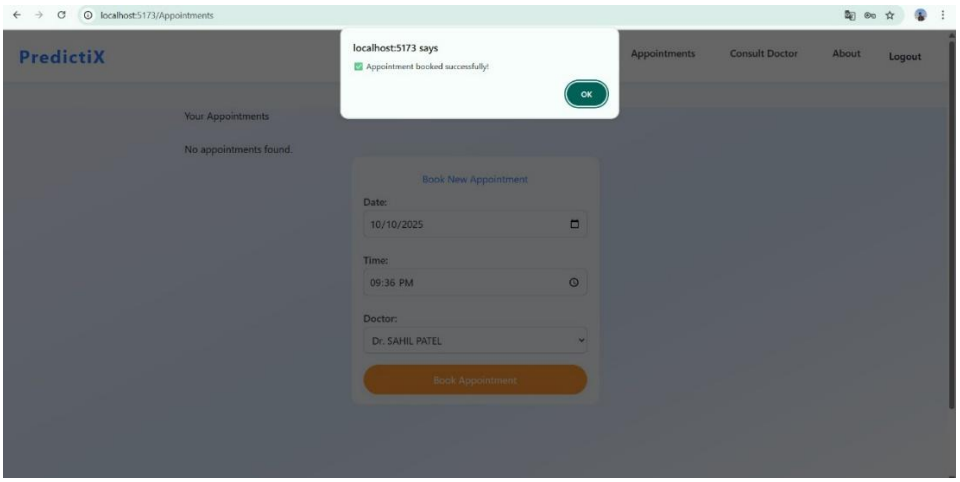


Fig. 6.1.1.10 My Appointments Page

System confirms booking with a success message showing “Appointment booked successfully!

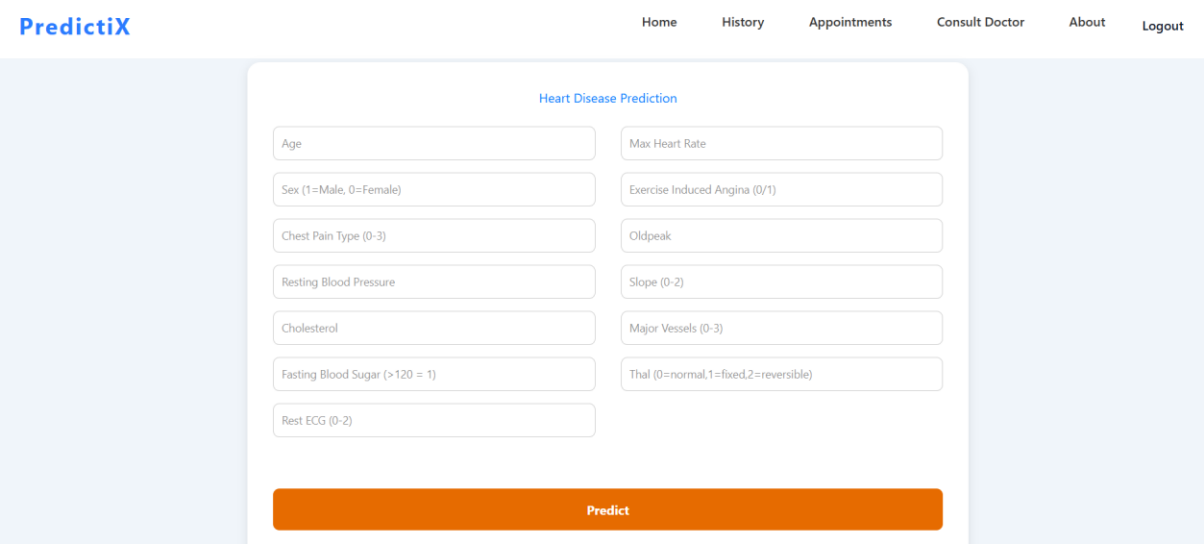


Fig. 6.1.1.11 Heart Disease Prediction Page

Heart Disease Prediction form collects health details like age, blood pressure, cholesterol, and heart rate.

System analyzes inputs using AI models to predict the risk of heart disease accurately.

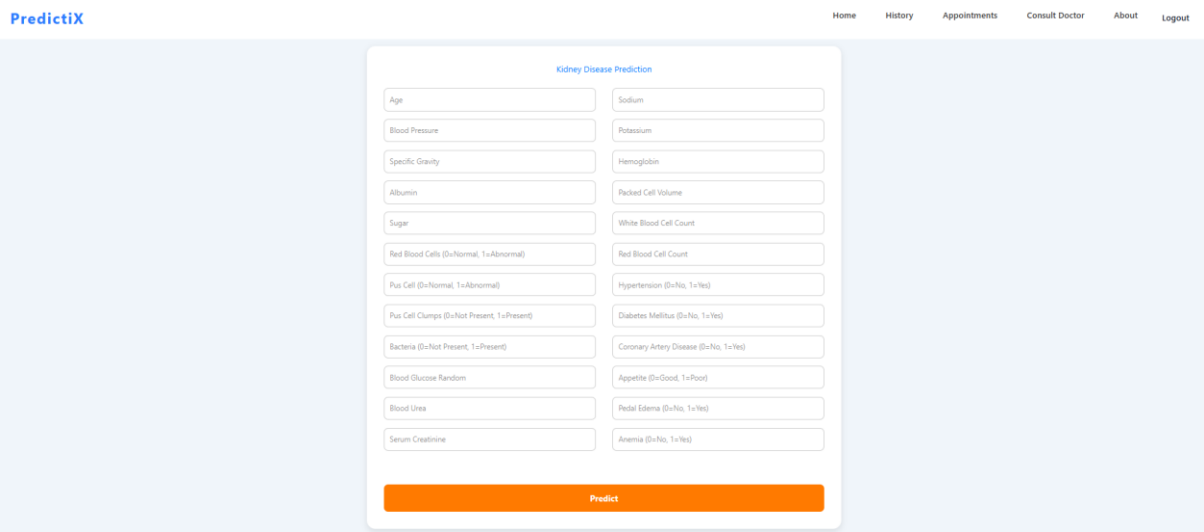


Fig. 6.1.1.12 Kidney Disease Prediction Page

Kidney Disease Prediction form gathers medical inputs like blood pressure, glucose, hemoglobin, and creatinine levels.

AI model processes the data to predict the likelihood of kidney disease for early diagnosis.

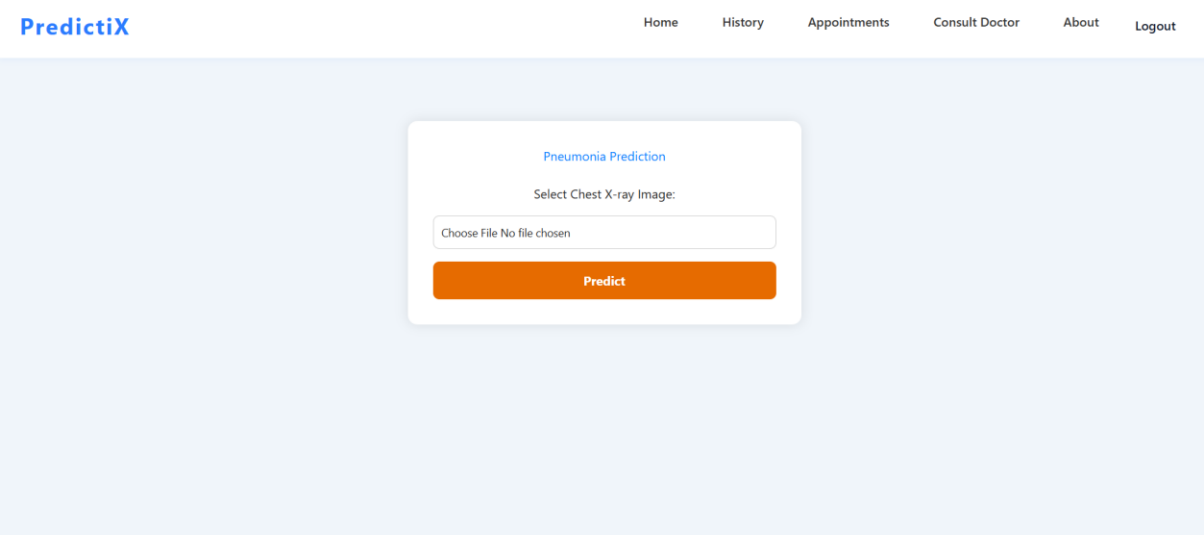


Fig. 6.1.1.13 Pneumonia Disease Prediction Page

Pneumonia Prediction page allows users to upload a chest X-ray image for analysis. AI model scans the image to detect signs of pneumonia and provide prediction results.

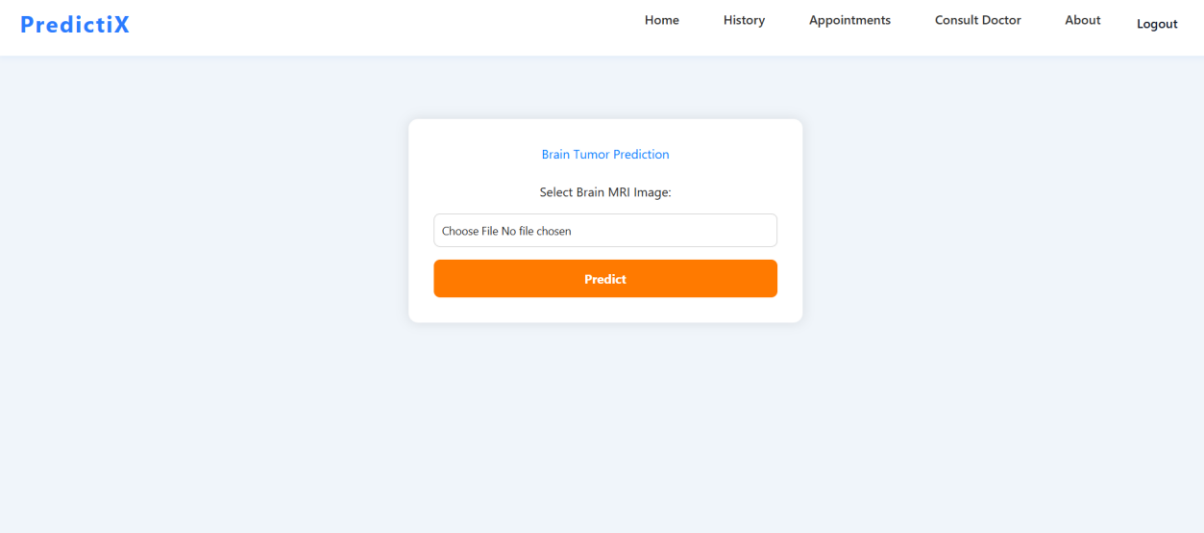


Fig. 6.1.1.14 Brain Tumor Disease Prediction Page

Brain Tumor Prediction page enables users to upload a brain MRI image for analysis. AI model evaluates the scan to detect the presence of a brain tumor and generate prediction results.

# CHAPTER 7
# TESTING

- **BLACK-BOX TESTING**
- **WHITE-BOX TESTING**
- **TEST CASES**

## 7.1 BLACK-BOX TESTING

Black-box testing is a method of software testing that examines the functionality of an application without peering into its internal structures or workings. This method of testing can be applied to virtually every level of software testing: unit, integration, system, and acceptance. It is sometimes referred to as specification-based testing. The tester is oblivious to the system architecture and does not have access to the source code. Typically, while performing a black-box test, a tester will interact with the system's user interface by providing inputs and examining outputs without knowing how and where the inputs are worked upon.

Following are some techniques that can be used for designing black box tests:

- **Equivalence Partitioning*:*** It is a software test design technique that involves dividing input values into valid and invalid partitions and selecting representative values from each partition as test data.
- **Boundary Value Analysis*:*** It is a software test design technique that involves the determination of boundaries for input values and selecting values that are at the boundaries and just inside/ outside of the boundaries as test data.

**Advantages :-**

- Tests are done from a user's point of view and will help in exposing discrepancies in the specifications.
- Testers need not know programming languages or how the software has been implemented.
- Tests can be conducted by a body independent from the developers, allowing for an objective perspective and the avoidance of developer- bias.

**Disadvantages :-**

- Only a small number of possible inputs can be tested and many program paths will be left untested.
- Tests can be made redundant if the software designer/developer has already run a test case.

## 7.2 WHITE-BOX TESTING

White-box testing (also known as Glass Box Testing, Transparent Box Testing, Structural Testing) is a software testing method in which the internal structure / design / implementation of the item being tested is known to the tester. The tester chooses inputs to exercise paths through the code and determines the appropriate outputs. White box testing is testing beyond the user interface and into the nitty-gritty of a system. This method is named so because the software program, in the eyes of the tester, is like a white/transparent box; inside which one clearly sees.

**Advantages: -**

- Testing can be commenced at an earlier stage. One need not wait for the GUI to be available.
- It helps in optimizing the code.
- Extra lines of code can be removed which can bring in hidden defects.
- Testing is more thorough, with the possibility of covering most paths.

**Disadvantages: -**

- Since tests can be very complex, highly skilled resources are required, with a thorough knowledge of programming and implementation.
- Test script maintenance can be a burden if the implementation changes too frequently.
- Since this method of testing is closely tied to the application being tools to cater to every kind of implementation/platform may not be readily available.

## 7.3 TEST CASES

Test cases are the specific scenarios designed to evaluate the system's functionality, performance, and behavior. In the context of the PredictiX system, the following are some of the test cases that can be used to ensure reliability and accuracy:

| Serial No. | Test Case | Expected Result | Test Result |
|---|---|---|---|
| **1a** | Patient attempts to register with valid credentials. | Patient account is created successfully, and the patient is redirected to the login page. | Pass |
| **1b** | Patient attempts to register with invalid/missing credentials. | An error message is displayed, indicating invalid or missing fields. Registration fails. | Fail |
| **2a** | Patient logs in with valid credentials. | Patient is successfully logged in and redirected to the dashboard. | Pass |
| **2b** | Patient logs in with invalid credentials. | Error message appears for incorrect credentials. Login fails. | Fail |
| **3a** | Patient uploads a valid **Chest X-ray** image for Pneumonia prediction. | System processes the image and displays the prediction result (e.g., *Pneumonia Detected* or *Normal*). | Pass |
| **3b** | Patient uploads an invalid file (e.g., non-image or corrupted file). | Error message displayed: "Invalid file format." Prediction not performed. | Fail |
| **3c** | Patient uploads a valid **Brain MRI** image for Tumor prediction. | System processes the image and displays result (*Tumor Detected* or *No Tumor*). | Pass |
| **3d** | Patient uploads an MRI image larger than allowed size. | Error message displayed: "File size too large." Prediction fails. | Fail |
| **4a** | Patient views prediction history. | All previous prediction records with date, result, and image are displayed. | Pass |
| **4b** | Patient has no previous predictions. | Message displayed: "No prediction history found." | Pass |

| | | | |
|---|---|---|---|
| **5a** | Patient attempts to book an appointment with an available doctor. | Appointment successfully booked; confirmation message shown. | Pass |
| **5b** | Patient attempts to book an already booked slot. | Error message displayed: “Slot unavailable.” Booking fails. | Fail |
| **6a** | Patient cancels an existing appointment. | Appointment canceled successfully; both doctor and patient schedules updated. | Pass |
| **6b** | Patient attempts to cancel a non-existing appointment. | System shows: “Appointment does not exist.” | Fail |
| **7a** | Doctor logs in with valid credentials. | Doctor redirected to doctor dashboard successfully. | Pass |
| **7b** | Doctor logs in with invalid credentials. | Error message displayed: “Invalid login details.” | Fail |
| **8a** | Doctor updates profile information (specialization, contact, etc.) with valid data. | Profile updated successfully and reflected on patient search results. | Pass |
| **8b** | Doctor updates profile with invalid/missing data. | Error message displayed: “Invalid input fields.” Update fails. | Fail |
| **9a** | Doctor views list of upcoming appointments. | List displays all patient appointments with date and time. | Pass |
| **9b** | Doctor has no upcoming appointments. | Message displayed: “No scheduled appointments.” | Pass |
| **10a** | Admin logs in with valid credentials. | Admin redirected to admin dashboard successfully. | Pass |

| | | | |
|---|---|---|---|
| **10b** | Admin logs in with invalid credentials. | Error message displayed: “Incorrect credentials.” | Fail |
| **11a** | Admin adds a new doctor with valid information. | Doctor account created successfully and added to list. | Pass |
| **11b** | Admin adds a doctor with invalid/missing fields. | Error message displayed and account creation fails. | Fail |
| **12a** | Admin deletes an existing doctor account. | Doctor successfully removed from the system. | Pass |
| **12b** | Admin tries to delete a doctor that doesn’t exist. | Error message displayed: “Doctor not found.” | Fail |
| **13a** | Patient views “Consult Doctor” page. | Available doctors displayed with specialization and contact info. | Pass |
| **13b** | No doctors available in the system. | Message displayed: “No doctors found.” | Pass |
| **14a** | System displays prediction page UI correctly (for Pneumonia/Brain Tumor). | Upload box, predict button, and navigation bar are visible and functional. | Pass |
| **14b** | Prediction page fails to load completely. | Error message displayed or blank page shown. | Fail |
| **15a** | Patient logs out successfully. | Redirected to homepage or login page with message: “Logout successful.” | Pass |
| **15b** | Patient session timeout occurs. | System logs out automatically and redirects to login page. | Pass |
| **16a** | System stores and retrieves prediction results efficiently. | History loads quickly and accurately. | Pass |
| **16b** | System fails to retrieve prediction results due to database issue. | Error message: “Unable to load history.” | Fail |

| | | | |
|---|---|---|---|
| **17a** | System handles concurrent uploads for predictions (multiple users). | No system crash; each prediction processed independently. | Pass |
| **17b** | Concurrent uploads cause server error or incorrect result. | System crash or incorrect output. | Fail |
| **18a** | User navigates through all website modules (Home, About, Appointments, etc.). | All pages load correctly with consistent UI. | Pass |
| **18b** | Navigation link broken or page not found. | “404 Page Not Found” error. | Fail |
| **19a** | System performance under multiple user requests. | Website remains stable and responsive. | Pass |
| **19b** | Website slows down or crashes under load. | Fail | Fail |

Table 7.3.1: Test Cases

# CHAPTER 8
# SYSTEM DESIGN

- **CLASS DIAGRAM**
- **USE-CASE DIAGRAM**
- **ACTIVITY DIAGRAM**
- **DATA FLOW DIAGRAM**
- **SEQUENCE DIAGRAM**

## 8.1 CLASS DIAGRAM

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application.

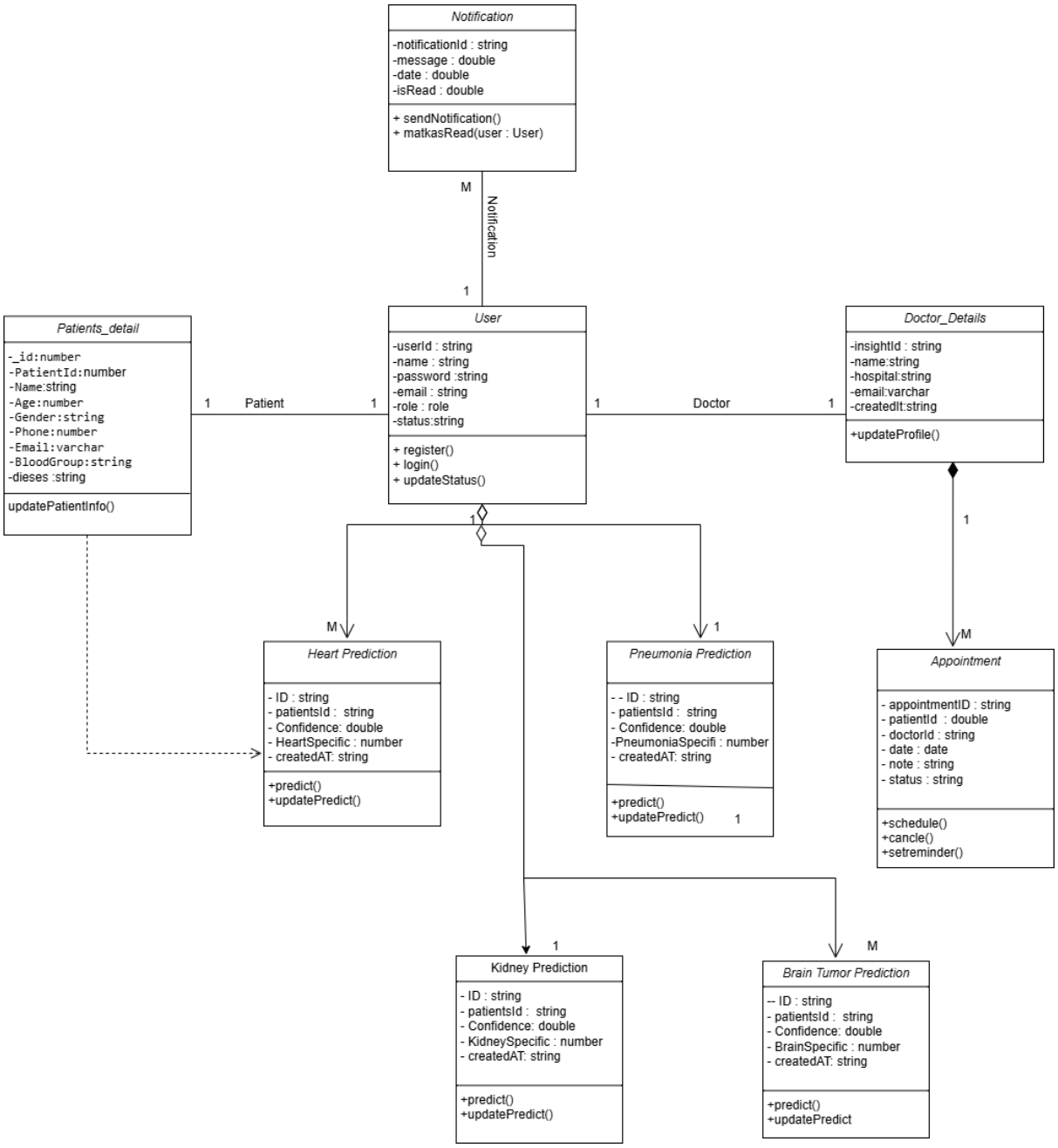


Fig. 8.1 Class Diagram

## 8.2 USE-CASE DIAGRAM

A use case diagram is used to represent the dynamic behavior of a system. It encapsulates the system's functionality by incorporating use cases, actors, and their relationships.

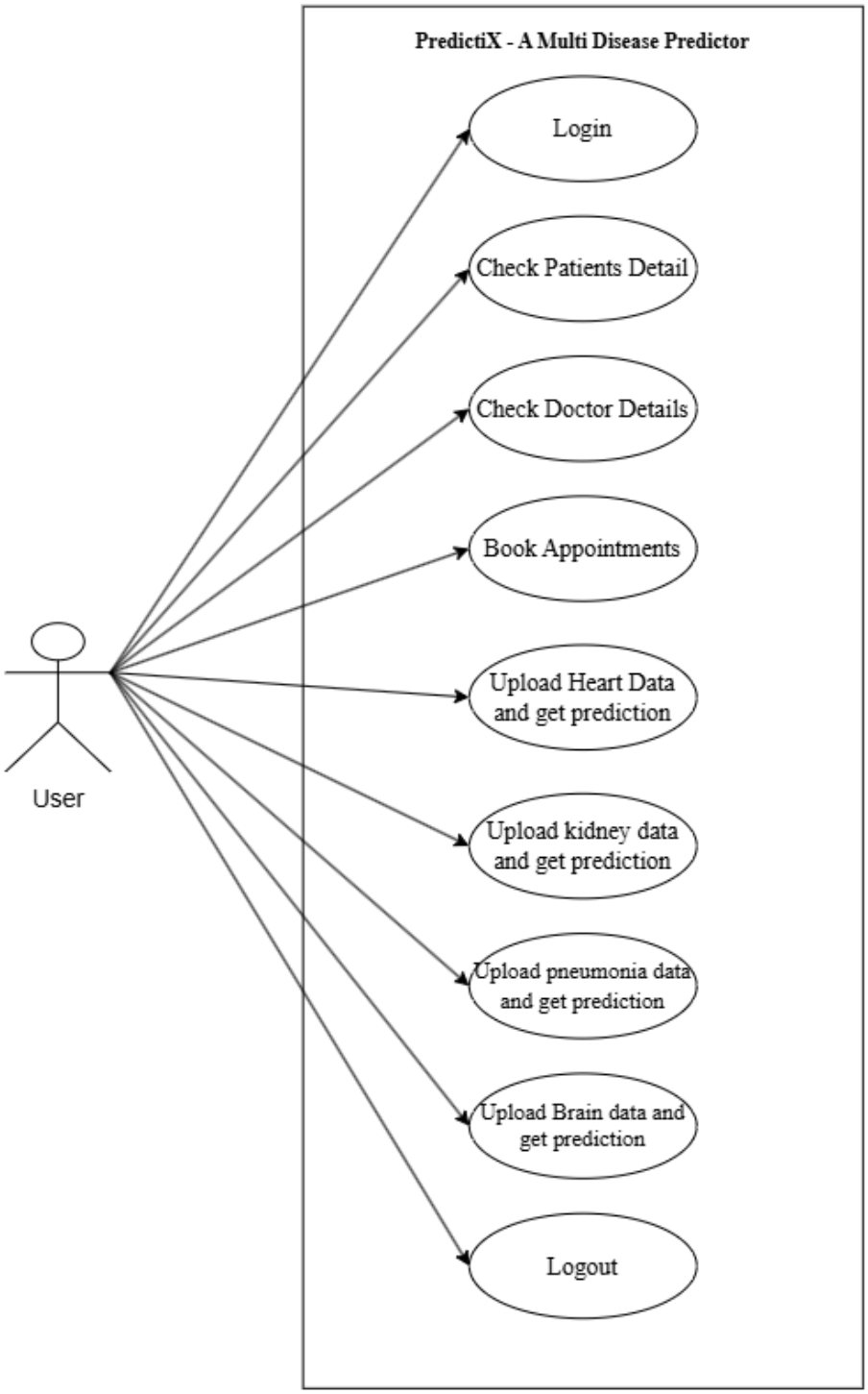


Fig. 8.2 Use-Case Diagram

## 8.3 ACTIVITY DIAGRAM

Activity diagram is essentially an advanced version of flow chart that modeling the flow from one activity to another activity.

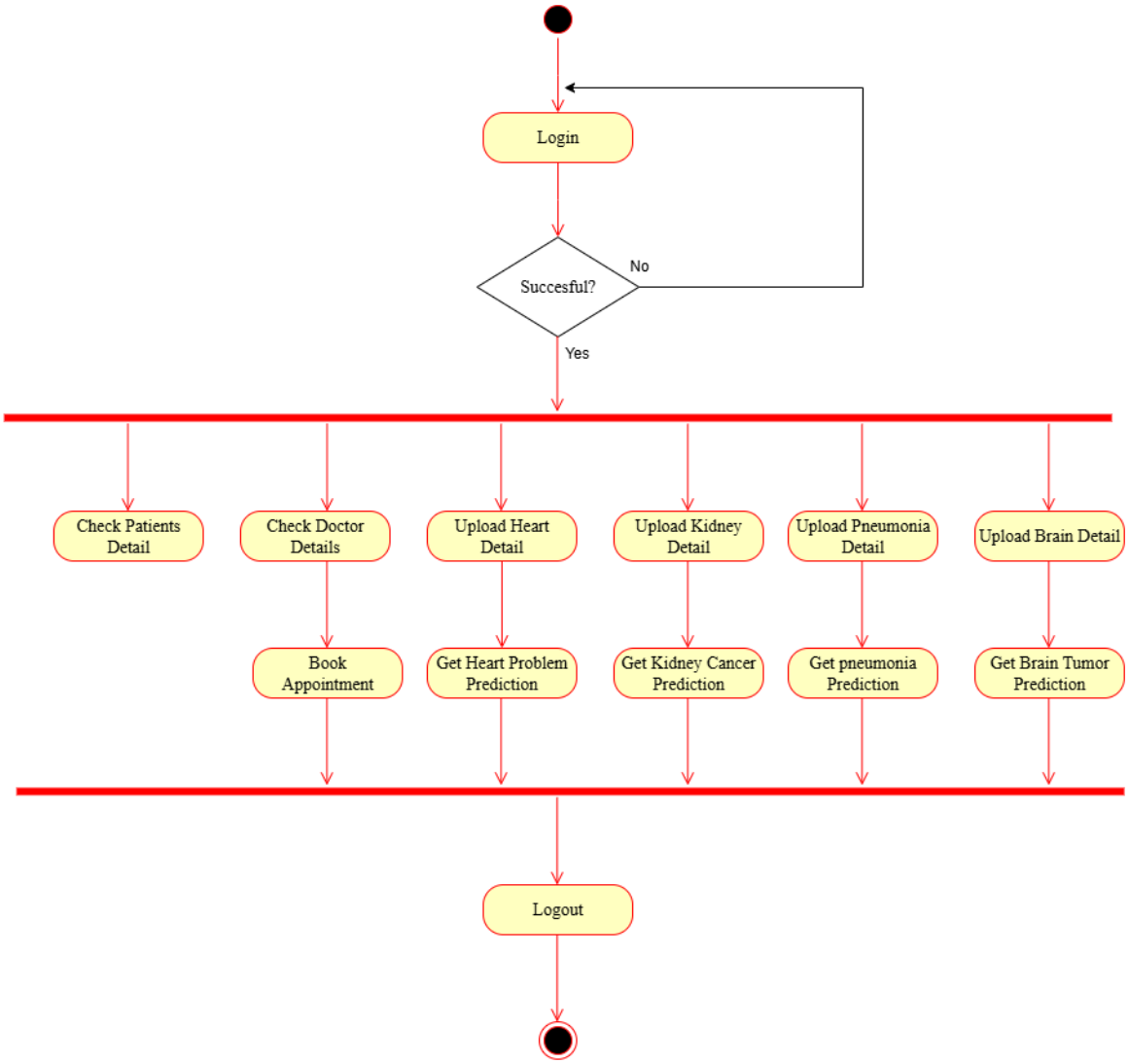


Fig. 8.3 Activity Diagram

## 8.4 DATA FLOW DIAGRAM

A data flow diagram (DFD) maps out the flow of information for any process or system.

**Context Diagram [LEVEL – 0]**

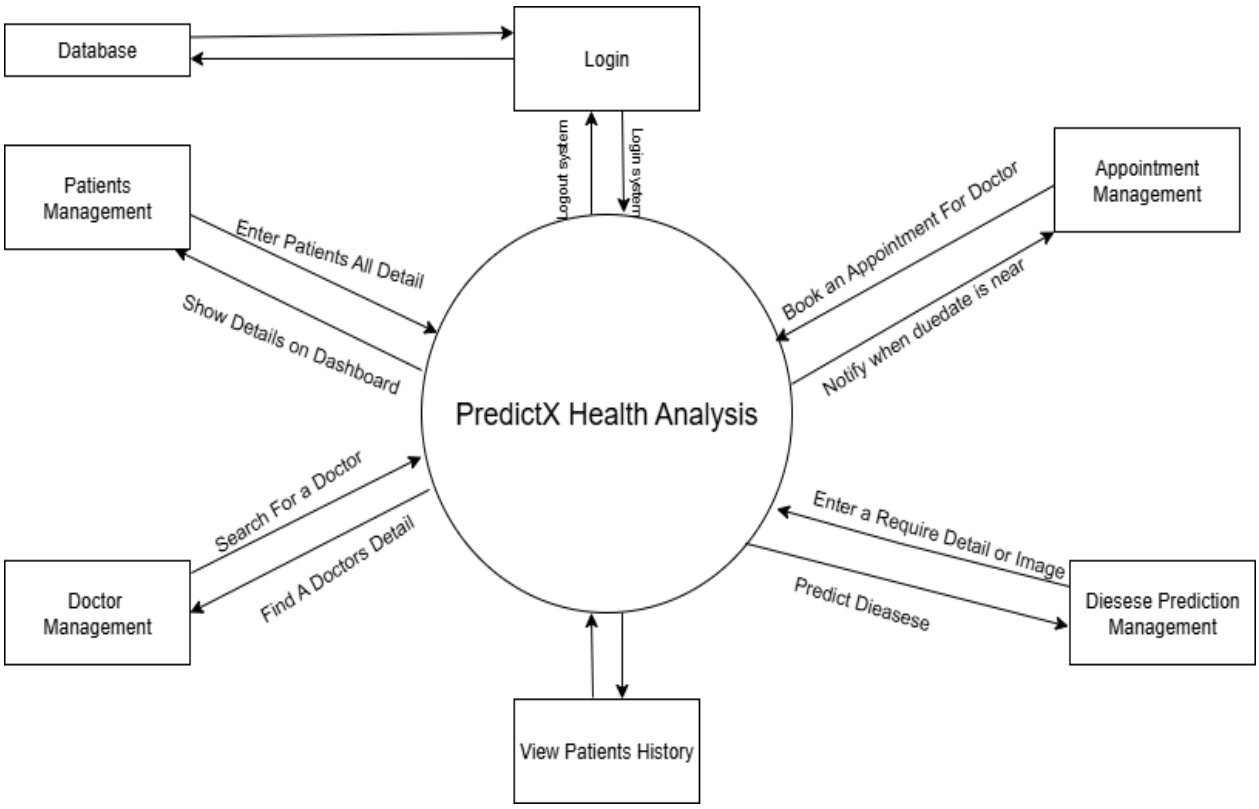


Fig. 8.4.1 Context Diagram [Level-0]

**First Level Diagram**

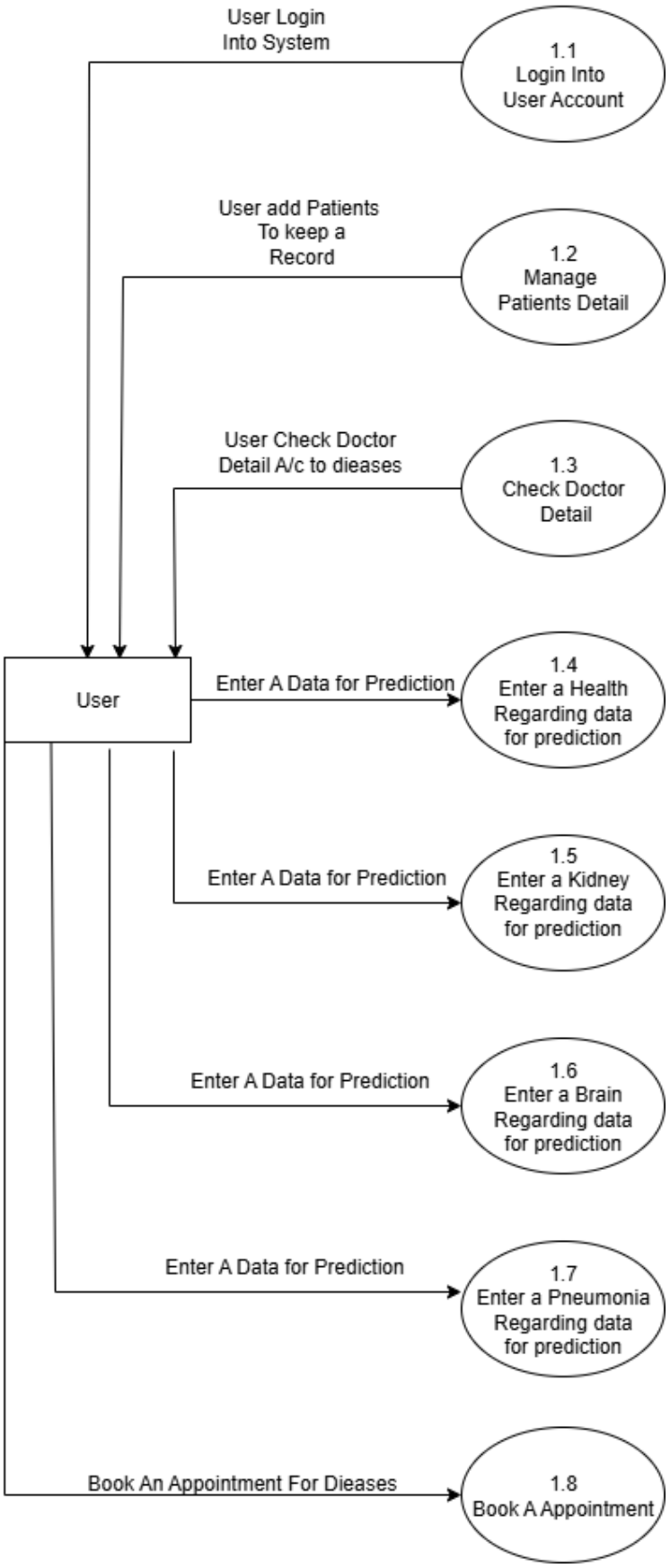


Fig. 8.4.2 First Level Diagram

**Second Level Diagram**

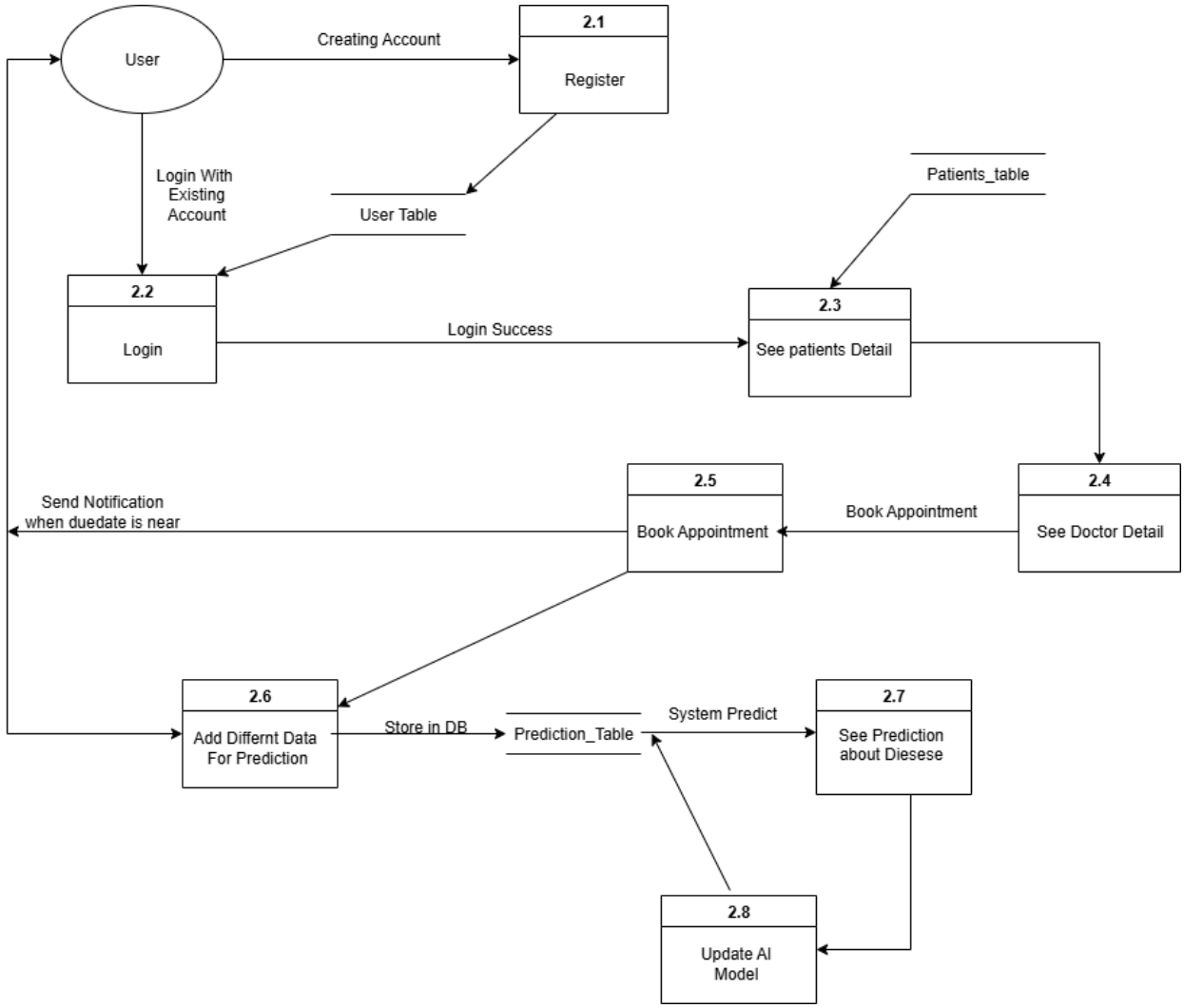


Fig. 8.4.3 Second Level Diagram

## 8.5 OBJECT DIAGRAM

An object diagram is a type of UML diagram that shows a snapshot of objects and their relationships at a particular moment in time. It represents instances of classes (objects) and the links between them. Object diagrams help visualize how objects interact in a real scenario.

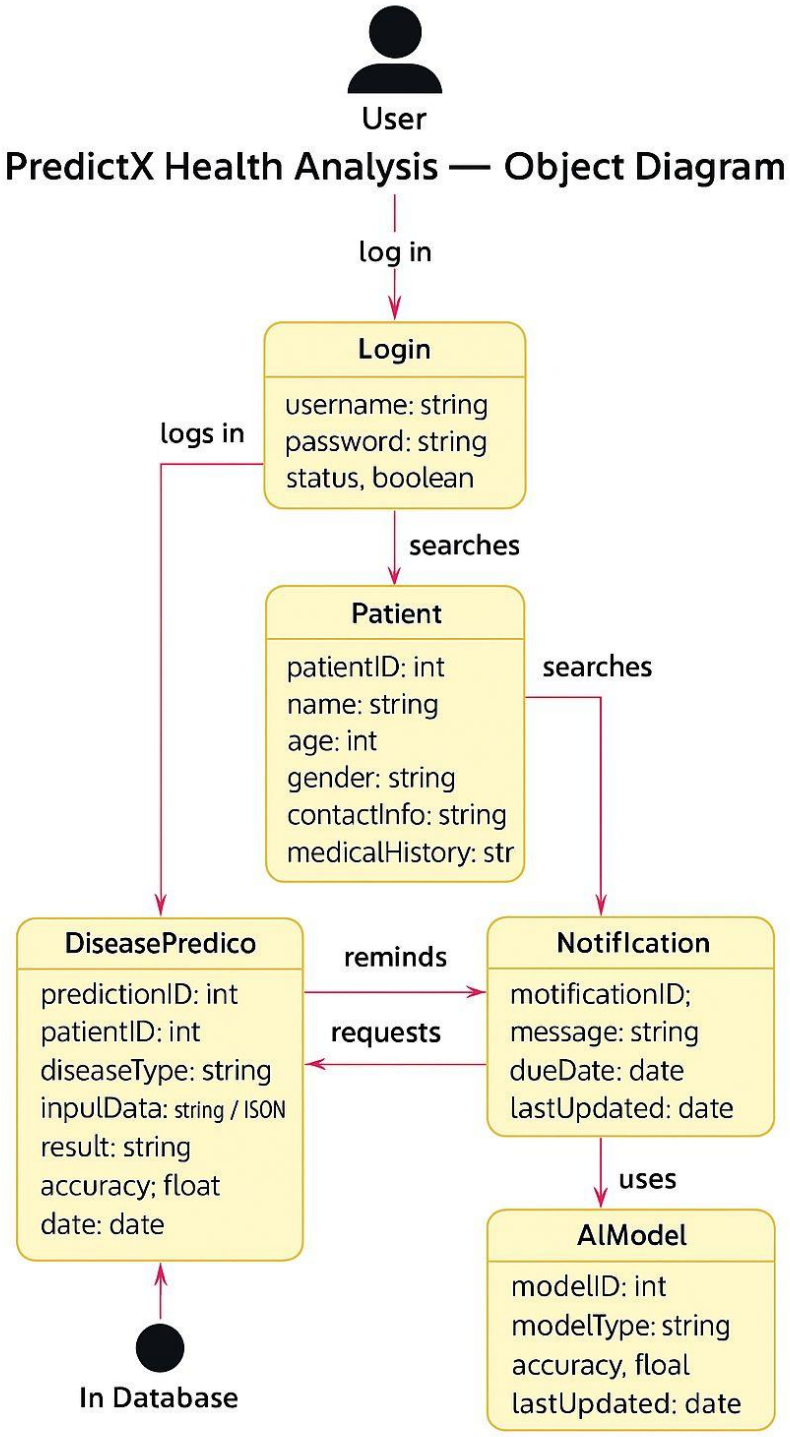


Fig. 8.5 Object Diagram

## 8.6 COMPONENT DIAGRAM

A component diagram illustrates how different parts of a system are organized and connected. It shows the relationships between software components, such as modules, databases, and interfaces, helping to understand the system's structure and how each component interacts to perform specific functions.

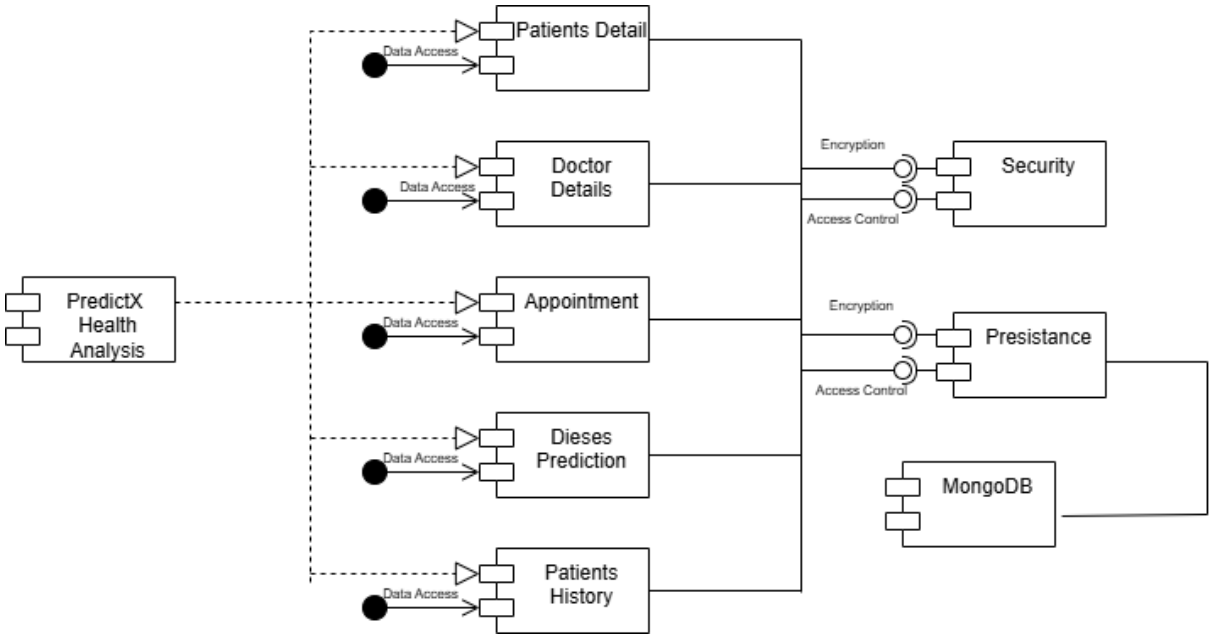


Fig. 8.6 Component Diagram

## 8.7 SEQUENCE DIAGRAM

This model is used to represent the interaction between objects in a sequential order. It shows how objects communicate with each other through messages over time. This model provides a clear visualization of system behavior and the order of operations, making it easy to understand and design interactions.

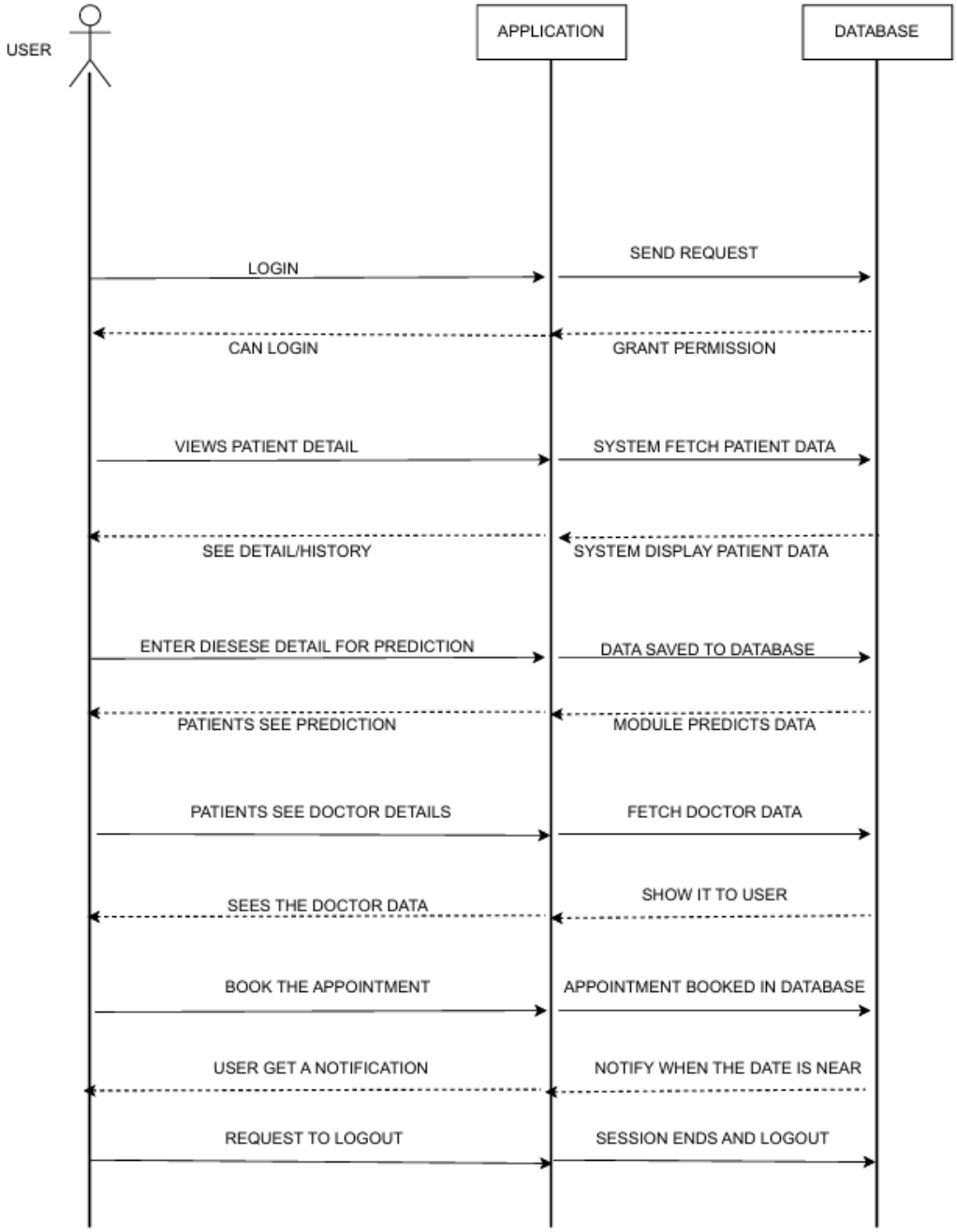


Fig. 8.7 Sequence Diagram

# Chapter 9
# Limitation And Future Enhancements

- **LIMITATION**
- **FUTURE ENHANCEMENT**

## 9.1 LIMITATION

Despite our efforts to develop a smart, flexible, and user-friendly healthcare platform through **PredictiX**, certain limitations remain. While the system integrates AI-based disease prediction (Heart, Kidney, Pneumonia, Brain Tumor) and online doctor appointment functionalities, a few advanced features could not be implemented due to time and resource constraints. Additionally, handling real-time medical data with full-scale accuracy and privacy involves complex challenges that require continuous optimization.

We have aimed to ensure the system is accessible and easy to use for patients, doctors, and administrators. However, we recognize that new users may initially face some usability challenges, especially in understanding disease input parameters.

**Limitations of the PredictiX System include:**

- Due to time constraints, **automated data export and report generation** (e.g., PDF summaries of predictions or appointment records) have not been developed.
- The system currently relies on **static machine learning models** and does not yet support continuous learning or model retraining with new medical data.
- **Real-time integration with hospital databases or APIs** for doctor availability and live patient monitoring has not been implemented.
- The disease prediction models are based on **standard datasets**, which may limit accuracy for diverse patient demographics.
- **Offline functionality** is not supported since the system primarily depends on cloud-based real-time data management.

## 9.2 FUTURE ENHANCEMENT

The future scope of PredictiX involves expanding its functionality and improving its accuracy, scalability, and accessibility. Key enhancements include:

- **Advanced Prediction Algorithms:** We will integrate more sophisticated AI and deep learning models to enhance the accuracy of tumor detection and classification.
- **Cloud-Based Deployment:** We plan to host the system on secure cloud servers to ensure global accessibility, enabling users and healthcare professionals to access the platform anytime, anywhere.
- **Performance Optimization:** We will focus on optimizing model performance and response time through efficient data processing and parallel computation techniques.
- **Database Management:** We aim to implement optimized and scalable databases to handle large volumes of MRI data, improving data retrieval and system efficiency.
- **User Interface Enhancement:** We will improve the user interface with better visualization tools, interactive reports, and easier navigation for all users.

- **Mobile and IoT Integration:** We plan to develop a mobile application and integrate with IoT or wearable health devices to collect real-time health data for predictive analysis.
- **Security and Backup Mechanisms:** We will strengthen data security with advanced encryption and implement regular backups to prevent data loss and ensure system resilience.

These enhancements will make PredictiX more reliable, intelligent, and accessible, ensuring it continues to evolve with advancements in AI and healthcare technology.

# Chapter 10
# Conclusion

- **CONCLUSION**

## 10.1 CONCLUSION

Our PredictiX system is designed as a comprehensive and user-friendly healthcare platform that integrates intelligent disease prediction with online doctor appointment management. The primary goal of this project was to create an accessible, efficient, and reliable system for patients, doctors, and administrators, combining AI-driven medical prediction with digital healthcare management.

PredictiX allows patients to easily predict potential health risks related to Heart Disease, Kidney Disease, Pneumonia, and Brain Tumor using machine learning models. It also enables patients to consult with doctors, book appointments, manage schedules, and receive reminders, all within a single, unified platform.

The system incorporates role-based interfaces for patients, doctors, and admins — ensuring smooth operations, data management, and secure access control. The admin can manage doctors and patient records, while doctors can view appointments and update their profiles.

In conclusion, the PredictiX system successfully fulfills the intended objectives by addressing real-world challenges in healthcare management and prediction.

Key accomplishments include:

- Clearly defining the project's objectives, scope, and healthcare applicability.
- Integrating multiple machine learning models for accurate disease prediction.
- Developing a robust appointment scheduling and management module.
- Designing intuitive and responsive user interfaces for all user roles.
- Ensuring data reliability, security, and efficient information flow between users.
- Thoroughly testing the system using defined test cases to ensure performance, usability, and stability.

Overall, PredictiX bridges the gap between artificial intelligence and healthcare accessibility, offering an effective digital solution for early disease prediction and streamlined doctor-patient interaction.

# Bibliography

# REFERENCES

**List of Web References:**


1. "React"
   https://react.dev/ *[Online Access 27/09/2025, 10:00 am]*

2. "Node.js Run JavaScript Everywhere"
   https://nodejs.org/en/ *[Online Access 27/09/2025, 10:00 am]*

3. "Express Fast, unopinionated, minimalist web framework for Node.js"
   https://expressjs.com/ *[Online Access 27/09/2025, 10:00 am]*

4. "The database for dynamic, demanding software"
   https://www.mongodb.com/ *[Online Access 31/10/2025, 9:30 am]*

5. "Rapidly build modern websites without ever leaving your HTML"
   https://tailwindcss.com/ *[Online Access 10/10/2025, 9:00 am]*

6. "Mongoose"
   https://mongoosejs.com/ *[Online Access 07/10/2025, 8:00 pm]*

7. "JSON Web Token(JWT) Debugg"
   https://jwt.io/ *[Online Access 31/10/2025, 9:00 am]*

8. "Vite, The build tool for the web"
   https://vite.dev/ *[Online Access 09/09/2025, 10:00 am]*

9. "Turn knowledge into action with agents"
   https://textcortex.com/ *[Online Access 27/10/2025, 9:00 am]*

10. "Visualize Engaging Experiences"
    https://cloudinary.com/ *[Online Access 09/09/2025, 11:00 am]*

11. "FreeProjectz"
    https://www.freeprojectz.com/ *[Online Access 09/09/2025, 11:00 am]*