Heavens' Light is Our Guide

Rajshahi University of Engineering & Technology

Department of Computer Science & Engineering

CSE 3100
Web-Based Application Project

HCV-Ai: Non-invasive Hepatitis C Virus Detection System

Submitted by

Kaif Ahmed Khan

Roll: 2103163

Submitted to

Md. Farhan Shakib
Lecturer

August 27, 2025

Contents

List of Codes

List of Figures

1 Introduction

The web application brings to life a machine learning framework developed for the non-invasive detection of Hepatitis C Virus (HCV), as outlined in our recent research. Designed with accessibility and accuracy in mind, the platform allows users—particularly patients and healthcare providers—to upload standard laboratory data and receive predictions about the possible stage of HCV infection: healthy, hepatitis, fibrosis, or cirrhosis.

The primary objective of the application is to make early and accurate HCV detection more accessible, especially in low-resource settings where traditional diagnostic methods are often invasive, costly, and difficult to access. By leveraging routine blood test results and state-of-the-art machine learning techniques—including synthetic data generation, feature selection, and explainable AI—this application aims to provide reliable, interpretable predictions that can assist in timely clinical decision-making.

This system addresses a critical gap in healthcare: the lack of affordable and scalable tools for multiclass liver disease classification. Whether you're a clinician, researcher, or concerned individual, this platform empowers you with actionable insights using only non-invasive lab data.

2 System Components

2.1 Frontend Components

2.2 Backend Components

2.3 Dataflow Diagram

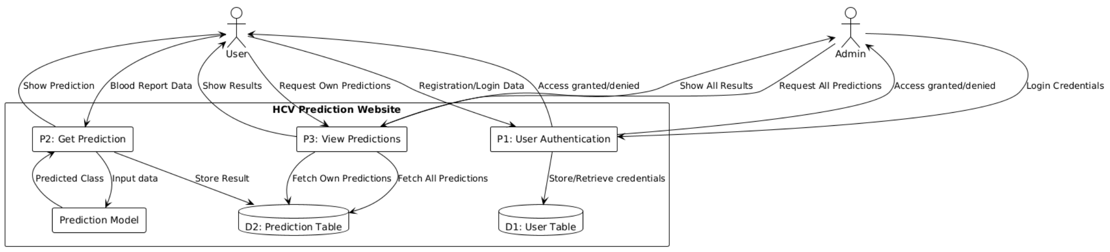

Figure 1: *Dataflow diagram of the website*

2.4 User Interaction Diagram

2.5 Key Features

User Management

- User registration and login system with PostgreSQL database authentication.
- Role-based access control: Patients (users) and Admins.
- Secure storage of user information.

Prediction System

- Input form for blood report data (patient submits laboratory values).
- Pretrained machine learning model processes the input.
- Prediction of HCV stage (4 classes: 0–3).
- Prediction result is displayed instantly to the user.

Prediction History

- Patients can view their own previous predictions.
- Admins can view all users' predictions for monitoring and analysis.

Database Integration

- PostgreSQL database with two main tables:
  - **User Table**: Stores user credentials and profile information.
  - **Prediction Table**: Stores patient input, prediction results, and timestamps.

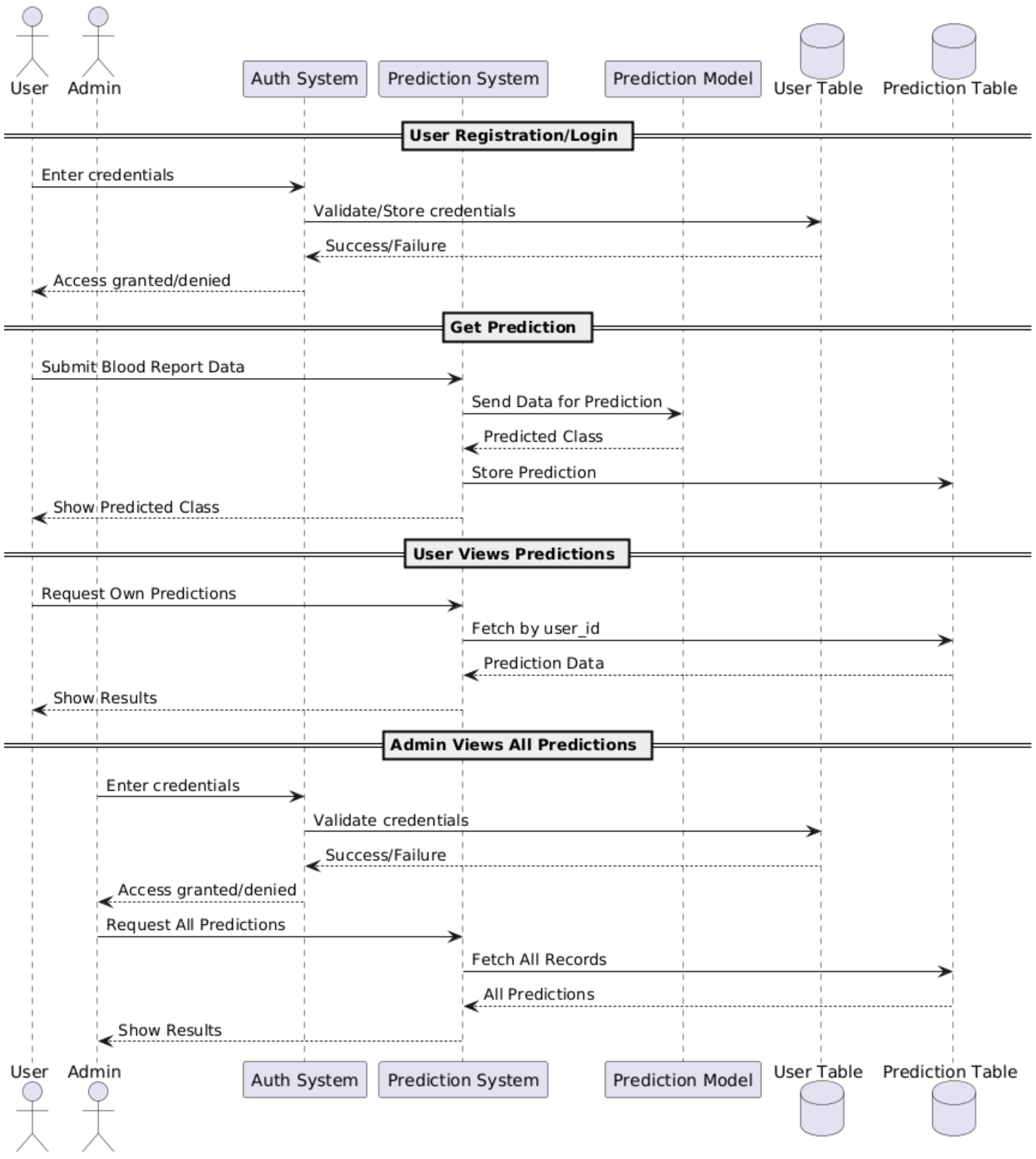

Figure 2: *Sequence diagram for showing how the user interacts with the website*

Security and Roles

- Authentication system for both patients and admins.
- Role-specific permissions (patients see only their own data, admins see all data).

3 Tools & Technologies

3.1 Languages

- HTML
- CSS
- JavaScript
- Python
- PostgresSQL

3.2 Next.js

3.3 FastAPI

3.4 Postgres(Supabase)

3.5 Scikit-Learn

Justification for Choice of Technology Stack

The technology stack for the medical prediction website was carefully chosen to balance performance, scalability, and ease of development. Next.js was selected for the frontend because it provides a modern React framework with support for server-side rendering and static site generation, ensuring fast page loads and a responsive user interface. FastAPI was chosen for the backend due to its lightweight design, asynchronous capabilities, and automatic API documentation, which allows for efficient handling of requests and easy integration with machine learning models. PostgreSQL, accessed via Supabase, serves as the database system because it is reliable, scalable, and supports complex queries while Supabase provides an easy-to-use backend-as-a-service interface that simplifies authentication and data management. Finally, Scikit-Learn was used for the machine learning component due to its simplicity, extensive functionality for classification tasks, and seamless integration with Python, enabling quick development and deployment of the pre-trained HCV prediction model. This combination of technologies ensures that the system is both performant and maintainable while allowing for future scalability and enhancements.

4 Sustainability and Environmental Considerations

In developing the medical prediction website, sustainability and environmental impact have been considered in both design and deployment. The following strategies are applied:

Performance Optimization

The backend, implemented using FastAPI, is lightweight and asynchronous, which reduces server load and ensures efficient handling of requests. The frontend, built with Next.js, supports static site generation and server-side rendering, which decreases repeated computation and improves response times. These optimizations reduce unnecessary resource consumption and improve the overall energy efficiency of the system.

Energy Efficiency in Coding and Infrastructure

The codebase follows modular and efficient practices to minimize redundant operations. Database queries are optimized to avoid excessive computation in Supabase. Containerization with Docker allows for consistent and isolated environments, reducing wasted resources across deployments. Additionally, horizontal scaling strategies are preferred over overprovisioning, ensuring resources are used only when required.

Long-Term Impact

By deploying the application on modern cloud platforms such as Render or Railway, the system benefits from shared, energy-efficient infrastructure. This reduces the need for dedicated high-power servers. The chosen stack supports scalability, which means the application can grow without significant increases in energy use per user. In the long term, the platform aims to provide medical predictions with minimal environmental footprint while supporting sustainable digital health practices.

5 Project Management and Cost Estimation

Project Timeline

The development phase is designed to be lightweight and completed in approximately 6 weeks:

- Requirement Analysis and Design: 0.5 week
- Backend Development (FastAPI): 1 week
- Frontend Development (Next.js): 1 week
- Database Setup (Supabase): 0.5 week
- Integration and Dockerization: 1 week
- Testing and Debugging: 1 week
- Deployment and Documentation: 1 week

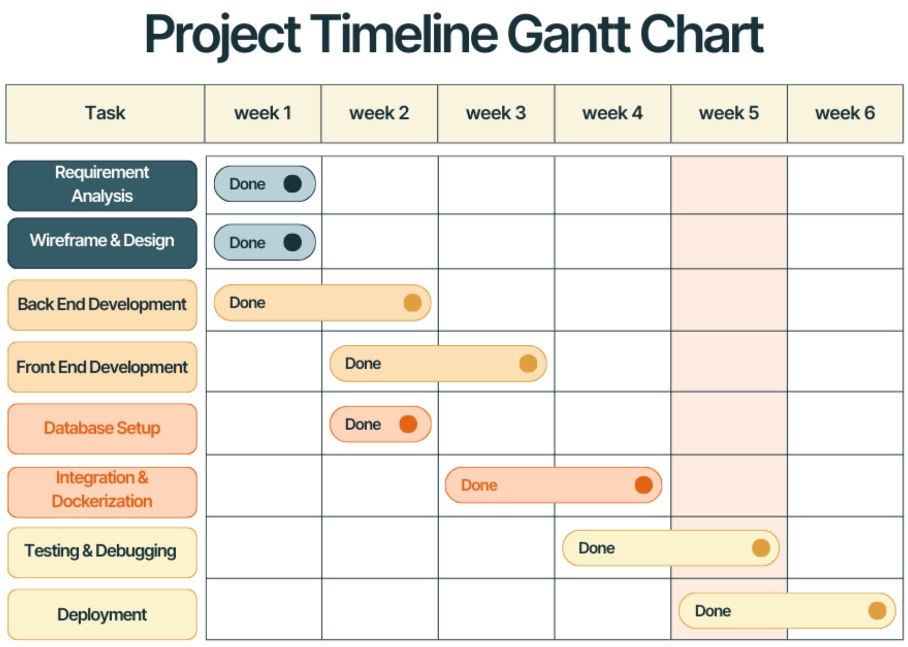

Figure 3: *Gantt chart of project timeline.*

Cost Estimation

The project is planned with minimal cost by leveraging free and open-source technologies:

- **Development Effort:** The project is developed by a single developer as part of an academic exercise, so direct labor cost is negligible. If monetized, the equivalent effort can be valued at approximately \$500–\$800.

- **Operational Costs:** Free tiers of Supabase (database) and Render/Railway (hosting) are sufficient for deployment. Docker is open-source and free. A custom domain is optional, costing about $10 per year.

Overall Estimate: The project can be developed with near-zero cost using free tiers, with only optional expenses for a domain name.

6 Conclusion

The medical prediction website successfully delivers a platform where users can register, log in, submit their blood report data, and receive predictions for HCV stages using a pretrained machine learning model. Users can view their prediction history, while administrators have access to all prediction records for monitoring purposes. The integration of FastAPI, Next.js, Supabase, and Docker has resulted in a lightweight, efficient, and scalable system.

During development, several lessons were learned and challenges encountered. Implementing the backend API with FastAPI highlighted the importance of asynchronous programming to efficiently handle multiple requests. Integrating the machine learning model required careful preprocessing and validation to ensure accurate predictions. Designing the database schema and optimizing queries in Supabase was crucial for reliable storage and retrieval of prediction data. Deploying the application with Docker and on cloud platforms presented challenges related to environment configuration and dependency management. Additionally, managing overlapping frontend and backend development tasks emphasized the need for careful project scheduling.

For future improvements, the system could incorporate enhanced user authentication mechanisms such as OAuth or multi-factor authentication to strengthen security. Visualization features could be added to display prediction trends over time for each user, and the machine learning model could be continuously retrained with anonymized new data to improve accuracy. Notifications or email summaries could be implemented to inform users of new predictions, and the platform could be expanded to support additional medical conditions or lab tests. Finally, the deployment process can be further improved with automated CI/CD pipelines and monitoring tools to ensure reliability and maintainability. This project lays a solid foundation for an efficient, scalable, and user-friendly medical prediction system while providing opportunities for future enhancement.