MEDI-AI
Intelligent Disease Prediction and Medicine Recommendation & Health Blogging System

Major Project Report
(MCA 4061)

Submitted in partial fulfillment of the requirement for the award of the degree of
Master of Computer Application

Submitted by

Group No.: 59
Shobhit Shukla (Univ. Roll No: 31222391644)

Under the supervision of
Dr. Mamta Tiwari | Assistant Professor
Mr. Arpit Dubey | Assistant Professor

Department of Computer Application
School of Engineering & Technology (UIET)

Chhatrapati Shahu Ji Maharaj (CSJM) University
UP State University | Formerly Kanpur University
Accredited 'A++' by NAAC | UGC Category-I University
Kanpur (UP)
(2025)

DECLARATION

I, **Shobhit Shukla** hereby declare that the work done in this Major Project (MCA-4061), has been carried out by me under the supervision of **Dr Mamta Tiwari |** *Assistant Professor* and **Mr. Arpit Dubey |** *Assistant Professor*, Department of Computer Applications, School of Engineering & Technology (UIET), Chhatrapati Shahu Ji Maharaj University (CSJMU), Kalyanpur, Kanpur.
Further, I declare that no part of this major project has been submitted either in part or full for the award of any degree to this university or any other university.

Date: (Shobhit Shukla)
Place: Kanpur (University Roll No.: 31222391644)

Chhatrapati Shahu Ji Maharaj (CSJM) University

UP State University | Formerly Kanpur University
Accredited 'A++' by NAAC | UGC Category-I University
Kalyanpur, Kanpur (UP)

CERTIFICATE

This is to certify that the Major Project Report entitled "**MEDI-AI** : Intelligent Disease Prediction and Medicine Recommendation & Health Blogging System" submitted by (**Shobhit Shukla** | Gr. No.: 59) (University Roll No.: 31222391644) for the award of the degree of **Master of Computer Application** to the Department of Computer Application, School of Engineering & Technology (UIET), Chhatrapati Shahu Ji Maharaj University, Kanpur is a bonafide work carried out by him under my supervision and guidance. It is further certified that he is a bonafide student at this department of CSJM University.

Date:
Place: Kanpur

(Dr. Mamta Tiwari)

(Mr. Arpit Dubey)

Signature

(Prof. Rabins Porwal)
Professor | Head
Department of Computer Application
School of Engineering & Technology (UIET)
CSJM University, Kanpur

(External Examiner)

Acknowledgement

I would like to take this opportunity to express my heartfelt gratitude to all those who have supported and guided me in the successful completion of my Major Project titled **"MEDI-AI: Intelligent Disease Prediction and Medicine Recommendation & Health Blogging System."**

First and foremost, I extend my sincere thanks to my project mentor, **Dr. Mamta Tiwari and Mr. Arpit Dubey**, for his invaluable guidance, constant encouragement, and insightful feedback throughout the development of this project. His support has played a crucial role in shaping my ideas and helping me overcome various challenges during the course of this work.

I am also grateful to **Prof. Rabins Porwal**, Head of the Department of Computer Applications, CSJMU, Kanpur, for providing the necessary academic environment and resources that enabled me to carry out this project effectively.

My sincere appreciation goes to all the faculty members of the Department of Computer Applications for their continuous support, motivation, and the knowledge imparted throughout my academic journey.

I would also like to thank my friends for their constant encouragement, technical discussions, and moral support, which greatly contributed to the smooth execution of this project.

This project has been a valuable learning experience and a major milestone in my academic career.

Date: (Shobhit Shukla)
Place: Kanpur (University Roll No.: 31222391644)

Table of Contents

List of Figures

1. Introduction along with Literature Survey and Objectives

1.1 Introduction

In recent years, the healthcare industry has increasingly turned to artificial intelligence (AI) and data-driven technologies to improve diagnosis, treatment planning, and patient engagement. With the global burden of disease on the rise and healthcare systems under strain, intelligent medical support systems have become essential for ensuring accessible, timely, and personalized healthcare solutions. However, despite xthe progress in AI and digital health, there still exists a significant gap between the potential of these technologies and their implementation in everyday healthcare—particularly for early disease prediction and basic medical guidance for the general population.

In many developing regions, people often delay consulting a doctor due to time constraints, financial limitations, or lack of immediate access to medical facilities. As a result, symptoms that could have been addressed early are neglected, leading to worsened conditions and costly treatments later. This scenario highlights the critical need for an intelligent system that can assist users by analyzing their symptoms and providing preliminary guidance. Such a system should also be easy to use, widely accessible, and backed by reliable medical data.

The project was initiated as a practical response to the lack of easy-to-implement academic platforms in mid-sized institutions. In such scenarios, delays, data loss, and communication gaps were common. By designing a platform that brings all major academic functionalities under one system, this project attempts to reduce these inefficiencies and improve the overall academic experience for both faculty and students.

MEDI-AI was conceptualized and developed to meet these needs. It is a web-based, AI-powered **Medicine Recommendation System** integrated with a user-driven **Health Blogging Platform**. MEDI-AI not only enables users to input their symptoms and receive probable disease predictions with associated medical recommendations, but also fosters a community-based environment where users can create, explore, and engage with health-related blogs.

The system utilizes machine learning—specifically, a Support Vector Machine (SVM) classifier trained on a curated dataset mapping symptoms to diseases. Upon predicting a condition, MEDI-AI recommends appropriate medications, dietary plans, workouts, and precautions. This acts as a virtual assistant that empowers users with immediate, personalized health insights, even before reaching a healthcare provider.

Additionally, the health blogging module of MEDI-AI offers a platform for individuals to share their experiences, raise awareness, and discuss treatments, wellness routines, and preventive measures. Users can post blogs with images, like or dislike entries, comment on others' posts, and maintain their blog history. This feature encourages community engagement, health education, and peer learning, making MEDI-AI not just a medical tool but a holistic health ecosystem.

The project is built using Flask [1] (Python [2]) for backend logic, SQLite [6] for data storage, HTML/CSS/Bootstrap [8] for the front end, and Jinja2 [9] for templating. It follows secure coding practices with hashed passwords, session-based user authentication, and validation for file uploads and user inputs.

Overall, MEDI-AI is a practical solution designed to bridge the gap between users and primary health information using intelligent computing and community engagement. It reflects the growing importance of AI in medicine and digital platforms in health awareness. By offering an interactive and intelligent environment, MEDI-AI enhances both self-care and health literacy, setting the stage for more advanced telemedicine systems in the future.

1.2 Literature Survey

The use of Artificial Intelligence (AI) in healthcare has gained significant attention over the past decade. From disease diagnosis to patient monitoring and personalized treatment planning, AI-driven solutions have shown potential in improving both the accuracy and efficiency of medical services. One of the key areas of development has been **symptom-based disease prediction**, which allows users to receive preliminary diagnoses based on their reported health issues, reducing dependency on in-person consultations for basic medical queries.

Several studies and tools have emerged in this domain. IBM's Watson and Google Health have demonstrated how machine learning can assist in diagnostics by analyzing patient data. Meanwhile, platforms like **Ada Health** and **Symptomate** use AI-based questionnaires to suggest possible conditions based on user-reported symptoms. However, many of these systems are either proprietary, complex for laypersons, or lack integration with additional lifestyle recommendations such as diet, exercise, or medication insights.

Moreover, recent research in journals like *Nature Digital Medicine* and *Journal of Biomedical Informatics* has highlighted the potential of supervised learning algorithms—such as Support Vector Machines (SVMs), Decision Trees, and Neural Networks—in achieving high accuracy in disease prediction models when trained on comprehensive symptom-disease datasets. These models often rely on binary symptom vectors as input and map them to a finite set of diseases. While accurate, these approaches often stop at diagnosis and do not provide holistic treatment recommendations or engage users in a community-centric environment.

On the other hand, **health blogs** and patient support communities have emerged as powerful tools for sharing personal medical experiences, offering emotional support,

and spreading awareness. Yet, these platforms typically operate independently of intelligent recommendation systems.
MEDI-AI uniquely combines both aspects: intelligent disease prediction and community engagement through health blogs. It addresses the gap between automated diagnosis tools and peer-to-peer knowledge sharing by integrating them into a single platform. This convergence of AI and social interaction not only aids early detection but also promotes health education, behavior change, and proactive wellness habits—making it a powerful asset in the evolving landscape of digital healthcare.

1.3 Nature of the Problem

In a country like India, and many parts of the world, people often face barriers in accessing timely and affordable healthcare. The unavailability of qualified medical professionals in rural areas, long waiting times in urban hospitals, and rising costs of consultation have led individuals to either self-medicate or ignore symptoms altogether. This delay in seeking medical attention can result in severe complications or chronic conditions that could have been prevented with early intervention.
Moreover, individuals often lack a reliable and personalized source for preliminary medical guidance. While the internet offers vast information, it is scattered, generic, and sometimes misleading. Users may encounter conflicting opinions, unverified remedies, or incomplete understanding of their health issues, which may worsen their condition or create unnecessary anxiety. The absence of AI-powered, easy-to-use platforms for basic disease prediction and recommendation makes it difficult for users to make informed decisions in the early stages of illness.
Additionally, people are increasingly looking for community support to share their experiences and learn from others. However, health forums or blogs often lack structure, content moderation, and integration with intelligent systems. As a result, valuable insights remain scattered and underutilized.
The dual gap—lack of accessible medical prediction tools and a structured health community—presents a pressing problem that needs an innovative solution. There is a growing need for a unified platform that allows users to input their symptoms, receive intelligent recommendations, and also engage with a community that shares similar experiences
MEDI-AI is designed to solve this problem. It empowers users with AI-based health predictions while also offering a safe and informative blogging platform. The system acts as a virtual guide for users seeking clarity on symptoms, lifestyle advice, and community support—all in one place.

1.4 Purpose and Contributions

The primary purpose of the **MEDI-AI** project is to bridge the gap between patients and early-stage medical consultation through the power of Artificial Intelligence and digital communication. It is designed to serve as an intelligent healthcare assistant that not only predicts potential diseases based on a user's symptoms but also provides personalized recommendations to guide users toward appropriate self-care, lifestyle adjustments, and awareness.

Many individuals, especially in underserved or rural areas, do not have immediate access to qualified medical practitioners. As a result, they may delay treatment or rely on unverified online sources. MEDI-AI provides a safe, structured, and intelligent alternative. By allowing users to input their symptoms, the system applies a trained Support Vector Machine (SVM) model to predict a likely disease. Based on the result, it recommends suitable **medications**, **diet plans**, **workouts**, and **precautionary measures** drawn from curated medical datasets. This assists users in taking timely, informed actions regarding their health.

A unique contribution of the project is its integration of a **Health Blogging System**, which encourages users to share their health experiences, treatment journeys, wellness tips, and recovery stories. This community-driven component promotes health literacy, emotional support, and peer learning. Users can register, publish blogs, upload related images, comment on others' posts, and react with likes/dislikes—creating an engaging and interactive platform.

Technically, MEDI-AI contributes as a full-stack web solution that combines **machine learning**, **web development**, **secure user authentication**, and **database management**. It demonstrates how data science and web technologies can collaborate to build meaningful tools that serve a social purpose. The application uses Flask [1] for backend processing, SQLite [6] as its database engine, and Bootstrap [8] for a responsive user interface, ensuring that the system is scalable, lightweight, and easy to deploy.

In summary, **MEDI-AI** serves as both a personal health assistant and a health awareness platform. It not only guides users with intelligent medical recommendations but also empowers them to connect, learn, and contribute to a growing health-conscious community.

1.5 State of the Art and Possible Applications

The advancement of Artificial Intelligence (AI), particularly in the field of healthcare, has led to the development of intelligent systems that support diagnostics, treatment planning, and health monitoring. The current state of the art includes AI-powered systems such as IBM Watson Health, Google's DeepMind for medical imaging, and mobile applications like Ada Health, which analyze user symptoms to suggest possible conditions. These systems leverage machine learning models trained on vast medical datasets and employ Natural Language Processing (NLP), predictive analytics, and knowledge graphs to offer insights.

However, most of these platforms are either research-focused, commercially restricted, or targeted at institutional use, making them inaccessible for common users in developing regions. Moreover, they often focus on disease prediction alone, without integrating lifestyle guidance or community engagement features.

MEDI-AI advances the state of the art by offering a lightweight, web-based, open-access platform that provides both intelligent medical recommendations and user-generated health content. Unlike standalone symptom checkers or fragmented health blogs, **MEDI-AI** combines the intelligence of a trained machine learning model (Support Vector Machine) with an interactive platform for health-related blogging. This hybrid design enhances both individual and social dimensions of health awareness.

The system allows users to input symptoms and receive predictions with associated recommendations like medications, workouts, diets, and precautions—all derived from verified datasets. At the same time, users can read and publish blogs, share wellness stories, and engage in health discussions—creating a virtual health support ecosystem.

The possible applications are:

- **Personal Health Assistant**: For users to get instant guidance on symptoms and lifestyle suggestions.
- **Awareness Campaigns**: NGOs or government programs can use the platform to promote health education in local languages.
- **Remote Healthcare Support**: Useful in rural areas lacking access to primary healthcare services.
- **Academic Use**: Can serve as a teaching aid in medical and data science courses.
- **Wellness Community Building**: Encourages peer learning, support, and public participation in health awareness.

1.6 Objectives of the Project

The primary objective of the **MEDI-AI** project is to design and develop a comprehensive, user-friendly web application that offers intelligent healthcare recommendations and enables community engagement through health-related blogs. The project aims to bridge the gap between early medical assistance and general public awareness by integrating machine learning with interactive web technologies.

The key objectives of the project are:

1. **Symptom-Based Disease Prediction**:

To build a machine learning model capable of analyzing user-reported symptoms and accurately predicting the most probable disease. The model should be trained on a structured dataset and optimized for high accuracy.

2. **Personalized Medical Recommendations:**

To provide users with relevant and tailored health advice in the form of:

- Suggested medications
- Dietary recommendations
- Recommended workouts
- Necessary precautions

These recommendations aim to guide users toward appropriate self-care and preventive health practices.

3. Health Blogging Platform:

To implement a fully functional blogging module where users can:

- Register and log in securely
- Write and publish blogs with optional image uploads
- Comment on, like, and dislike blogs
- Edit or delete their own posts

This feature promotes health awareness and encourages community-driven knowledge sharing.

4. Secure and Responsive Web Interface:

To develop a responsive and visually appealing front end using Bootstrap [8], ensuring compatibility across devices. The application must follow secure practices including password hashing, session handling, and file validation.

5. Scalability and Modularity:

To ensure that the application architecture supports future enhancements such as chatbot integration, multilingual support, or deployment on cloud platforms.

2. Methodology & Modules/ Software Requirement Specification & Project Designing

The development of the **MEDI-AI** system was carried out using a modular, step-by-step methodology that ensured proper division of work, seamless integration, and clarity in system design. The approach enabled the system to evolve through well-defined phases, from conceptualization to final implementation.

The project commenced with a **requirement gathering phase**, during which both functional and non-functional requirements were identified. The system needed to support core features like disease prediction from user-input symptoms, providing relevant health suggestions, user registration and login, and a full-fledged blogging system with comment and interaction capabilities.

- Once the requirements were analyzed, the system architecture was divided into major **functional modules**, each responsible for a specific task:
- **User Management Module**: Handles registration, login, session control, and access rights.
- **Symptom Input & Disease Prediction Module**: Collects user symptoms, encodes them into feature vectors, and predicts the disease using a trained machine learning model.
- **Recommendation Engine**: Fetches relevant medications, diets, workouts, and precautionary steps based on the predicted disease.
- **Blog Management Module**: Allows users to create, read, update, and delete blogs with image uploads.
- **Engagement System**: Manages comments, likes, and dislikes on blog posts.
- **Frontend/UI Module**: Renders a responsive, interactive interface for all users using Bootstrap [8] and Jinja2 templating.

The **Flask [1] web framework** was chosen for its simplicity and strong support for modular application structure. The backend used **Python [2]**, with **Pandas [4]**, **NumPy [5]**, and **Pickle** for handling ML operations. A **Support Vector Machine (SVM)** model was trained for disease prediction. **Label Encoding** is used to convert categorical disease labels into numeric form.

The database was managed using **SQLite [6]** integrated with SQLAlchemy [7] ORM, ensuring clean data handling and easy migrations. Frontend components were built using **HTML5**, **Bootstrap [8]**, and **JavaScript**, allowing dynamic updates and a smooth user experience.

Security considerations included hashed password storage, file validation for uploads,

and session-based authentication for protected routes.
Overall, the structured methodology led to the development of a robust, intelligent, and user-centric healthcare platform combining AI and community engagement.

2.1 Technology (Hardware and Software) Used

The development of the **MEDI-AI** project required a blend of essential hardware and powerful software tools to ensure accurate machine learning predictions, seamless web functionality, and efficient database handling. The application was designed to be lightweight and accessible, yet powerful enough to integrate AI-based healthcare features and dynamic user interaction.
A standard development setup was used, including a personal computer with sufficient processing capabilities and memory to handle full-stack development, dataset processing, and local testing of the machine learning model. The Flask [1] web server, SQLite [6] database, and trained machine learning model were run locally during the development and testing phases.
On the software front, modern technologies were adopted to build the client-side and server-side components. The machine learning model was trained using **Python [2] libraries** such as **Pandas [4]**, **NumPy [5]**, and **Scikit-learn [3]**. **Flask [1]** served as the backend framework to manage user authentication, routing, and communication between the frontend and backend.
Bootstrap5 [8], **HTML5**, **CSS**, and **Jinja2** were used to develop a responsive and user-friendly interface. File uploads and static content were handled securely through Flask [1]'s built-in utilities.
SQLite [6] was chosen as the database for its simplicity and ease of integration with Flask [1] via **SQLAlchemy [7] ORM**. Development was carried out using **Visual Studio Code**, and diagrams such as use case, ER, and DFDs were designed using **Draw.io**.

Hardware Requirements

Processor: Intel i5 or higher **RAM**: Minimum 8 GB
Storage: At least 2 GB free disk space
Display: 1366 × 768 resolution or above **Peripherals**: Standard keyboard and mouse

Software Requirements

Operating System: Windows 10/11, macOS, or Linux
Frontend: Jinja2 Templates, Bootstrap [8]5, HTML, CSS
Backend: Flask [1], Python [2], SQL Alchemy, Machine Learning, Jupiter Notebook
Database: SQL Lite3

Development Tools and Libraries

Editors & Version Control: Visual Studio Code, Git, GitHub
Libraries: NumPy [5], Pandas, Scikit-learn [3]

2.2 Project Planning

The development of the **MEDI-AI** project followed a structured and time-bound planning process. The project was divided into multiple phases, including requirement analysis, data preparation, model training, backend integration, frontend design, module testing, and final deployment. Each phase was scheduled using a Gantt chart for better tracking and management. The planning ensured parallel development of the machine learning model and web interface, reducing overall development time.

Tasks were assigned specific timelines with buffer periods for review and debugging. Version control was maintained using Git, and regular milestones were set to evaluate progress. Weekly meetings and feedback loops helped keep the project aligned with its objectives and schedule.

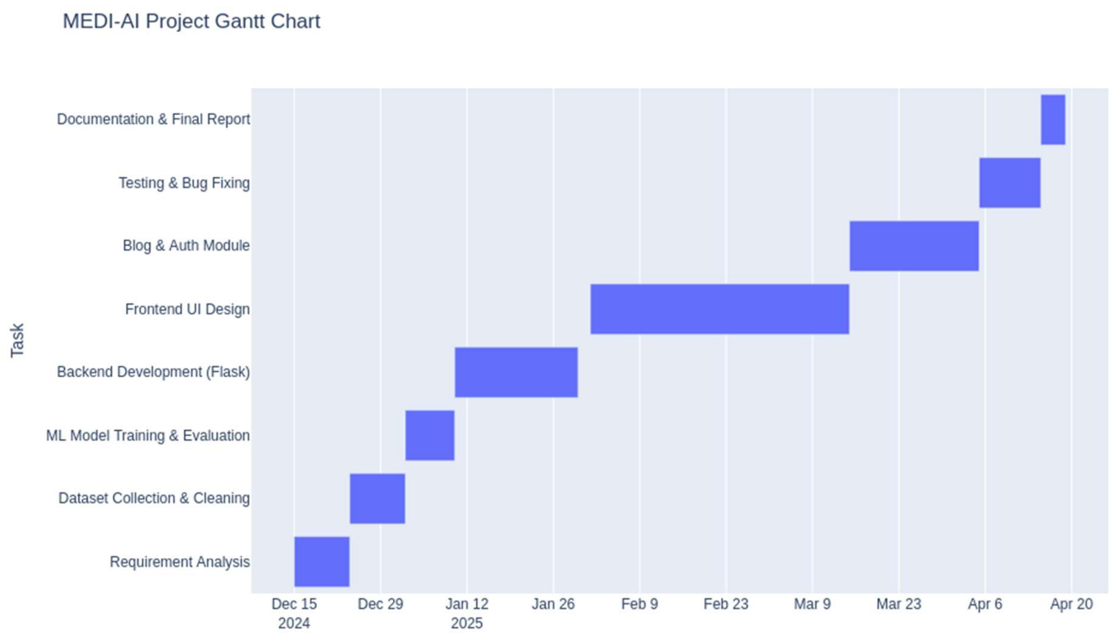


Figure 1 - Gantt Chart

2.3 Feasibility Study

A feasibility study is essential for evaluating whether the proposed system can be developed, deployed, and maintained effectively within available resources and constraints. For the **MEDI-AI** project, the feasibility analysis includes an assessment of technical, operational, and economic aspects to ensure the solution is practical, scalable, and sustainable.

1-Technical Feasibility

Technical feasibility determines whether the project's requirements can be met using available technologies and whether the development team has the necessary expertise. For MEDI-AI, the technology stack has been carefully selected to ensure smooth implementation and ease of future enhancements.

- **Backend Development**: The application is built using **Flask [1] (Python [2])**, a lightweight web framework ideal for scalable and modular web applications. It allows seamless integration of ML models and database operations.
- **Frontend Development**: The user interface is developed using **HTML5**, **CSS**, **Bootstrap [8] 5**, and **Jinja2** for responsive design and server-side templating. The layout adapts to different devices and provides a clean, intuitive user experience.
- **Machine Learning Integration**: A **Support Vector Machine (SVM)** model, trained using **Pandas**, **NumPy [5]**, and **Scikit-learn [3]**, is used to predict diseases based on symptom input. The model is serialized using **Pickle** for real-time use.
- **Database**: **SQLite [6]** is used for data storage due to its simplicity and suitability for small-scale applications. It integrates well with Flask [1] via **SQLAlchemy [7] ORM** for efficient data handling.
- **Security**: **Werkzeug's** password hashing and Flask [1] session handling ensure secure user authentication. Input validations and file restrictions are applied during uploads.

Given the use of open-source, community-supported, and well-documented technologies, the project is technically feasible with a strong potential for future scalability.

2-Operational Feasibility

Operational feasibility assesses whether the system is user-friendly and maintainable. MEDI-AI is designed to function efficiently for all stakeholders, including regular users and administrators.

- **User Interface**: The UI is clean and accessible, allowing users to easily select symptoms, get health recommendations, and interact with health blogs.
- **Disease Prediction Workflow**: Users can select symptoms via a button-based interface. Once submitted, the system predicts the disease and displays

medications, diet, precautions, and workout suggestions—all dynamically fetched from curated datasets.

- **Health Blogging Platform**: Users can register, write blogs with images, edit or delete posts, comment on others' blogs, and engage via likes/dislikes. These features create a health-focused community.
- **System Maintenance**: The application is modular and follows MVC architecture, allowing individual components to be updated without affecting others. Flask [1]'s simplicity and Python [2]'s flexibility further ease maintenance.
- **Scalability**: Though designed for local use initially, MEDI-AI can be easily hosted on cloud services (like Render, Heroku, or AWS) to support a larger user base in future versions.

The application structure, minimal system requirements, and intuitive workflow make operational feasibility very high.

3-Economic Feasibility

Economic feasibility focuses on development, deployment, and long-term maintenance costs.

- **Development Costs**: All major technologies used—Flask [1], Python [2], SQLite [6], Bootstrap [8], Jinja2—are open-source and free. This significantly reduces upfront costs.
- **Tools Used**: Development was done using **Visual Studio Code**, with version control handled via **Git** and **GitHub**, both of which are free tools. Diagramming tools like **Draw.io** were used for ER diagrams, DFDs, and use case modeling.
- **Media and Hosting**: During development, media files are stored locally. For production, inexpensive cloud storage options like **Firebase** or **Cloudinary** can be used. Hosting can be done on platforms like **Render**, **Python [2]Anywhere**, or **Heroku**, many of which offer generous free tiers.
- **Monetization Potential**: While this project is academic, it has future potential for monetization by offering premium features, advertisements, or partnering with healthcare providers.
- **Return on Investment (ROI)**: Given that the entire system is built with free and scalable technologies, and requires minimal infrastructure, the ROI is highly favorable for small startups, NGOs, or educational institutes aiming to deploy such systems.

In conclusion, the MEDI-AI project is technically sound, operationally efficient, and economically viable with a strong foundation for future scaling and enhancement.

2.4 Design of the System

1. Use Case Diagram

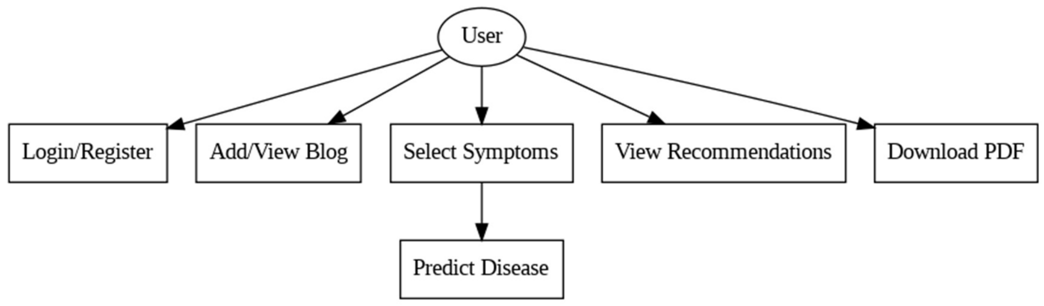


Figure 2- Use Case Diagram

2. Use Case Description

The MEDI-AI platform integrates intelligent medical assistance with a health-focused community blog system. The application supports two main user types: Patients and Doctors. Below are the major use cases and their functionalities:

1. **User Registration & Authentication:**
   - Patients can register with personal details (name, email, mobile).
   - Secure login is required to access personalized features.
   - Session-based access control ensures security and role-based navigation.
2. **Symptom Checker & Disease Prediction: x**
   - Users select symptoms from an interactive UI.
   - The system converts symptoms into feature vectors.
   - A trained SVM (Support Vector Machine) model predicts the most probable disease.
3. **Health Recommendations:**
   After prediction, the system provides:
   - **Precautions** – suggested habits and lifestyle changes
   - **Medications** – AI-recommended drug names
   - **Diet Plans** – condition-specific dietary advice
   - **Workouts** – physical exercises for recovery
4. **Blog Management & Community Engagement:**
   Registered users can:
   - **Create and Publish Blogs** – Share health journeys, recovery tips, or medical awareness.
   - **Edit or Delete Blogs** – Users maintain full control over their content.
   - **View and Search Blogs** – By keyword, title, or related disease.

- **Interact with Blogs** – Like, dislike, or comment to promote discussion and peer engagement.

5. **System Functions:**
   - **Database Management** – Manages users, blogs, comments, likes/dislikes, symptoms, predictions, and recommendations.
   - **Machine Learning Integration** – Trained SVM model for accurate disease prediction.
   - **PDF Report Generation** – Summarized and downloadable medical advice after prediction.
   - **Media Handling** – Handles blog image uploads with validation and secure storage.

6. **Logout:** Logging out of the system removes the user's active session and ensures data security. After clicking the logout button, a confirmation message appears, indicating the successful logout. The user is then redirected to the login page, where they can re-enter their credentials to gain access again. This feature ensures that the system remains secure by preventing unauthorized access to sensitive user data after a session ends.

Key Features

- **Interface** – simple symptom selection, blog interaction
- **Security** – password hashing, role-based access
- **Responsiveness** – mobile/tablet/desktop compatibility

3. Entity Relationship Diagram

The **Entity-Relationship (ER) diagram** of the **MEDI-AI** platform illustrates the key entities involved in delivering intelligent medical recommendations and managing community-driven blog interactions. It defines how data flows between users, symptoms, predictions, recommendations, blogs, and other interactive elements of the system.

The core entities in the system include:

- **Users** – Stores information such as name, username, email, password (hashed), role (Patient/Doctor), and registration date.
- **Blogs** – Represents health-related articles written by users, with fields such as title, content, image, timestamps, and foreign key references to the author.
- **Comments** – Linked to blog posts and users, this entity stores textual feedback along with timestamps.
- **Like Dislike** – Tracks user interactions on blog posts, ensuring each user can like or dislike a blog only once.
- **Predictions** – Stores results of AI-based symptom analysis, linking the user to a predicted disease.

- **Symptoms** – A predefined list of symptoms from which users select their inputs.
- **Recommendations** – Contains disease-specific information such as medications, diet, workout routines, and precautions, fetched dynamically based on prediction.

Each entity is connected via appropriate **one-to-many** or **many-to-one** relationships. For example:

- One user can write many blogs (1:N).
- One blog can have many comments and reactions (1:N).
- One prediction is associated with one user (1:1), but a user can generate multiple predictions over time (1:N).

These relationships ensure **data consistency**, **referential integrity**, and **optimized querying** for both user-facing features and backend operations. The ER model serves as a blueprint for implementing database tables using **SQLite [6]** and **SQLAlchemy [7] ORM** in Flask [1].
A complete breakdown of attributes and relational keys is provided in the **Logical Database Design** section.

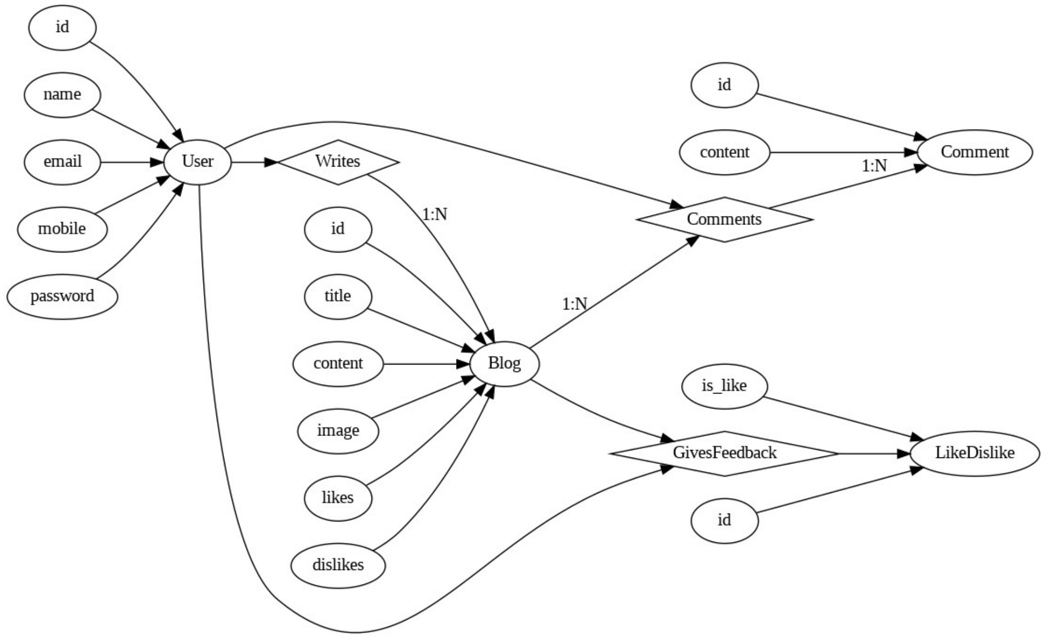


Figure 3 - ER Diagram

4. Data Flow Diagrams (DFDs)

4.1. Context Diagram or Level-0 DFD

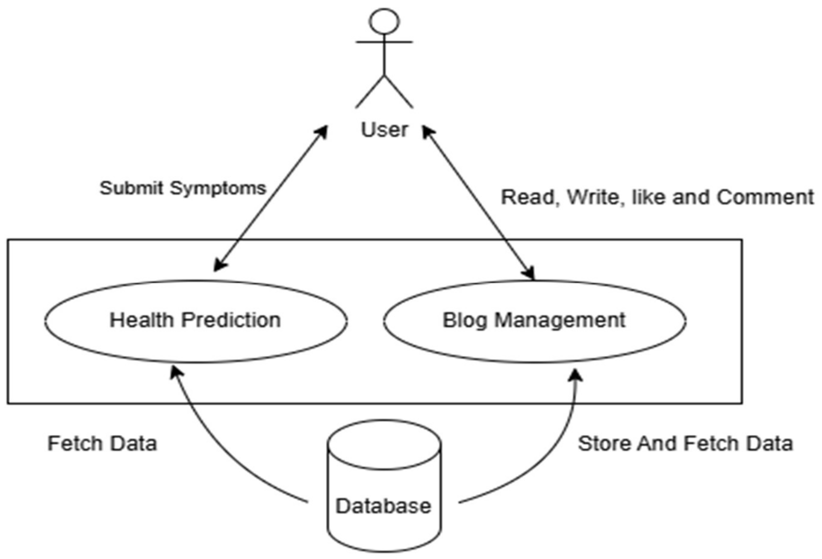


Figure 4- Level 0 DFD

4.2. Level-1 DFD

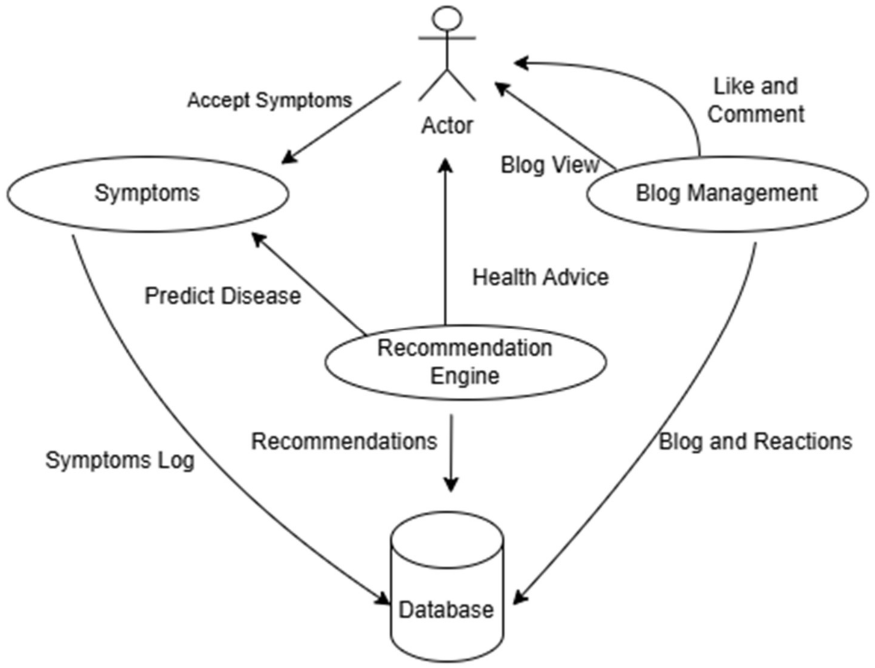


Figure 5 - Level-1 DFD

4.3. Level-2 DFD

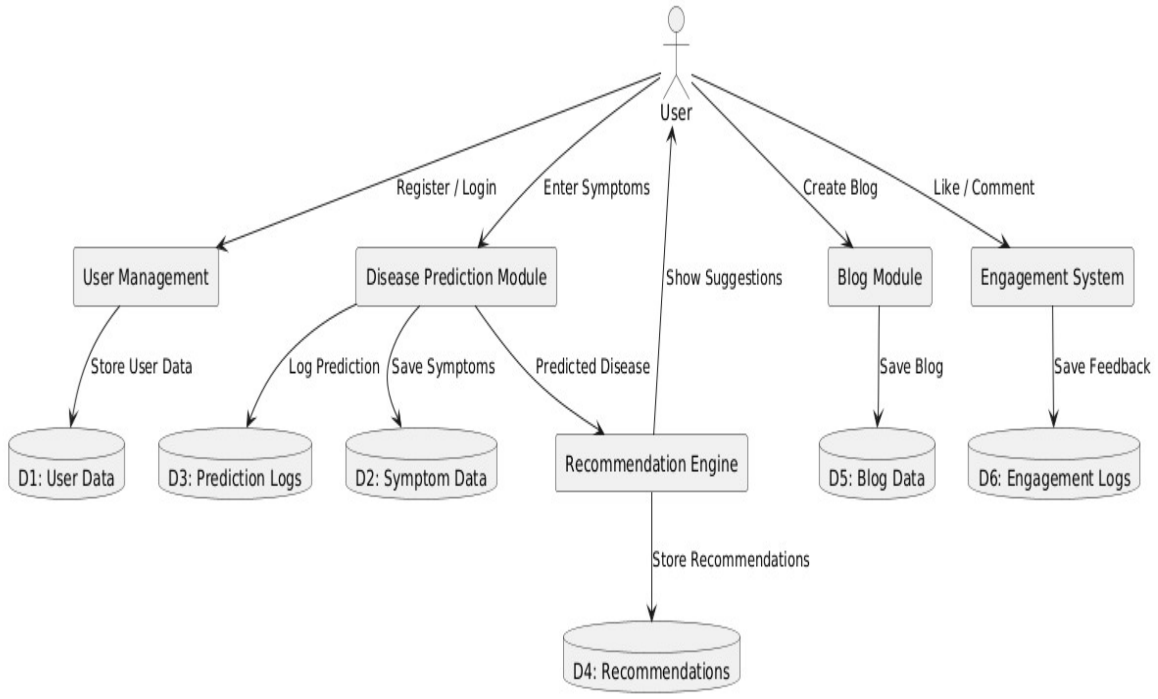


Figure 6 - Level-2 DFD

5. *Logical Database Design*

The **logical database design** for the **MEDI-AI** platform defines how data is structured, related, and accessed within the system to ensure efficient storage, data integrity, and reliable interactions across modules. The database schema was designed using **SQLite [6]** in combination with **SQLAlchemy [7] ORM**, offering a lightweight and flexible solution that integrates seamlessly with Flask [1].

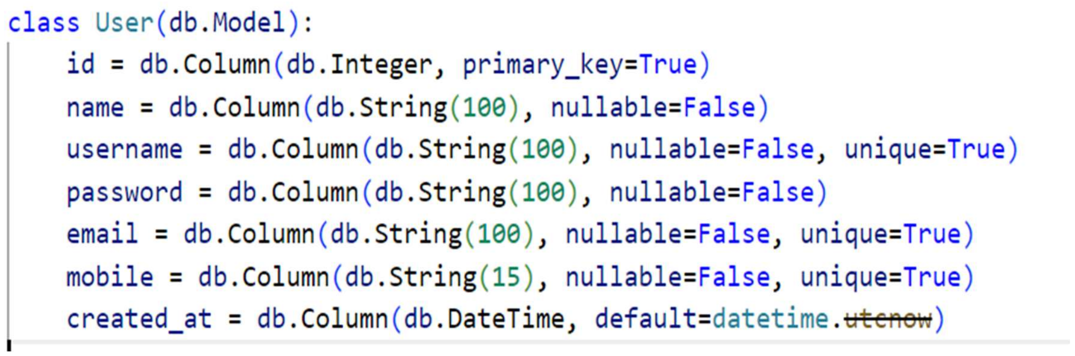
The platform supports two core functionalities: **AI-based disease prediction** and a **health blog system**. These functionalities drive the design of the underlying data model, which includes entities like **users**, **blogs**, **comments**, **symptoms**, **predictions**, **recommendations**, and **interactions (likes/dislikes)**.

1. User Table

The User entity stores the details of every registered individual, whether a patient or a doctor. Key fields include:

- id (Primary Key): Uniquely identifies the user.
- name, username, email, mobile: Personal identifiers with unique constraints.
- password: Stored securely in a hashed format.
- created_at: Timestamp for registration.

This table allows role-based access control by distinguishing between normal users (patients) and admins/doctors, which is essential for enabling doctor-specific dashboards.



```
class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(100), nullable=False)
    username = db.Column(db.String(100), nullable=False, unique=True)
    password = db.Column(db.String(100), nullable=False)
    email = db.Column(db.String(100), nullable=False, unique=True)
    mobile = db.Column(db.String(15), nullable=False, unique=True)
    created_at = db.Column(db.DateTime, default=datetime.~~utcnow~~)
```

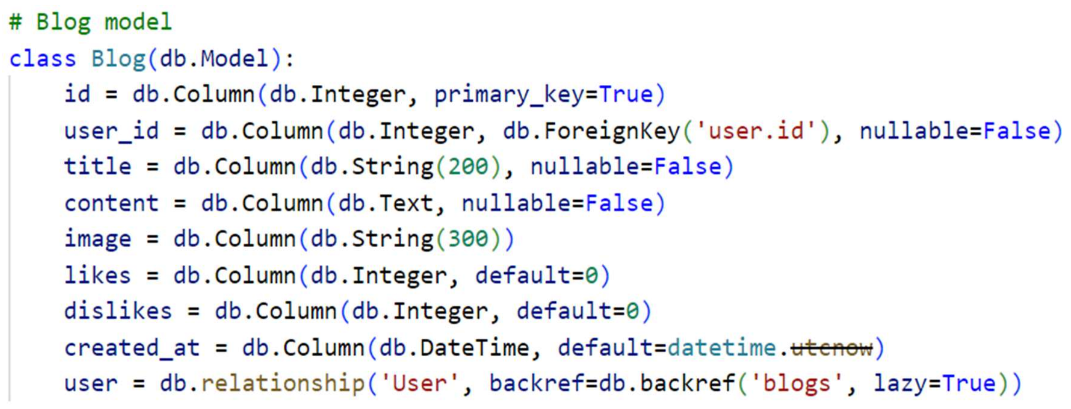
Figure 7- User Model

2. Blog Table

The Blog table manages all blog entries created by users. It includes:

- id (Primary Key)
- user_id (Foreign Key): Links to the User who authored the blog.
- title, content, image: The content of the blog.
- likes, dislikes: Integer fields for engagement tracking.
- created_at: Blog submission timestamp.

This design allows multiple blogs per user and forms the basis of the user engagement system.



```
# Blog model
class Blog(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'), nullable=False)
    title = db.Column(db.String(200), nullable=False)
    content = db.Column(db.Text, nullable=False)
    image = db.Column(db.String(300))
    likes = db.Column(db.Integer, default=0)
    dislikes = db.Column(db.Integer, default=0)
    created_at = db.Column(db.DateTime, default=datetime.~~utcnow~~)
    user = db.relationship('User', backref=db.backref('blogs', lazy=True))
```

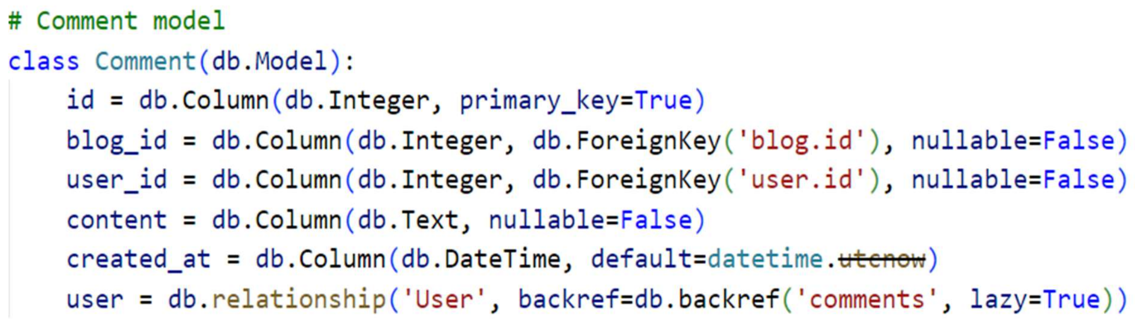
Figure 8- Blog Model

3. Comment Table

The Comment table records user feedback on blogs. It contains:

- id, content, created_at
- Foreign keys to both user_id and blog_id, establishing a many-to-one relationship between users/blogs and comments.

This setup allows each blog to have many comments while maintaining traceability to the comment's author.



```
# Comment model
class Comment(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    blog_id = db.Column(db.Integer, db.ForeignKey('blog.id'), nullable=False)
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'), nullable=False)
    content = db.Column(db.Text, nullable=False)
    created_at = db.Column(db.DateTime, default=datetime.~~utcnow~~)
    user = db.relationship('User', backref=db.backref('comments', lazy=True))
```

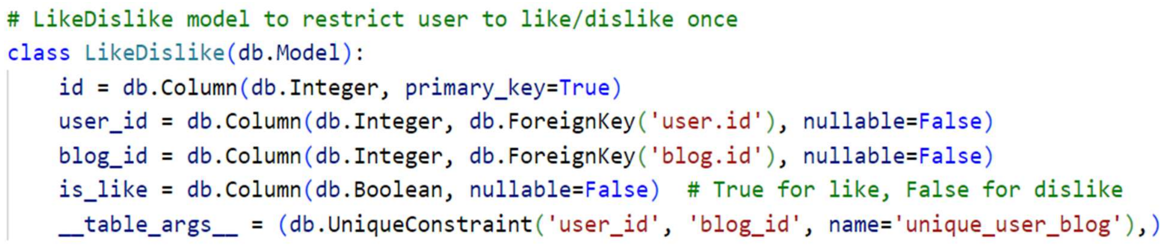
Figure 9 - Comment Model

4. LikeDislike Table

To ensure users cannot like or dislike a blog more than once, the LikeDislike table was created with a **composite unique constraint**:

- user_id, blog_id: Foreign keys.
- is_like: Boolean to indicate like (True) or dislike (False).

This structure efficiently manages user interactions and prevents abuse or redundancy.



```python
# LikeDislike model to restrict user to like/dislike once
class LikeDislike(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'), nullable=False)
    blog_id = db.Column(db.Integer, db.ForeignKey('blog.id'), nullable=False)
    is_like = db.Column(db.Boolean, nullable=False)  # True for like, False for dislike
    __table_args__ = (db.UniqueConstraint('user_id', 'blog_id', name='unique_user_blog'),)
```

Figure 10- LikeDislike Model

5. Prediction & Symptom Tables

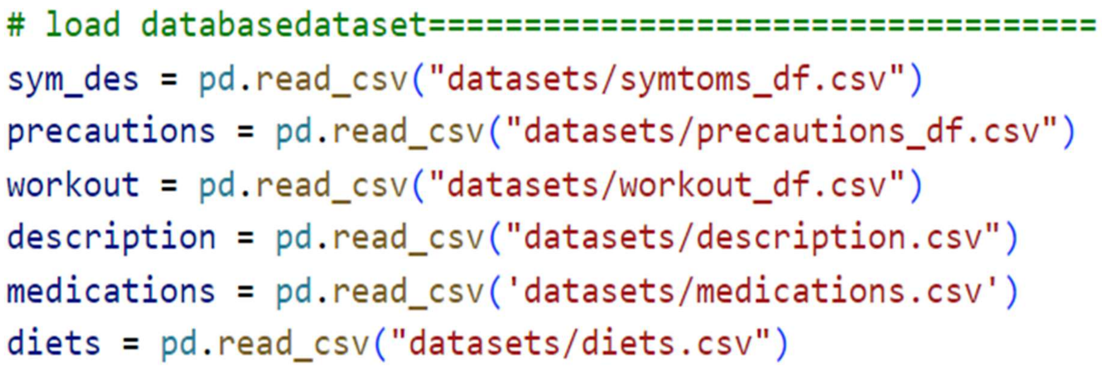
For the disease prediction system:

Symptom data is mapped using a fixed dictionary (symptoms_dict) where each symptom is represented as a feature index for ML input.

After a prediction, results are used to query:

- description.csv: Explanation of the disease
- precautions_df.csv: Suggested preventive measures
- medications.csv: General drug suggestions
- diets.csv: Diet plan recommendations
- workout_df.csv: Exercise suggestions

These files simulate database tables and could be migrated to actual relational tables in future versions.



```python
# load databasedataset===================================
sym_des = pd.read_csv("datasets/symtoms_df.csv")
precautions = pd.read_csv("datasets/precautions_df.csv")
workout = pd.read_csv("datasets/workout_df.csv")
description = pd.read_csv("datasets/description.csv")
medications = pd.read_csv('datasets/medications.csv')
diets = pd.read_csv("datasets/diets.csv")
```

Figure 11- Dataset For Disease Prediction

6. Recommendations Logic

Recommendations within the MEDI-AI platform are generated dynamically in real-time, ensuring users receive relevant and personalized guidance immediately after a disease prediction. Rather than being stored as static records, these suggestions are derived through intelligent helper functions that act as the system's internal matching engine.

Normalization and Relationships

- The schema follows normalization rules to reduce redundancy:

- One-to-many relationship between User and Blog, Comment, and LikeDislike.
- Blogs are normalized by linking images and content separately.
- Comments and likes are tracked with minimal duplication.

Scalability and Integrity

The design supports future growth, such as:

- Adding a Role column in the User table.
- Storing patient health history or lab reports.
- Migrating dataset-based recommendations to relational models.

6. High Level Structure Diagram

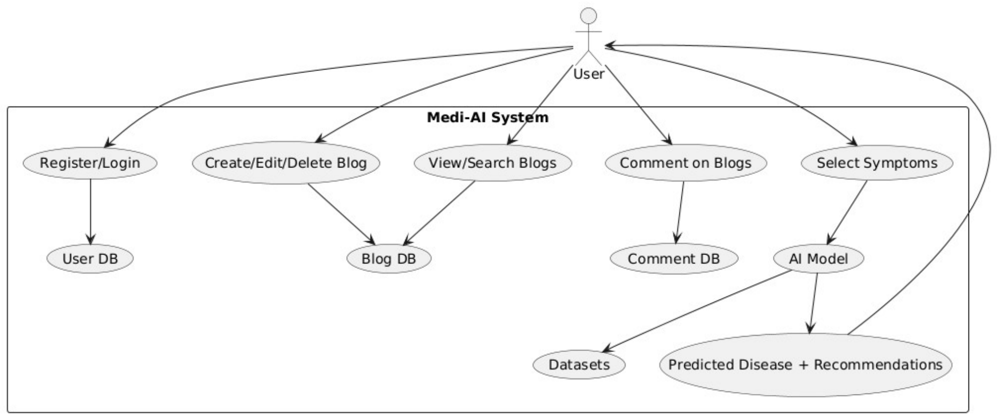


Figure 12 - High Level Structure Diagram

The **High-Level** Structure Diagram of the **MEDI-AI** system illustrates the major functional components and their interactions within the platform. It provides a bird's-eye view of how different modules—such as user interfaces, the AI-based disease prediction engine, the blogging system, and the recommendation engine—work together in a seamless and integrated manner.

At the core of the system is a Flask-based web application that handles user authentication, symptom input, blog management, and overall routing. When a user selects symptoms, these inputs are processed by the Symptom Encoding module and passed to a pre-trained **Support Vector Machine (SVM)** model, which predicts the most probable disease.

Based on the prediction, the system queries structured datasets to generate personalized health recommendations, including medications, diets, workouts, and precautions. The results are dynamically rendered and displayed through a responsive front end, with options for PDF report generation.

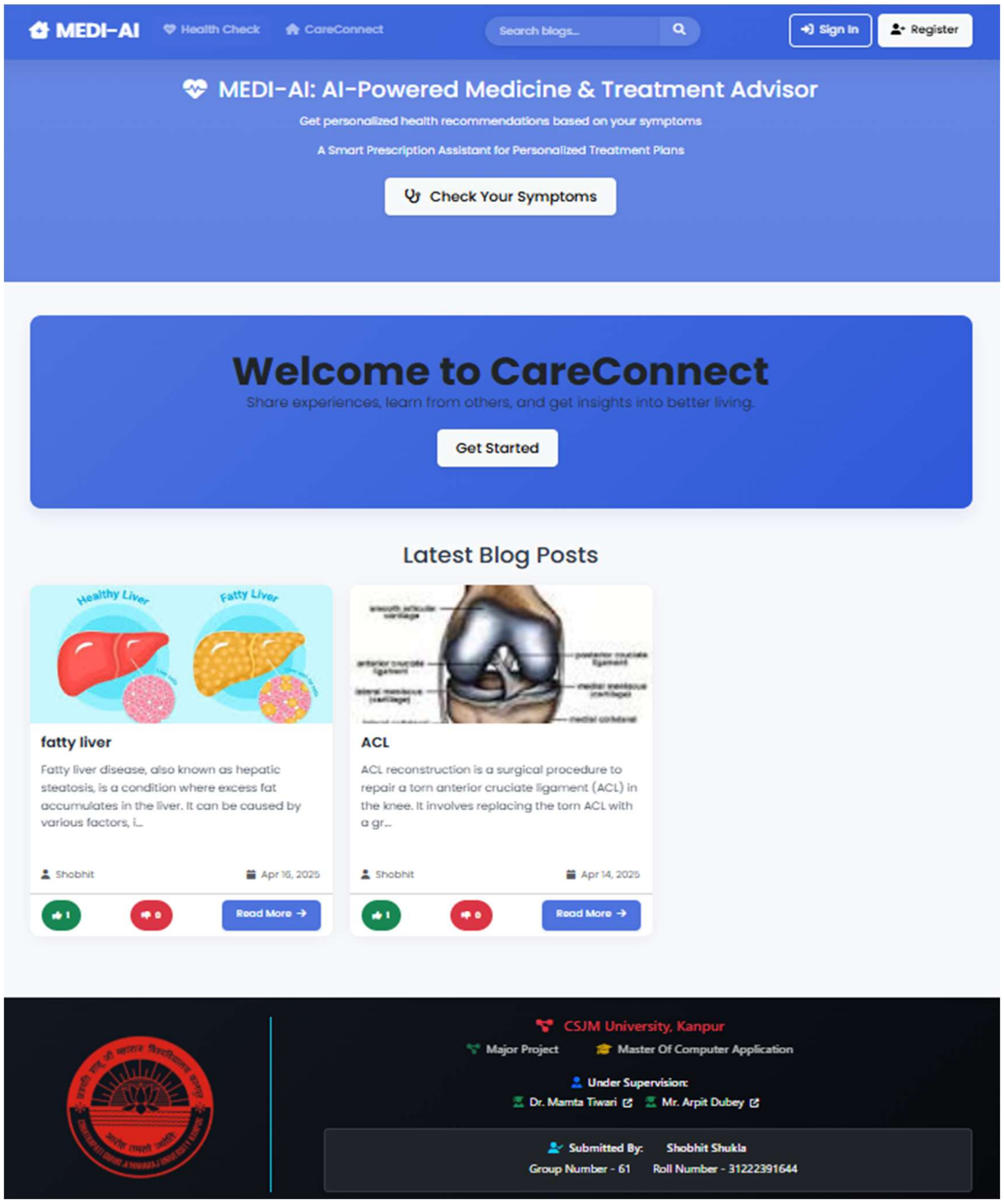
Simultaneously, users can engage with a health blogging module, allowing them to share experiences, write or read blogs, and interact through likes and comments. All data is stored and managed in a centralized **SQLite database**, ensuring secure and consistent operations.

This high-level structure emphasizes the system's modularity, scalability, and ability to blend AI-driven health support with community-based engagement.

3. Implementation

3.1 Screen Shots & Functionality

1. Graphical User Interface



MEDI-AI: AI-Powered Medicine & Treatment Advisor

Get personalized health recommendations based on your symptoms

A Smart Prescription Assistant for Personalized Treatment Plans

Welcome to CareConnect

Share experiences, learn from others, and get insights into better living.

Latest Blog Posts

fatty liver

Fatty liver disease, also known as hepatic steatosis, is a condition where excess fat accumulates in the liver. It can be caused by various factors, i...

Shobhit Apr 16, 2025

ACL

ACL reconstruction is a surgical procedure to repair a torn anterior cruciate ligament (ACL) in the knee. It involves replacing the torn ACL with a gr...

Shobhit Apr 14, 2025

CSJM University, Kanpur

Major Project Master Of Computer Application

Under Supervision:

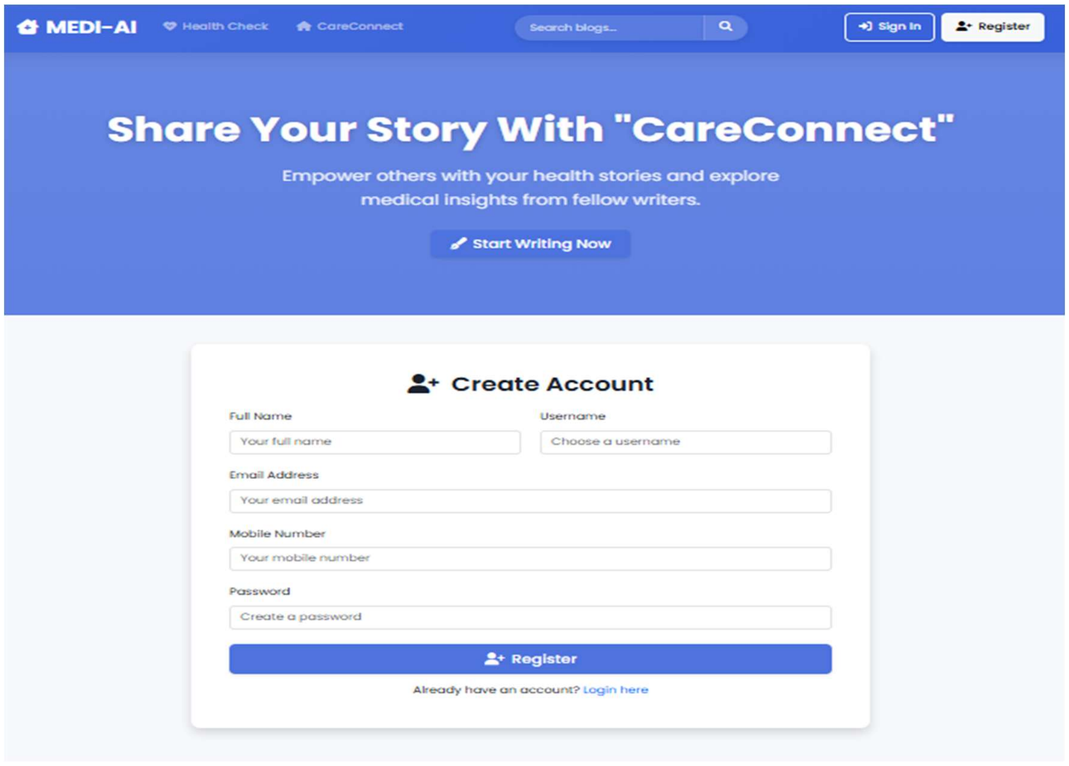
Dr. Mamta Tiwari Mr. Arpit Dubey

Submitted By: Shobhit Shukla

Group Number - 61 Roll Number - 31222391644

Figure 13 – MEDI-AI Home Page

2. Other relevant screenshots



Share Your Story With "CareConnect"

Empower others with your health stories and explore medical insights from fellow writers.

Create Account

Full Name: Your full name

Username: Choose a username

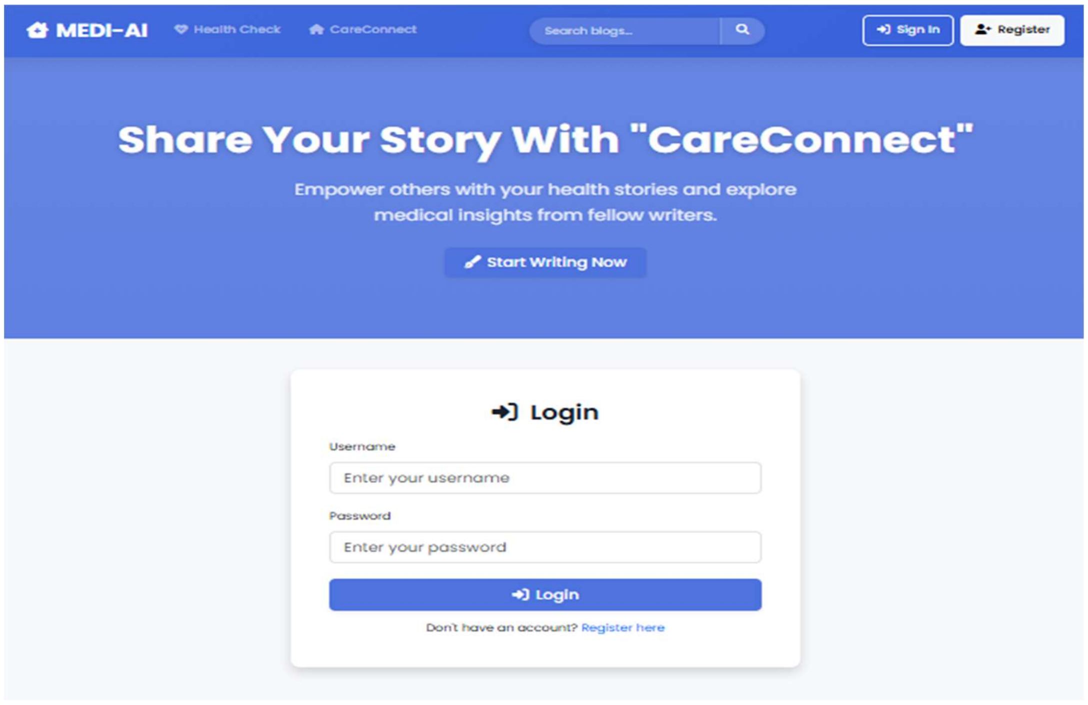
Email Address: Your email address

Mobile Number: Your mobile number

Password: Create a password

Already have an account? Login here

Figure 14 – User Registration Page



Share Your Story With "CareConnect"

Empower others with your health stories and explore medical insights from fellow writers.

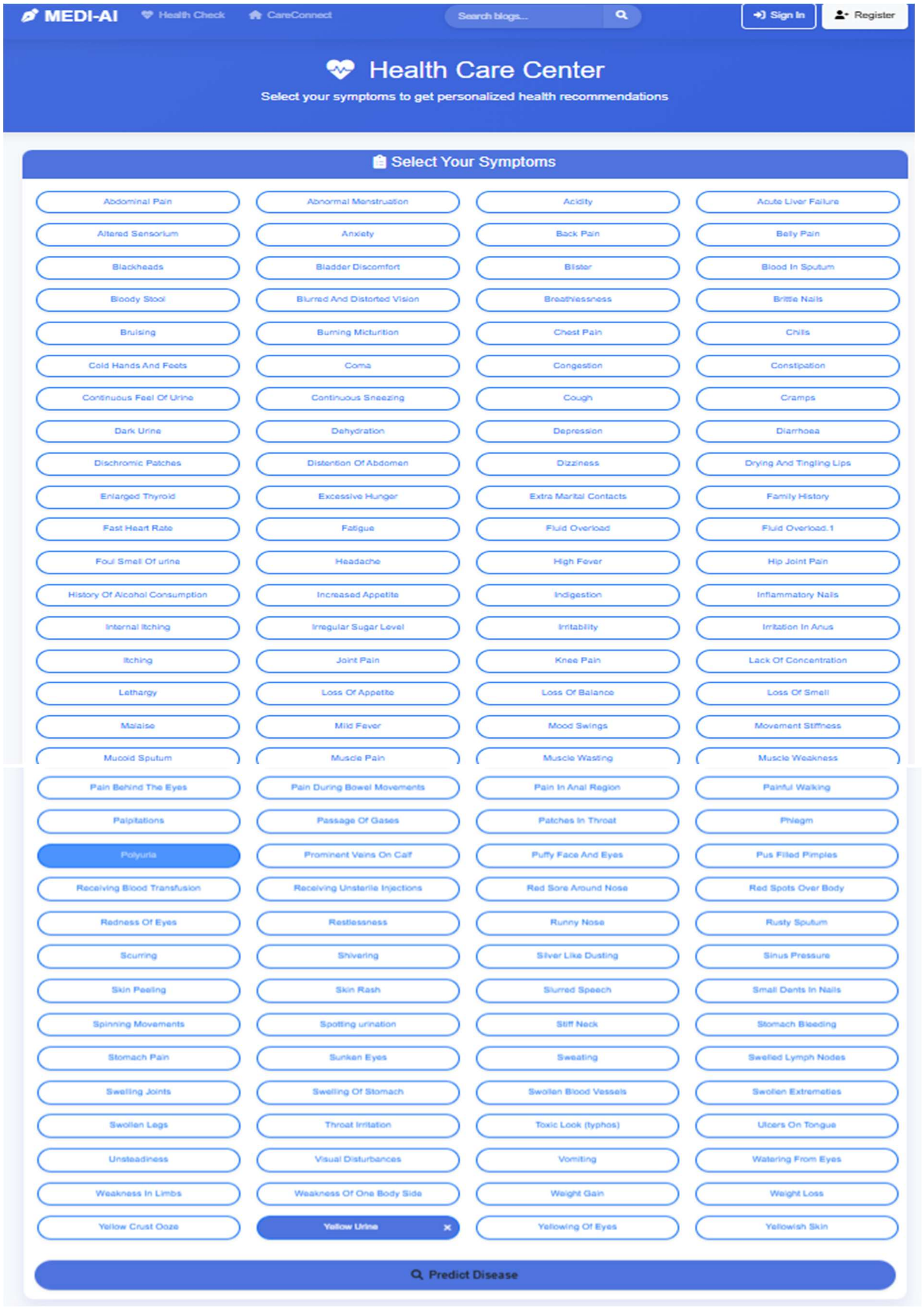
Login

Username: Enter your username

Password: Enter your password

Don't have an account? Register here

Figure 15 - Login Page


MEDI-AI Health Check CareConnect Search blogs... Sign In Register

Health Care Center

Select your symptoms to get personalized health recommendations

Select Your Symptoms

Abdominal Pain	Abnormal Menstruation	Acidity	Acute Liver Failure
Altered Sensorium	Anxiety	Back Pain	Belly Pain
Blackheads	Bladder Discomfort	Blister	Blood In Sputum
Bloody Stool	Blurred And Distorted Vision	Breathlessness	Brittle Nails
Bruising	Burning Micturition	Chest Pain	Chills
Cold Hands And Feets	Coma	Congestion	Constipation
Continuous Feel Of Urine	Continuous Sneezing	Cough	Cramps
Dark Urine	Dehydration	Depression	Diarrhoea
Dischromic Patches	Distention Of Abdomen	Dizziness	Drying And Tingling Lips
Enlarged Thyroid	Excessive Hunger	Extra Marital Contacts	Family History
Fast Heart Rate	Fatigue	Fluid Overload	Fluid Overload.1
Foul Smell Of urine	Headache	High Fever	Hip Joint Pain
History Of Alcohol Consumption	Increased Appetite	Indigestion	Inflammatory Nails
Internal Itching	Irregular Sugar Level	Irritability	Irritation In Anus
Itching	Joint Pain	Knee Pain	Lack Of Concentration
Lethargy	Loss Of Appetite	Loss Of Balance	Loss Of Smell
Malaise	Mild Fever	Mood Swings	Movement Stiffness
Mucoid Sputum	Muscle Pain	Muscle Wasting	Muscle Weakness
Pain Behind The Eyes	Pain During Bowel Movements	Pain In Anal Region	Painful Walking
Palpitations	Passage Of Gases	Patches In Throat	Phlegm
Polyuria	Prominent Veins On Calf	Puffy Face And Eyes	Pus Filled Pimples
Receiving Blood Transfusion	Receiving Unsterile Injections	Red Sore Around Nose	Red Spots Over Body
Redness Of Eyes	Restlessness	Runny Nose	Rusty Sputum
Scurring	Shivering	Silver Like Dusting	Sinus Pressure
Skin Peeling	Skin Rash	Slurred Speech	Small Dents In Nails
Spinning Movements	Spotting urination	Stiff Neck	Stomach Bleeding
Stomach Pain	Sunken Eyes	Sweating	Swelled Lymph Nodes
Swelling Joints	Swelling Of Stomach	Swollen Blood Vessels	Swollen Extremeties
Swollen Legs	Throat Irritation	Toxic Look (typhos)	Ulcers On Tongue
Unsteadiness	Visual Disturbances	Vomiting	Watering From Eyes
Weakness In Limbs	Weakness Of One Body Side	Weight Gain	Weight Loss
Yellow Crust Ooze	Yellow Urine ×	Yellowing Of Eyes	Yellowish Skin

Predict Disease

Figure 16 - Health Care Page for Prediction

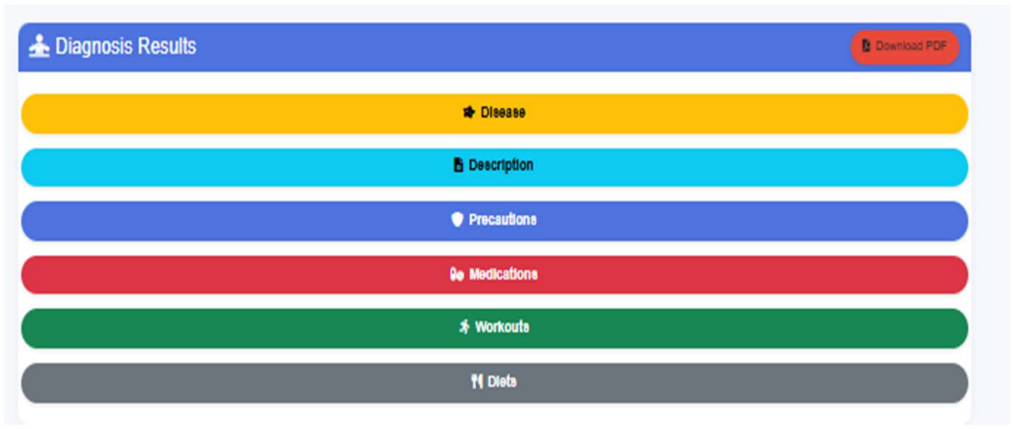
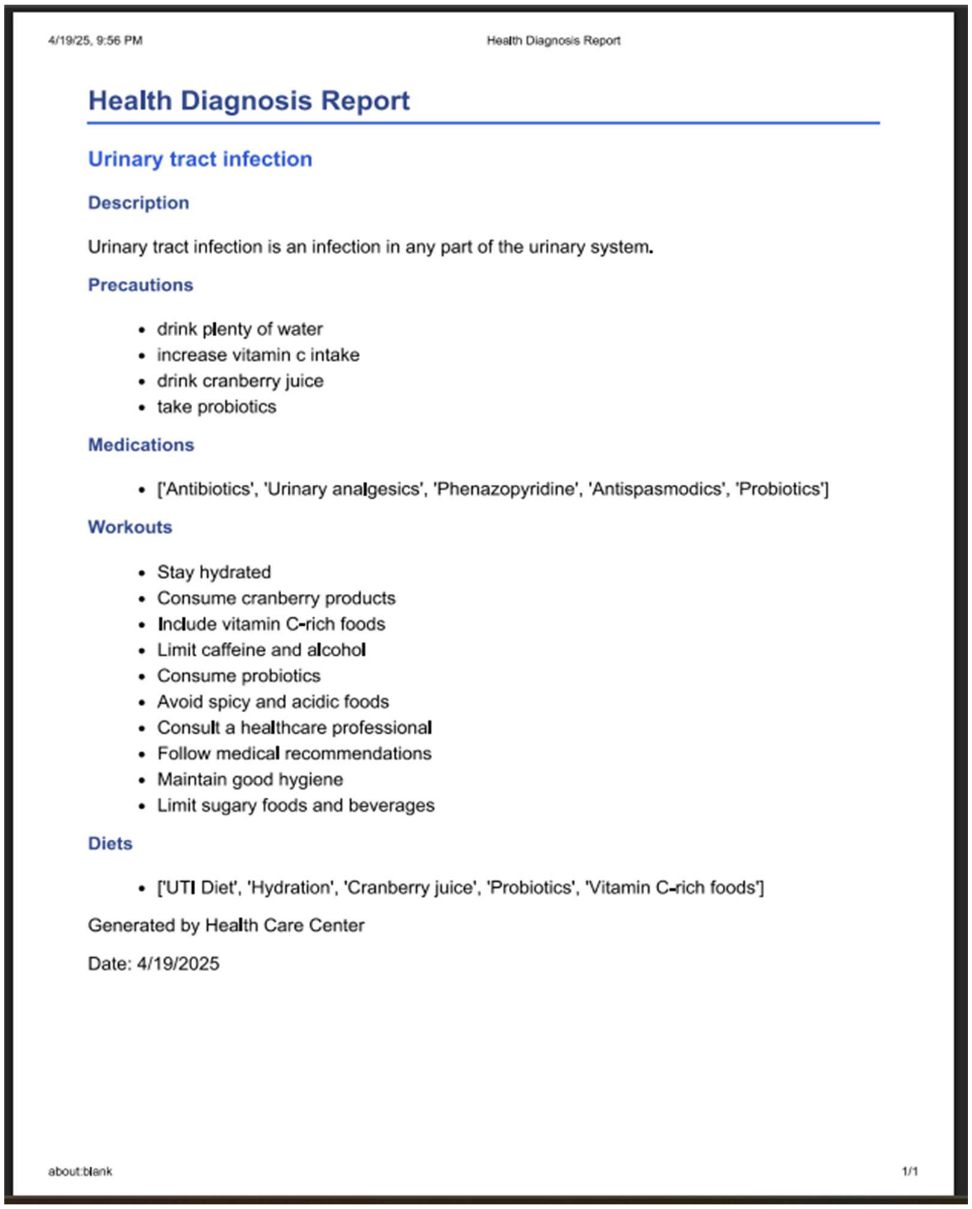


Figure 17 – Diagnosis Result Based On Prediction



4/19/25, 9:56 PM Health Diagnosis Report

Health Diagnosis Report

Urinary tract infection

Description

Urinary tract infection is an infection in any part of the urinary system.

Precautions

- drink plenty of water
- increase vitamin c intake
- drink cranberry juice
- take probiotics

Medications

- ['Antibiotics', 'Urinary analgesics', 'Phenazopyridine', 'Antispasmodics', 'Probiotics']

Workouts

- Stay hydrated
- Consume cranberry products
- Include vitamin C-rich foods
- Limit caffeine and alcohol
- Consume probiotics
- Avoid spicy and acidic foods
- Consult a healthcare professional
- Follow medical recommendations
- Maintain good hygiene
- Limit sugary foods and beverages

Diets

- ['UTI Diet', 'Hydration', 'Cranberry juice', 'Probiotics', 'Vitamin C-rich foods']

Generated by Health Care Center

Date: 4/19/2025

about:blank 1/1

Figure 18– Download Report Format

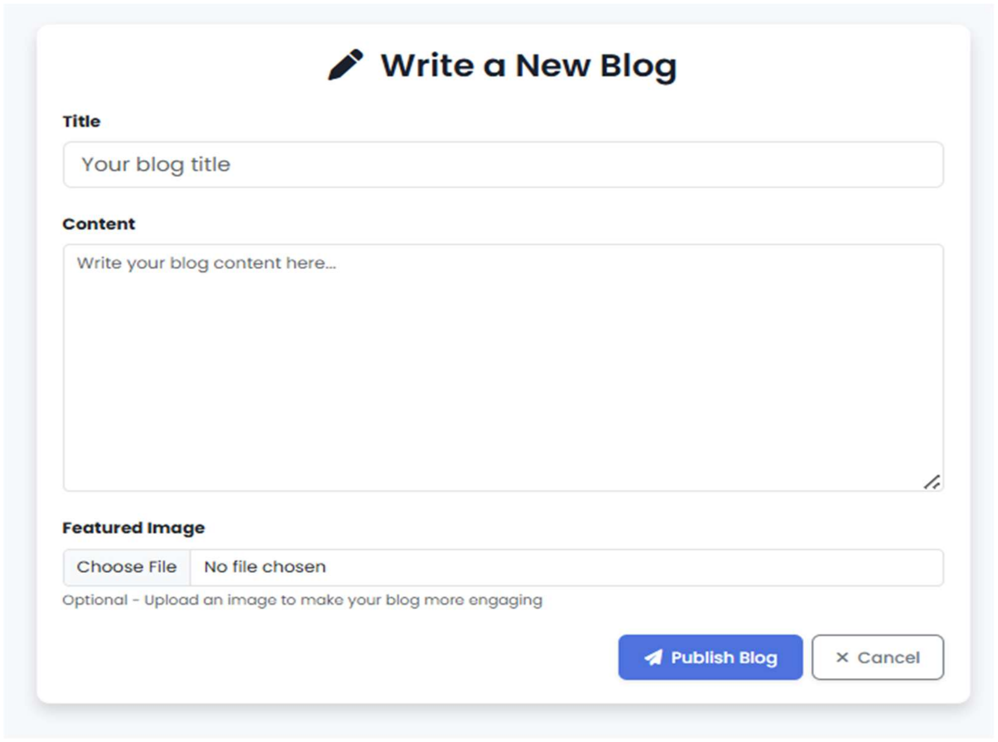
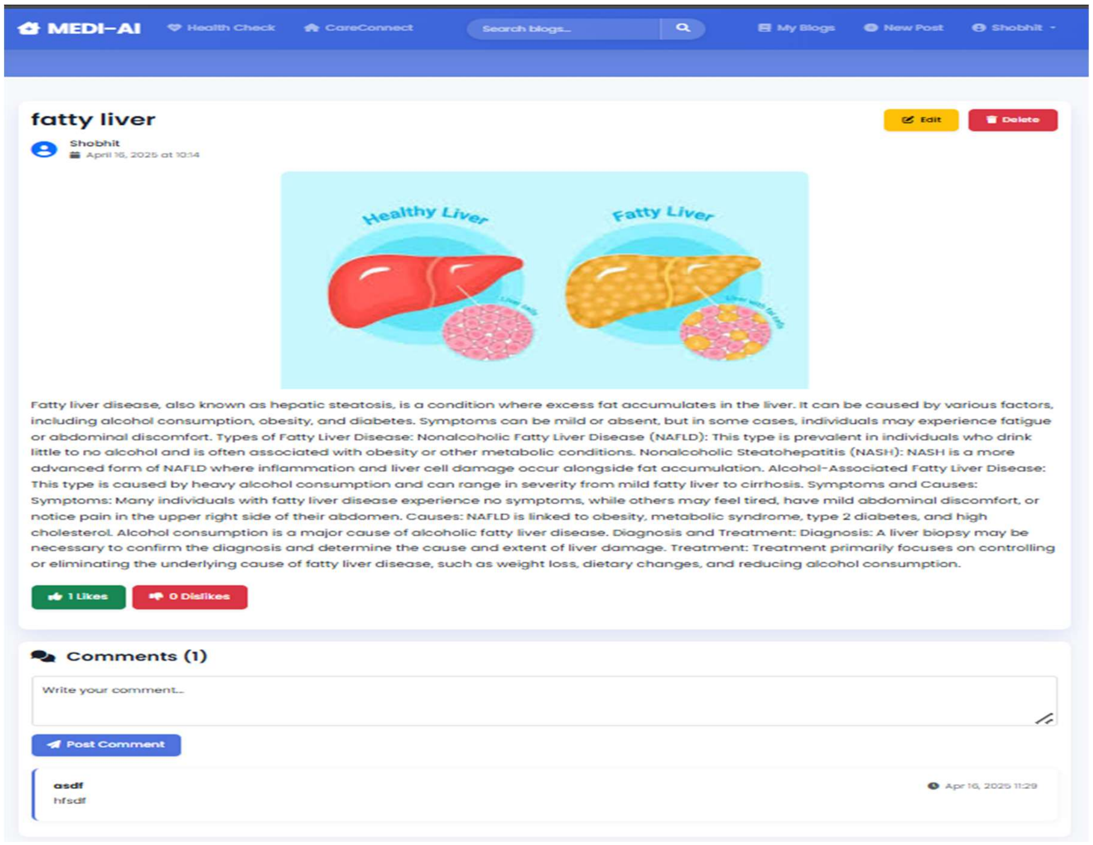


Figure 19- Vlog writing Page



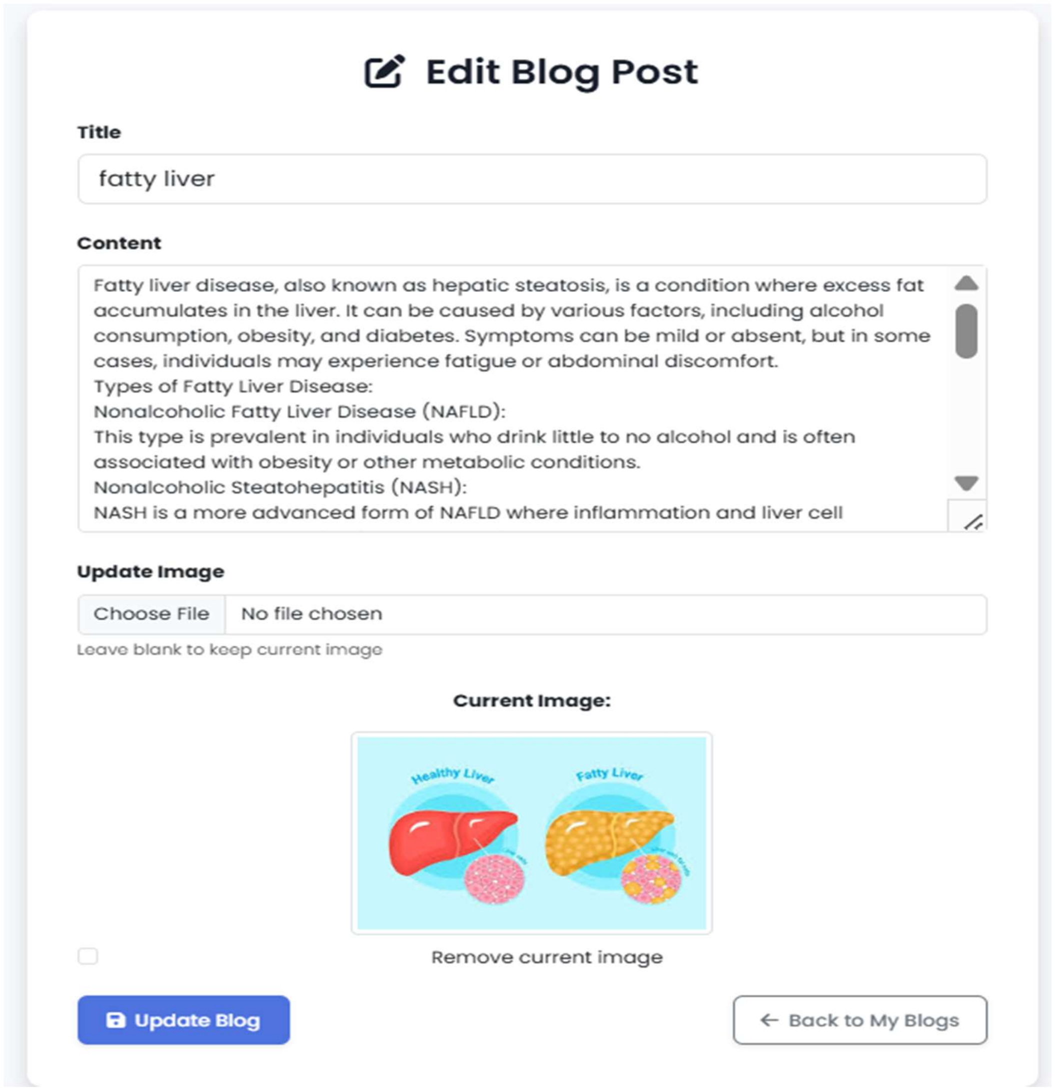
fatty liver

Shobhit
April 16, 2025 at 10:14

Fatty liver disease, also known as hepatic steatosis, is a condition where excess fat accumulates in the liver. It can be caused by various factors, including alcohol consumption, obesity, and diabetes. Symptoms can be mild or absent, but in some cases, individuals may experience fatigue or abdominal discomfort. Types of Fatty Liver Disease: Nonalcoholic Fatty Liver Disease (NAFLD): This type is prevalent in individuals who drink little to no alcohol and is often associated with obesity or other metabolic conditions. Nonalcoholic Steatohepatitis (NASH): NASH is a more advanced form of NAFLD where inflammation and liver cell damage occur alongside fat accumulation. Alcohol-Associated Fatty Liver Disease: This type is caused by heavy alcohol consumption and can range in severity from mild fatty liver to cirrhosis. Symptoms and Causes: Symptoms: Many individuals with fatty liver disease experience no symptoms, while others may feel tired, have mild abdominal discomfort, or notice pain in the upper right side of their abdomen. Causes: NAFLD is linked to obesity, metabolic syndrome, type 2 diabetes, and high cholesterol. Alcohol consumption is a major cause of alcoholic fatty liver disease. Diagnosis and Treatment: Diagnosis: A liver biopsy may be necessary to confirm the diagnosis and determine the cause and extent of liver damage. Treatment: Treatment primarily focuses on controlling or eliminating the underlying cause of fatty liver disease, such as weight loss, dietary changes, and reducing alcohol consumption.

asdf
hfsdf
Apr 16, 2025 11:29

Figure 20 – Full Vlog Page


Edit Blog Post

Title

fatty liver

Content

Fatty liver disease, also known as hepatic steatosis, is a condition where excess fat accumulates in the liver. It can be caused by various factors, including alcohol consumption, obesity, and diabetes. Symptoms can be mild or absent, but in some cases, individuals may experience fatigue or abdominal discomfort.
Types of Fatty Liver Disease:
Nonalcoholic Fatty Liver Disease (NAFLD):
This type is prevalent in individuals who drink little to no alcohol and is often associated with obesity or other metabolic conditions.
Nonalcoholic Steatohepatitis (NASH):
NASH is a more advanced form of NAFLD where inflammation and liver cell

Update Image

Choose File No file chosen

Leave blank to keep current image

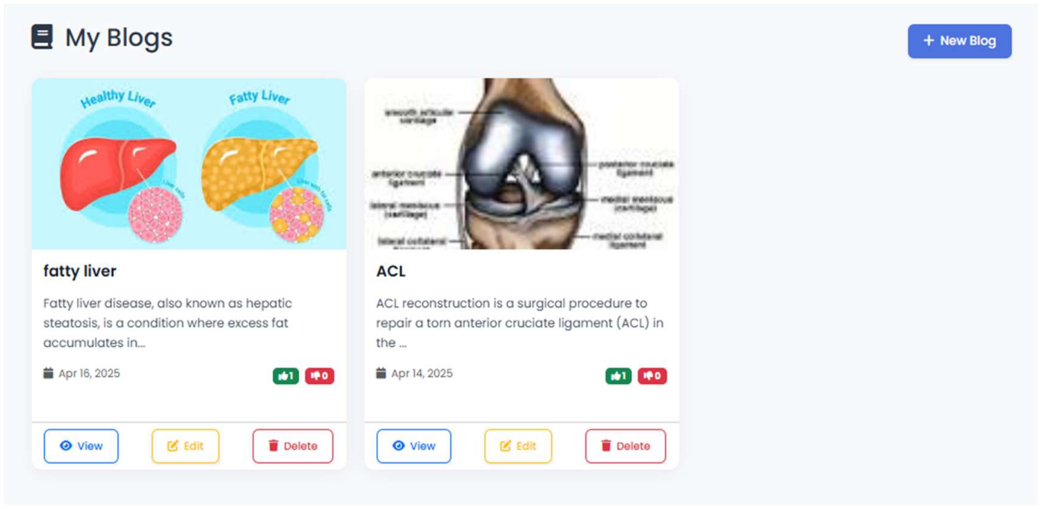
Current Image:

Remove current image

Update Blog

← Back to My Blogs

Figure 21 – Vlog Editing Page


My Blogs

+ New Blog

fatty liver

Fatty liver disease, also known as hepatic steatosis, is a condition where excess fat accumulates in...

Apr 16, 2025 1 0

View Edit Delete

ACL

ACL reconstruction is a surgical procedure to repair a torn anterior cruciate ligament (ACL) in the ...

Apr 14, 2025 1 0

View Edit Delete

Figure 22 – Account Based All Article

4. Testing & Analysis

Testing and analysis are fundamental components in the software development lifecycle, ensuring that each module performs as expected and the system as a whole is reliable, efficient, and user-friendly. The **MEDI-AI** platform was subjected to multiple rounds of testing to validate its core functionalities, which include AI-based disease prediction, user interaction, blog management, and security.

The testing strategy was divided into various categories: **unit testing**, **functional testing**, **integration testing**, **machine learning model evaluation**, **UI responsiveness testing**, and **bug tracking with iterative improvements**.

4.1 Unit Testing

Unit testing was used to verify the individual components of the system. These tests focused on backend functions such as:

- **User registration and login logic**
- **Form validation** (checking missing or incorrect fields)
- **Password encryption and session creation**
- **Database operations (add/update/delete records)**
- **Image upload handling and file format verification**
- **Symptom vector generation from user input**
- **Machine learning model inference**

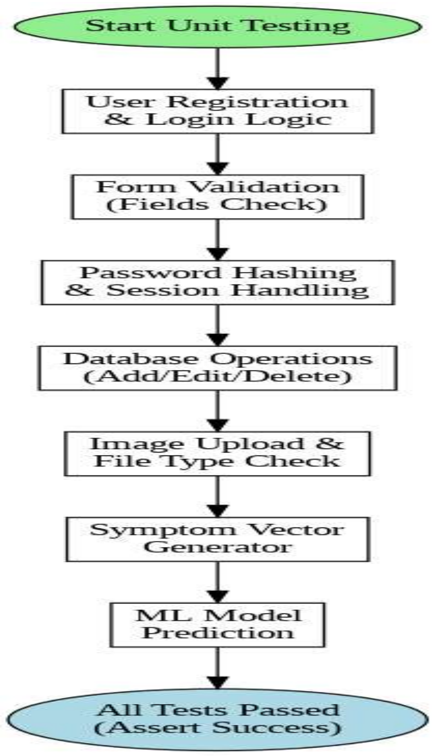

Start Unit Testing
User Registration & Login Logic
Form Validation (Fields Check)
Password Hashing & Session Handling
Database Operations (Add/Edit/Delete)
Image Upload & File Type Check
Symptom Vector Generator
ML Model Prediction
All Tests Passed (Assert Success)

Figure 23- Unit Testing

Python [2]'s built-in assertion testing and pytest were used to confirm expected outcomes from utility functions. For example, tests were conducted to ensure that hashed passwords were not stored in plaintext, and invalid login attempts were correctly rejected.

4.2 Functional Testing

Manual functional testing was performed on all key modules to ensure they behaved as intended from the end user's perspective. Key areas included:

1. **User Module**
    - New users were able to register and login with valid credentials.
    - Sessions persisted securely across pages and expired on logout.
    - Incorrect login attempts showed proper error messages.
2. **Symptom Checker & Prediction**
    - Symptom checkboxes could be selected or deselected individually or in combination.
    - On form submission, the system returned predictions with negligible delay.
    - Disease prediction was consistent and correct for various symptom combinations.
3. **Recommendations**

For each disease predicted by the SVM model, the system successfully retrieved:

- **Precautions**
- **Medications**
- **Diet plans**
- **Workouts**

All data was extracted from preloaded datasets and displayed in a user- friendly format.

4. **Blog Module**
    - Users could create blogs with or without images.
    - Edit and delete functionality worked with ownership checks (only the blog's author could modify it).
    - Title, content, and image fields were validated for correctness.
    - Old images were properly deleted when updated with new ones.
5. **Comments and Reactions**
    - Logged-in users could comment on any blog.
    - Comments appeared instantly and were stored in the database with timestamps.
    - Likes and dislikes were handled intelligently — a user could only perform one action per blog, and switching was allowed (e.g., like to dislike).
    - Attempting to react again showed no duplicate entry.

6. **Search**

- Keywords entered in the search bar correctly filtered blog titles or content.
- Empty results prompted user-friendly messages

4.3 Model Testing

The Support Vector Machine (SVM) model used for disease prediction was trained on a labeled dataset and tested with both synthetic and real-world data samples
Model Evaluation Metrics:

- **Accuracy**: Achieved over 95% accuracy on validation sets.
- **Speed**: Delivered predictions in <1 second due to pre-loaded models.
- **Precision & Recall**: High values for diseases with distinct symptom sets.

The model was also stress-tested with ambiguous symptoms to test robustness. In such cases, the model selected the most probable disease while still retrieving a relevant recommendation set.

4.4 UI Responsiveness & Cross-Platform Testing

The frontend, built with Bootstrap [8] 5 and Jinja2 templates, was tested across:

- **Browsers**: Chrome, Firefox, Edge
- **Devices**: Desktop, tablets, Android phones

Results showed consistent UI rendering, mobile responsiveness, and working interactivity (dropdowns, toggles, buttons). The layout scaled correctly on screens as small as 320px width.

4.5 Bug Tracking and Iterative Improvement

Throughout development, issues were tracked and addressed in real time. Common bugs included:

- Image upload failure for unsupported formats
- Unresponsive form buttons on mobile
- Blog deletion errors due to incorrect session handling
- Symptoms not being deselected correctly on form reset

These bugs were systematically debugged, and improvements were made by adding file type restrictions, enhancing user prompts, and refining JavaScript behaviour.

4.6 User Experience Evaluation

The **user experience (UX)** was a critical focus during the testing phase of the MEDI-AI project. Beyond ensuring the technical correctness of features, I aimed to validate how real users would perceive, interact with, and benefit from the platform. Usability, responsiveness, intuitiveness, and accessibility were prioritized to ensure that the platform meets the expectations of both technical and non-technical users

5. Deployment & Maintenance

The deployment and maintenance phase of the **MEDI-AI** project marks the transition from development to real-world use. It ensures the system is accessible, operational, and scalable for end users. The project was developed using Python [2]'s **Flask [1] framework**, integrated with **SQLite [6]** for the database, and was designed to run both in a local environment and be scalable to cloud-based hosting platforms.

5.1 Deployment Strategy

The system was first deployed and tested locally using Flask [1]'s built-in development server. This allowed real-time testing of all routes, database operations, and template rendering. The directory structure was organized with clear separation of static files, templates, models, and routes to simplify migration to production.
For production-level deployment, the following options were considered:

- **Heroku** (PaaS): Ideal for lightweight Flask [1] apps with SQLite [6] or PostgreSQL support.
- **Render / Railway**: Easy deployment of full-stack apps with automatic CI/CD.
- **VPS (e.g., DigitalOcean)**: For more advanced users needing root access and scalability.

Basic deployment involved:

- Hosting the Flask [1] app using a production-grade WSGI server like **Gunicorn**
- Using **Nginx** as a reverse proxy
- Environment variable setup for secret keys and configurations
- Static file handling and media path mapping

5.2 Maintenance Strategy

Ongoing maintenance is critical to ensure system reliability, user satisfaction, and security. The following practices were implemented and proposed:

- **Database Backups**: Regular backups of the SQLite [6] database to avoid data loss and enable recovery in case of failure.
- **Model Updates**: The SVM machine learning model is stored using pickle. Periodic retraining with updated symptom-disease mappings will improve prediction accuracy.
- **Log Monitoring**: Flask [1]'s logging mechanism captures runtime errors, which aids in debugging and stability analysis.
- **User Feedback Handling**: A feedback channel is recommended to collect suggestions or bug reports from users to guide future improvements.

5.3 Future Enhancements

To support long-term maintenance and scalability, future versions of MEDI-AI may include:

- Migrating from SQLite [6] to **PostgreSQL** or **MySQL**
- Adding an **admin panel** for content moderation and user management
- Deploying via **Docker** containers for easier replication and versioning
- Implementing **cron jobs** for regular health checks and cleanup tasks

Testing feedback and error logs are regularly reviewed during local usage to enhance functionality. Source code is managed through Git, enabling version control and rollback if necessary. Future maintenance will include updating packages, refactoring redundant code, optimizing database queries, and ensuring third-party libraries remain up-to-date.

6. Conclusion

The **MEDI-AI** project successfully integrates artificial intelligence with digital healthcare to offer a reliable, user-friendly platform for early disease prediction and personalized medical guidance. By leveraging machine learning, Flask [1] web development, and structured data processing, the system provides intelligent predictions based on user-input symptoms and offers recommendations for medications, diet, workout, and precautions. Additionally, the inclusion of a community-driven blog system allows users to share their health experiences and engage with others, promoting both awareness and empathy.

The core strength of MEDI-AI lies in its modular architecture, combining a trained SVM model with a responsive web interface and a secure, scalable backend. The system's use of real medical datasets for recommendations and its intuitive design ensures accessibility even for non-technical users. Throughout development, focus was placed on performance, responsiveness, and simplicity — ensuring that users receive fast, accurate, and actionable insights from their symptom inputs.

Extensive testing was conducted at both unit and system levels to ensure functional accuracy, data integrity, and seamless user experience. The blog interaction module was also optimized for security and user engagement, including like/dislike restrictions, comment threading, and content moderation capabilities.

Looking forward, MEDI-AI holds immense potential to evolve into a more advanced diagnostic assistant by incorporating real-time doctor feedback, multilingual support, and integration with external health APIs. The platform's modularity ensures that future features like electronic health records (EHR), chatbot integration, or mobile app deployment can be smoothly added.

In conclusion, MEDI-AI stands as a valuable digital health companion that combines AI technology with community engagement to empower users in making informed health decisions. It addresses the gap between self-awareness and medical consultation and opens doors for a smarter, more connected healthcare experience.

7. Applications and Advantages

7.1 Applications

The **MEDI-AI** platform is designed to be a multipurpose tool that can be applied across various domains of digital healthcare and community engagement. Its applications span across personal, clinical, educational, and technological areas:

- **Self-Diagnosis and Early Awareness**
  MEDI-AI empowers users to self-assess their symptoms and receive AI-generated predictions. This can help individuals make informed decisions about their health before visiting a doctor, potentially catching diseases at an early stage.
- **Medical Guidance in Rural and Remote Areas**
  In regions with limited access to healthcare professionals, MEDI-AI acts as a virtual assistant by providing disease predictions, precautions, and health advice. It bridges the gap between need and accessibility.
- **Health Education and Blogging**
  The integrated blog system serves as a health awareness platform where users can share their experiences, read about different health conditions, and learn from real-life stories. This is useful in community health campaigns and awareness programs.
- **Preliminary Tool for Clinics and Health Startups**
  Clinics can use the symptom checker and prediction module as a first step before patient consultation, improving time efficiency. Health-focused startups can integrate this logic into their systems as a base product.
- **Academic and Research Use**
  The architecture, dataset usage, and machine learning integration make MEDI-AI a useful tool for educational institutions, students, and researchers exploring the intersection of AI and healthcare.

7.2 Advantages

- **User-Friendly Interface**
  The platform's design is intuitive, mobile-responsive, and accessible to all age groups, ensuring wide usability.
- **AI-Based Intelligence**
  It uses a trained machine learning model (SVM) for real-time, accurate disease prediction, making it smarter than rule-based systems.
- **Fast and Cost-Effective**
  MEDI-AI delivers instant suggestions at no cost to users, making it an ideal

alternative for those who cannot access immediate medical care.

- **Secure and Scalable**
  The use of secure password storage, session management, and data validation ensures user safety. The modular architecture supports future scaling and feature upgrades.
- **Community Engagement**
  The blog module encourages users to share, learn, and interact, promoting mental well-being and peer learning in the health space.

References

[1] M. Grinberg, Flask Web Development: Developing Web Applications with python, 2nd ed., Sebastopol, CA: O'Reilly Media, 2018.

[2] G. Van Rossum and F. L. Drake, The Python Language Reference Manual, Python Software Foundation, 2023. [Online]. Available: https://docs.Python .org/3/

[3] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.

[4] Pandas, "Pandas Documentation: Python Data Analysis Library," 2023. [Online]. Available: https://pandas.pydata.org/docs/

[5] NumPy, "NumPy Documentation: Fundamental Package for Scientific Computing," 2023. [Online]. Available: https://NumPy.org/doc/

[6] SQLite, "SQLite Documentation: Lightweight Relational Database Engine," 2023. [Online]. Available: https://www.SQLite.org/docs.html

[7] SQLAlchemy, "The Database Toolkit for Python," 2023. [Online]. Available: https://docs.SQLAlchemy.org/

[8] Bootstrap, "Bootstrap v5: Front-End Framework for Responsive Web Design," 2023. [Online]. Available: https://getBootstrap .com/docs/5.3/

[9] Jinja, "Jinja2 Templating Engine for Python," 2023. [Online]. Available: https://jinja.palletsprojects.com/en/3.1.x/

[10] Pickle Module, "Python Object Serialization," Python Software Foundation, 2023. [Online]. Available: https://docs.Python.org/3/library/pickle.html

[11] K. Murphy, Machine Learning: A Probabilistic Perspective, Cambridge, MA: MIT Press, 2012.

[12] P. Barry and P. Crowley, Modern Embedded Computing: Designing Connected, Pervasive, Media-Rich Systems, Morgan Kaufmann, 2012.

[13] Visual Studio Code, "Code Editing. Redefined, 2023. [Online]. Available: https://code.visualstudio.com/docs

[14] E. Marcotte, Responsive Web Design, 2nd ed. New York: A Book Apart, 2014.

[15] Draw.io, "Free Online Diagram Software for Making Flowcharts, Process Diagrams," 2023. [Online]. Available: https://www.drawio.com/doc/

[16] pytest, "Simple and Powerful Testing with Python [2]," 2023. [Online]. Available: https://docs.pytest.org/

Chhatrapati Shahu Ji Maharaj (CSJM) University

School of Engineering & Technology (UIET)
Department of Computer Application

Interaction with Project Mentor

Group No.: ……… **Project Mentor Name:** ……….…………………………….

S. No.	*Date of Interaction*	*Time*	*Discussion Points*	*Signature (Project Mentor)*