CHAPTER 1

INTRODUCTION

1.1 Overview

Liver cirrhosis is a chronic disease characterized by the replacement of healthy liver tissue with scar tissue, leading to progressive deterioration of liver function. It is often caused by factors such as chronic alcohol consumption, hepatitis infection, and fatty liver disease. Early detection and continuous monitoring are crucial for managing this condition and preventing severe complications.

The project focuses on the early detection and continuous monitoring of liver cirrhosis through advanced machine learning techniques. Leveraging the ILPD dataset from the UCI repository, an ensemble model comprising Random Forest, XGBoost, and Gradient Boosting is utilized to predict liver cirrhosis based on key liver function parameters. Users can input their data manually or upload liver function test reports, from which the necessary parameters are extracted using image processing techniques with Pytesseract. The system predicts the presence of liver cirrhosis using parameters such as age, gender, AG ratio, total bilirubin, direct bilirubin, alkaline phosphatase, SGPT, SGOT, total proteins, and albumin. The project uses Flask for web development and MySQL for database management.

The prediction results are displayed on a user-friendly webpage, providing an accessible interface for patients and healthcare providers. Additionally, the project features a comprehensive analysis tool that tracks and visualizes the progress of each patient's liver function parameters over time, aiding in personalized healthcare and improving patient outcomes. Overall, the project aims to enhance early diagnosis and ongoing monitoring of liver cirrhosis, offering a valuable tool for both patients and healthcare professionals to manage liver health effectively.

1.2 Objectives

- Predictive Model: Using Ensemble Technique to predict liver cirrhosis.

- User Interface: Develop a user-friendly platform for patients to upload medical reports or upload data manually.
- Early detection: Improve early detection methods for liver cirrhosis.
- Visualization: for gaining insights and timely management of liver disease progression.

1.3 Purpose, Scope and Applicability

1.3.1 Purpose

The project aims to address the critical need for early detection and continuous monitoring of liver cirrhosis, a debilitating condition that can lead to severe health complications if left unmanaged. By leveraging machine learning techniques, the project seeks to provide an accurate, efficient, and accessible method for predicting liver cirrhosis based on key liver function parameters. This early detection capability can significantly improve patient outcomes by enabling timely medical intervention and personalized treatment plans.

The project enhances the existing healthcare system by integrating data extraction from liver function test reports and providing a comprehensive analysis tool that tracks the progress of liver health parameters over time. This approach not only supports better clinical decision-making but also empowers patients with valuable insights into their liver health, fostering a proactive approach to disease management. The theoretical framework combines machine learning, web development, and database management to create a cohesive and impactful solution for liver cirrhosis detection and monitoring.

1.3.2 Scope

The methodology involves leveraging an ensemble of machine learning models, Random Forest, XGBoost, and Gradient Boosting, trained on the ILPD dataset to predict liver cirrhosis based on key liver function parameters. User data is collected either through manual entry or by uploading liver function test reports, with the necessary parameters extracted using Pytesseract. The system, developed using Flask for the web interface and MySQL for database management, displays prediction results and tracks the progress of liver function parameters over time.

Assumptions

- The ILPD dataset is representative of the broader population affected by liver cirrhosis.
- The parameters extracted from liver function test reports are accurate and sufficient for prediction.

Limitations

- The accuracy of predictions depends on the quality and completeness of the input data.
- The system's performance may vary across different demographic groups not well-represented in the training dataset.

1.3.3 Applicability

Direct Applications:

- **Clinical Decision Support**: The project provides healthcare professionals with a tool for early and accurate detection of liver cirrhosis, enhancing decision-making and enabling timely medical interventions.
- **Patient Monitoring**: It offers a platform for patients to monitor their liver health over time through accessible web-based visualization, supporting proactive management of their condition.
- **Integration with Health Records**: The system can be integrated with existing electronic health records (EHR) systems, improving the workflow for clinicians by providing additional insights into liver function.

Indirect Applications

- **Advancement of Machine Learning Techniques**: The project demonstrates the application of ensemble learning methods in medical prediction, contributing to the broader field of machine learning and its applications in healthcare.

- **Educational Tool**: It serves as a resource for educational purposes, providing students and researchers with a practical example of integrating machine learning, data extraction, and web development in a healthcare context.
- **Healthcare Data Analysis**: By analysing liver function data, the project aids in understanding trends and patterns in liver health, which can inform future research and public health strategies.

1.4 Organisation of report

In the chapters that follow, this project report provides a comprehensive and structured examination of the development and implementation of the Prediction of Liver Cirrhosis and Analysis System

Chapter 2: Literature Survey begins by introducing the landscape of liver cirrhosis detection using machine learning technologies. It offers a detailed summary of significant research papers and studies that have shaped the current understanding and advancements in this field. Each paper is reviewed to extract methodologies, findings, and their implications. Following this, the chapter identifies the drawbacks of existing systems, highlighting the areas where current approaches fall short. The problem statement is then defined. The chapter concludes with the proposed solution, outlining the innovative approach this project takes to overcome the identified challenges.

Chapter 3: Requirement Engineering outlines the necessary software and hardware tools used for the project. It delves into conceptual and analysis modelling, presenting sequence diagrams, data flow diagrams to illustrate the system's functionality and behaviour. The Software Requirements Specification (SRS) section details the functional and non-functional requirements, providing a clear framework of what the system is expected to achieve.

Chapter 4: System Design explores the architectural framework of the system, describing the overall system architecture and breaking down the components and modules. It details the design of each component, including module decomposition, and elaborates on the interface design and data structure design.

Chapter 5: Implementation presents the plan of implementation, discussing the standards used throughout the process. It covers the coding details and explains the strategies employed to ensure code efficiency and optimization.

Chapter 6: Testing outlines the approach taken for testing the system. It details unit testing, where individual modules are tested for functionality, and integrated testing, where all modules are brought together to ensure interoperability and identify any errors or bugs. This chapter ensures that the system meets the specified requirements and functions.

Chapter 7: Results Discussion and Performance Analysis provides a thorough examination of the test reports, discussing the outcomes and performance of the system. It includes user documentation to guide users in understanding and utilizing the system effectively.

Chapter 8: Conclusion, Applications, and Future Work wraps up the report by summarizing the findings and achievements of the project. It discusses the practical applications of the system, highlighting how it can benefit users and the healthcare industry. The chapter also addresses the limitations of the system and suggests areas for future research and development to enhance its functionality and impact.

CHAPTER 2

LITERATURE SURVEY

2.1 Introduction

The application of machine learning in healthcare has transformed the approach to disease prediction and diagnosis, particularly for conditions like liver cirrhosis. Machine learning (ML) algorithms can analyse complex datasets to identify patterns and make accurate predictions, facilitating early detection and improving patient outcomes. Several ML algorithms have been explored for liver cirrhosis detection, each with unique strengths and weaknesses.

Random Forest is an ensemble learning method that combines multiple decision trees to improve accuracy and robustness. It leverages the diversity of multiple trees to enhance model performance and reduce overfitting. However, it can be slow to train and interpret, particularly with large datasets.

Gradient Boosting builds models sequentially, each correcting the errors of its predecessors, which improves predictive performance. It is highly effective for complex problems but is prone to overfitting and can be computationally expensive, requiring careful tuning to avoid these issues.

Support Vector Machines are used to find the optimal hyperplane for separating classes in high-dimensional spaces, making them effective for complex classification tasks. Despite their power, SVMs can be computationally intensive and may not perform well with noisy or overlapping data.

Naive Bayes is a probabilistic classifier based on Bayes' theorem and the assumption of feature independence. It is simple to implement and performs well with large datasets, but its assumption of independence between features can limit accuracy when features are correlated.

Convolutional Neural Networks are deep learning models designed for analyzing visual and spatial data. They excel in image classification and segmentation tasks but require substantial computational resources and large amounts of labeled data to achieve high accuracy.

To complete the project, a combination of essential technologies is employed. **Ensemble techniques** such as Random Forest, Gradient Boosting, and Support Vector Machines (SVM) are utilized to enhance prediction accuracy and robustness through advanced model aggregation and sequential learning. **Data extraction** is facilitated by **Pytesseract**, an OCR tool that converts text from liver function test reports into a usable format, complemented by preprocessing techniques like feature selection and outlier removal to optimize model performance. The web application is developed using **Flask** for its lightweight and flexible framework, along with **HTML/CSS/Bootstrap** for creating a responsive and visually appealing interface. **MySQL** is used for managing and storing patient records and liver function data, ensuring efficient data retrieval and management.

2.2 Summary of Papers

Aviral Srivastava; V Vineeth Kumar; Mahesh T R; V. Vivek – [1] This study utilizes Logistic Regression, Naive Bayes, and K-Nearest Neighbours to predict liver diseases based on enzyme levels, with model accuracy affected by data quality.

A. Gulia et al. – [2] This research applies Bayesian Network, SVM, and Random Forest to the UCI repository dataset, identifying Random Forest as the best-performing model. Limitations include dataset size and feature count.

Y. Kumar et al. – [3] Enhances decision tree performance with 20 classification rules for liver disease prediction. Limitations include rule interpretability and scalability to large datasets.

Tamilarasi A; Chitra K; Swetha J; Nihila R – [4] Employs ML and Deep Learning algorithms to compare Random Forest and SVM on cirrhosis data. Limitations involve data quality, feature selection, and model overfitting.

M. Abdar et al – [5] Evaluates MLPNN, CART, CHAID, and See5(C5.0) on the ILPD dataset, with limitations related to dataset size and accuracy of -14%.

EI-Shafeiy et al. – [6] Applies Boosted C5.0, SVM, and Naive Bayes on a dataset of 7000 patients with 23 attributes. Limitations include computational complexity and model interpretability.

Vijayarani et al. – [7] Compares Naive Bayes and SVM for liver disease data, with SVM showing better performance. Limitations include a noisy dataset.

M. Minnoor and V. Baths – [8] Evaluates algorithms with Extra Trees classifier using 5 attributes, assessing metrics like F1 score, accuracy, and precision. Limitations involve dataset size and model robustness across populations.

Sunil Kumar and Pooja Rani – [9] Compares various ML techniques for liver disease detection, focusing on metrics like accuracy, precision, recall, F1-score, AUC, and specificity. Limitations include dataset representativeness and algorithm scalability.

A. Srivastava et al. – [10] Compares ML algorithms with Random Forest on the UCI liver disease dataset. Limitations include dataset noise and sensitivity to feature selection.

T. M. Ghazal et al. – [11] Develops a prediction model using Logistic Regression, Naive Bayes, and Gradient Boosting, aimed at early prediction using hospital records. Limitations include dataset size and model interpretability.

S. S. Rasheed and I. H. Glob – [12] Classifies patient diseases using KNN, SVM, and Neural Networks with multiple clinic records, focusing on accuracy, F1-score, and specificity. Limitations are dataset representativeness and model scalability.

C. A. Reddy et al. – [13] Conducts a comparative analysis using Decision Trees, SVM, and Ensemble Methods on a regional health database, measuring AUC and recall. Limitations include model generalizability to diverse populations.

L. D. Sawant et al. – [14] Predicts liver cirrhosis using Decision Trees and Random Forests, with Random Forests performing better. Limitations include dataset representativeness and model interpretability.

P. K. V. Priyadharshini et al. – [15] Utilizes CNNs with advanced segmentation techniques for liver disease prediction, emphasizing deep learning methods. Limitations include dataset size and computational resource requirements.

Sateesh Ambesange; Vijayalaxmi A; Rashmi Uppin et al. – [16] Predicts liver disease using Random Forest algorithms and preprocessing techniques such as feature analysis, outlier removal, and hyperparameter tuning. Limitations include dataset imbalance affecting performance.

Priyanka Verma; Ketki Deshmukh; Deepa Krishnan – [17] Evaluates Decision Trees, SVM, and Ensemble Methods, focusing on performance metrics and their applicability to liver disease prediction. Limitations include challenges with model scalability and generalization.

Vilaskumar Patil et al. – [18] Applies CNNs with segmentation techniques to the Hepatitis C dataset, leveraging MRMR for feature selection. Limitations include test accuracy issues due to dataset quality.

2.3 Drawbacks of Existing System

- **Delayed Detection:** Current techniques often detect cirrhosis at an advanced stage when significant liver damage has already occurred.
- **Invasive Procedures:** Some methods, like liver biopsies, are invasive and uncomfortable for patients.
- **Intermittent Monitoring:** Reliance on periodic tests and clinical visits leads to missed opportunities for early intervention and timely management.
- **Higher Healthcare Costs:** Advanced liver disease requires more intensive treatments, increasing costs. Early detection can reduce these costs by preventing disease progression.

2.4 Problem Statement

Implement AI-driven predictive model and patient monitoring system for early detection and management of liver cirrhosis.

Input: Medical Report or manual entry of required details.

Output: Predict if person has liver cirrhosis or not and visualize uploaded data.

2.5 Proposed Solution

The proposed system uses Ensemble Technique to predict liver cirrhosis and provides visualizations for liver health monitoring. Patients can upload their medical reports on an easy-to-use platform. Early detection of high-risk patients allows for timely interventions and tailored care plans. This ultimately reduces the healthcare costs by preventing disease progression and optimizing resource use.

CHAPTER 3

REQUIREMENT ENGINEERING

3.1 Software and Hardware Tools Used

Software tools

- **Frontend Development:** HTML, CSS: For building the user interface.
- **Backend Development:** Flask: A micro web framework for Python, used for web development.
- **Database:** MySQL: A relational database management system for data storage and mysqldb: A MySQL database connector for Python.
- **Data Processing and Analysis:** Python with libraries like Pandas, Scikit-learn and Pytesseract for optical character recognition (OCR) to extract data from uploaded liver function test reports.
- **VS Code:** An interactive development environment for writing and testing code.

Hardware Tools

- **Operating System:** Windows 10 or macOS 10.13 (or newer)
- **Processor:** Intel Core i5/i7 or AMD Ryzen 5/7
- **RAM:** Minimum 8GB
- **Storage:**512GB SSD

3.2 Conceptual/ Analysis Modeling

3.2.1 Sequence Diagram

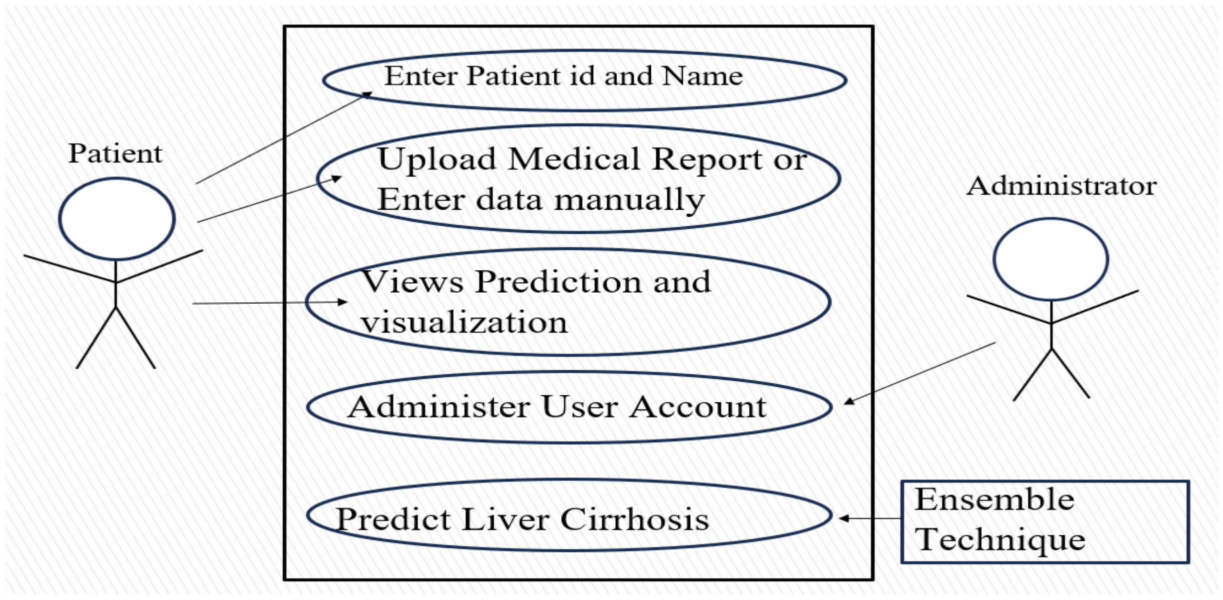

Patient
Enter Patient id and Name
Upload Medical Report or Enter data manually
Views Prediction and visualization
Administer User Account
Predict Liver Cirrhosis
Administrator
Ensemble Technique

Fig 3.1: Use Case Diagram

3.2.2 Data Flow Diagram

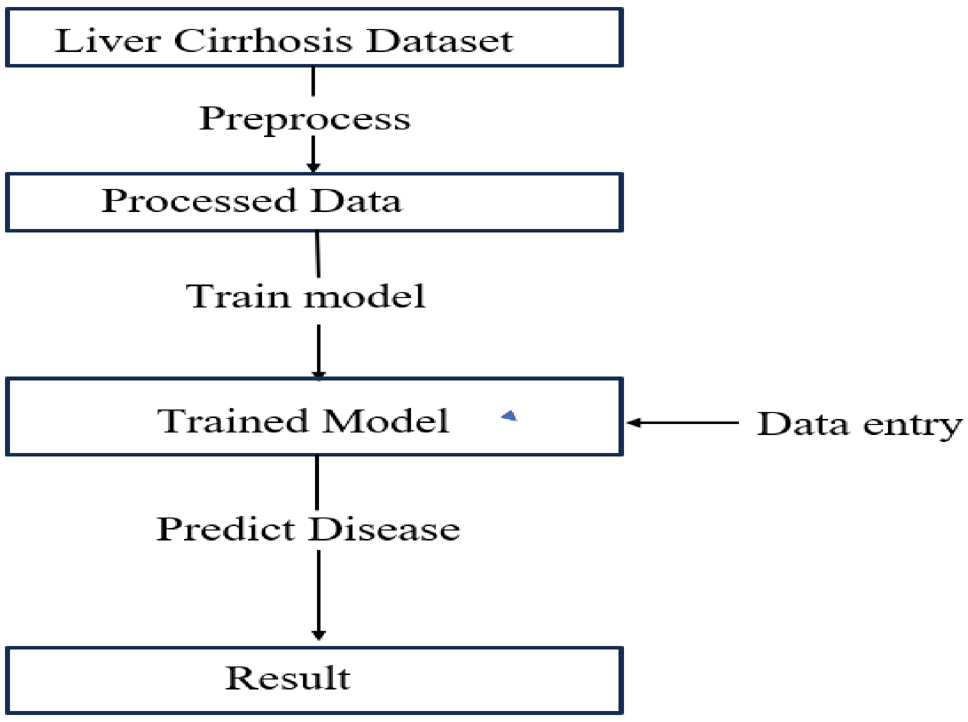


Fig 3.2: Data Flow Diagram

3.3 Software Requirements Specification

3.3.1 User Requirements

- **User Registration and Login**
  - Users should be able to register by providing personal details such as name,patient ID
  - Users should be able to log in using their patient ID and Name
- **Data Input**
  - Users should be able to input health-related data such as age, gender, AG ratio, total bilirubin, direct bilirubin, alkaline phosphatase, SGPT, SGOT, total proteins, and albumin.
  - Users can also upload their liver function test report from which the required parameters are extracted using Pytesseract.
- **Prediction Results**
  - Users should receive a prediction indicating the likelihood of liver cirrhosis based on the input data.
  - The results should be clearly displayed with visual aids (e.g., graphs, color-coded indicators) to help users understand their health status.
- **User Dashboard**
  - Users should have access to a dashboard where they can view their previous inputs and prediction results.

3.3.2 System Requirements

Functional Requirements

- **User Authentication**
  - The system must handle user registration, login, and logout functionalities securely.

- **Data Validation**
  - The system must validate the input data to ensure it is complete and in the correct format before processing.
- **Prediction Engine**
  - The system must use pre-trained machine learning models (Random Forest, XGBoost, Gradient Boosting) to process user input data and generate predictions.
  - The prediction engine should provide results within a reasonable timeframe (e.g., a few seconds).
- **Result Visualization**

- The system must present prediction results using graphs a to enhance user comprehension.

- **User Dashboard**
  - The system must store user data and prediction results in a secure database.

Non-Functional Requirements

- **Performance**
  - The system must provide fast response times for user interactions and should handle concurrent user sessions efficiently without performance degradation.
- **Security**
  - The system should implement measures to prevent unauthorized access and data breaches.
- **Usability**
  - The system must have an intuitive and user-friendly interface.
- **Scalability**

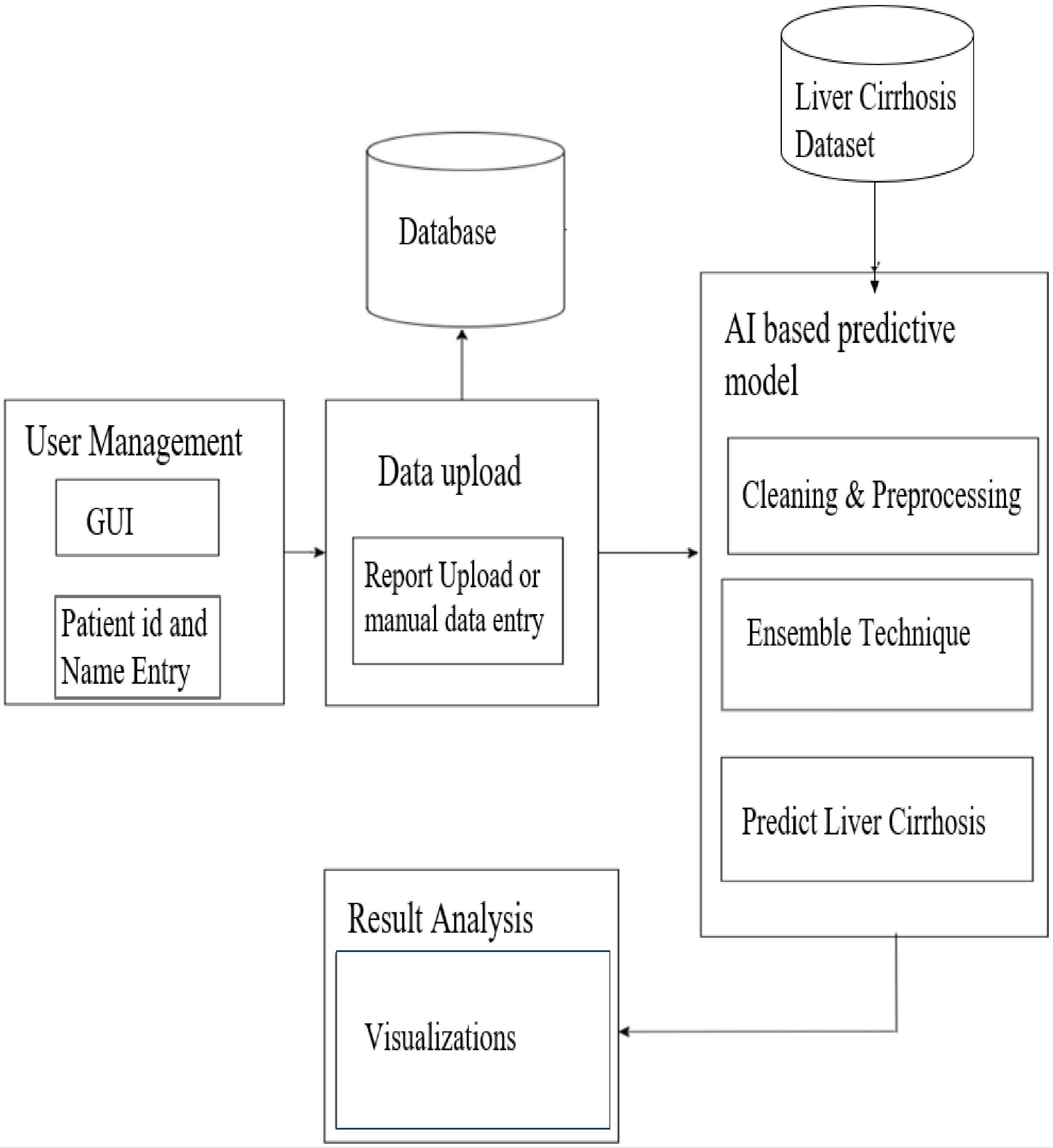
  - The system must be able to scale to accommodate increasing numbers of users and data volume.

- **Reliability**
  - The system must ensure high availability with minimal downtime.

CHAPTER 4

SYSTEM DESIGN

4.1 System Architecture



4.1: System Architecture

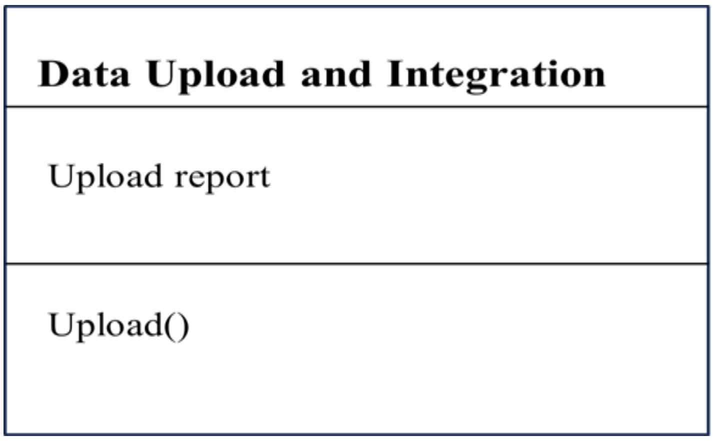
4.2 Component Design/ Module Decomposition

a) Front-end and Report Upload:

This module serves as the user interface and is responsible for interacting with the user.

- **User Interface (UI):** Allows users to interact with the system, providing a seamless experience to input data and receive results.

- **Upload Liver Test Data:** Allows users to upload their liver function test data to the system for further analysis.

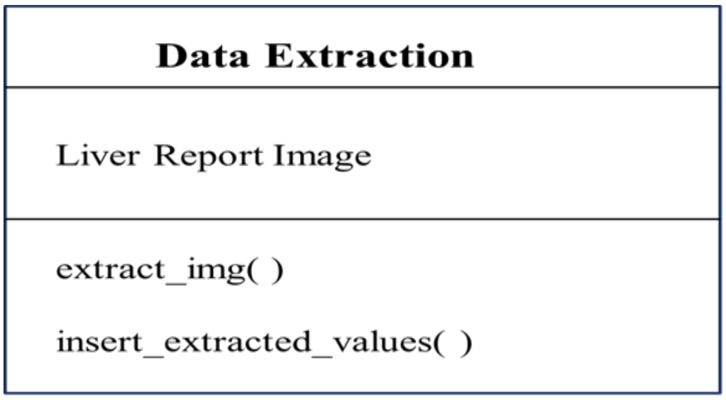


4.2: Component design of report upload

b) Data Extraction

This module is responsible for extracting the data from the patient reports.

- **Data Extraction:** Extracts the required details from the liver test report uploaded by the patient.

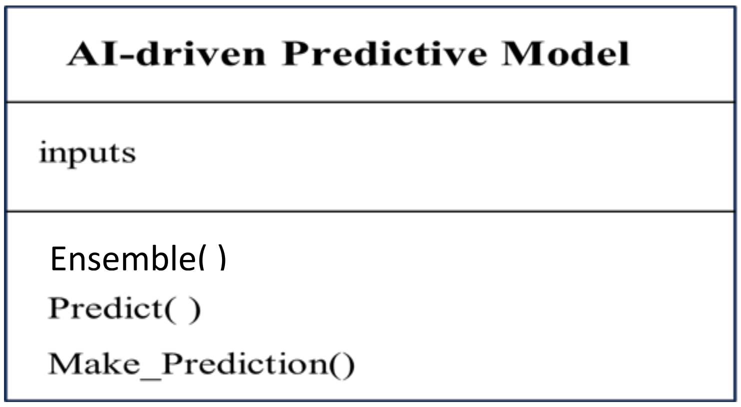


4.3: Component design of data extraction

c) Machine Learning Models

This module includes various machine learning algorithms used for predicting liver cirrhosis.

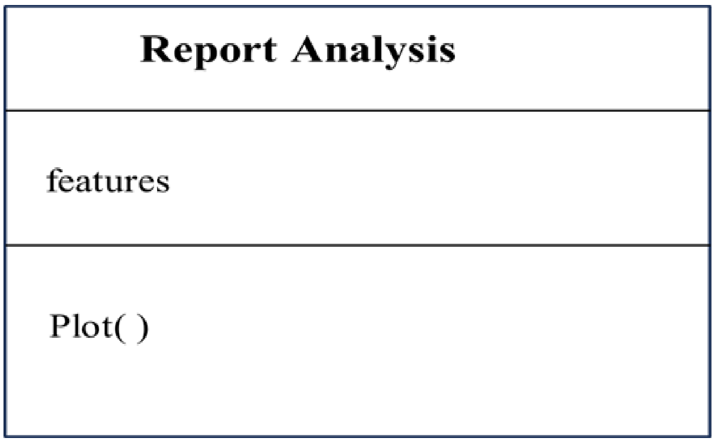
- **Random Forest:** An ensemble learning method using multiple decision trees for improved accuracy and robustness.
- **XGBoost:** An optimized gradient boosting library designed for speed and performance.
- **Gradient Boosting:** Another ensemble technique that builds models sequentially to correct errors made by prior models.



4.4: Component design of predictive model

d) Report Analysis

This module analyses the report obtained from the user and provides visualizations.



4.5: Component design of report analysis

4.3 Interface Design

4.3.1 User, Task, Environment Analysis

User Analysis:

- **Primary Users:**
  - **Healthcare Professionals:** Doctors, Nurses, and Medical Technicians: Use the system to input patient data and obtain liver cirrhosis prediction results.
  - **Patients:** Individuals: Use the system to input their own data and get a preliminary prediction.
  - **Researchers**: Medical Researchers and Data Scientists: Interested in analysing aggregated data and model performance.

Task Analysis:

- **Healthcare Professionals:**
  - **Input Patient Data:** Entering patient information such as age, gender, AG ratio, total bilirubin, direct bilirubin, alkaline phosphatase, SGPT, SGOT, total proteins, and albumin.
  - **View Predictions:** Accessing and interpreting the prediction results generated by the system.

  - **Manage Patient Records:** Storing and retrieving patient data securely.

- **Patients:**
  - **Input Personal Data:** Entering personal health information for preliminary analysis.
  - **View Results:** Accessing prediction results and recommendations.

Researchers:

  - **Analyse Data:** Accessing and analyzing aggregated patient data and model performance metrics.
  - **Adjust Model Parameters:** Experimenting with different model parameters

Environment Analysis:

- **Healthcare Environment:**
  - **Clinics and Hospitals:** The system will be used in a clinical setting with access to secure medical databases.
  - **Remote Access:** Healthcare professionals may need to access the system remotely via secure connections.
  - **Home Environment:** Patients may use personal computers or mobile devices to access the system.

Mapping Requirements to Develop a User Interface

- **User-Friendly Design**: The interface must be intuitive and easy to navigate, especially for users with limited technical skills.
- **Accessibility**: The interface must be accessible on various devices, including desktops, tablets, and smartphones.
- **Security:** User data must be handled securely, with encryption and secure login mechanisms.

- **Efficiency:** The interface must enable quick data entry and fast access to prediction results.

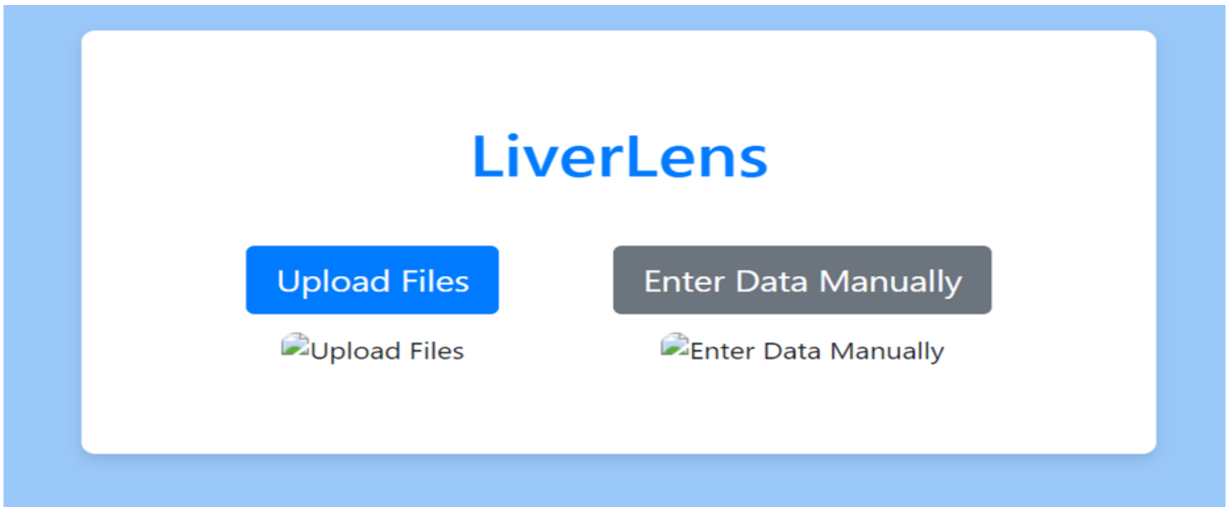
4.3.2 External and Internal Components of the User Interface

External Components:

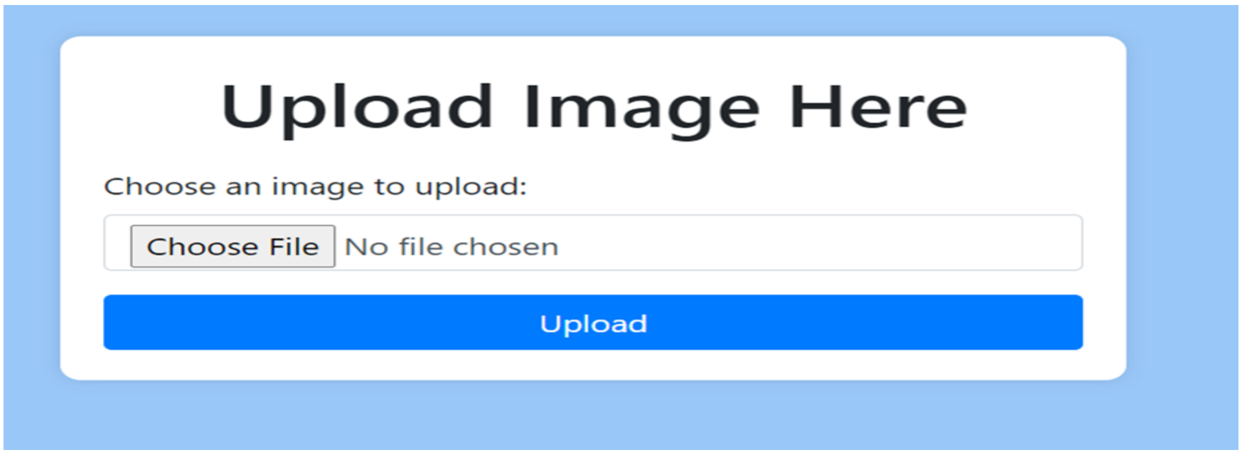
- **Web Browser:** Users interact with the system via a web browser, accessible from any internet-connected device.
- **Report Upload and Input Forms:** Space for uploading liver test report and Forms for entering patient data (e.g., age, gender, health parameters) using text fields, dropdowns, and checkboxes.
- **Result Display:** Sections for showing prediction results with charts, graphs, and text.

Internal Components:

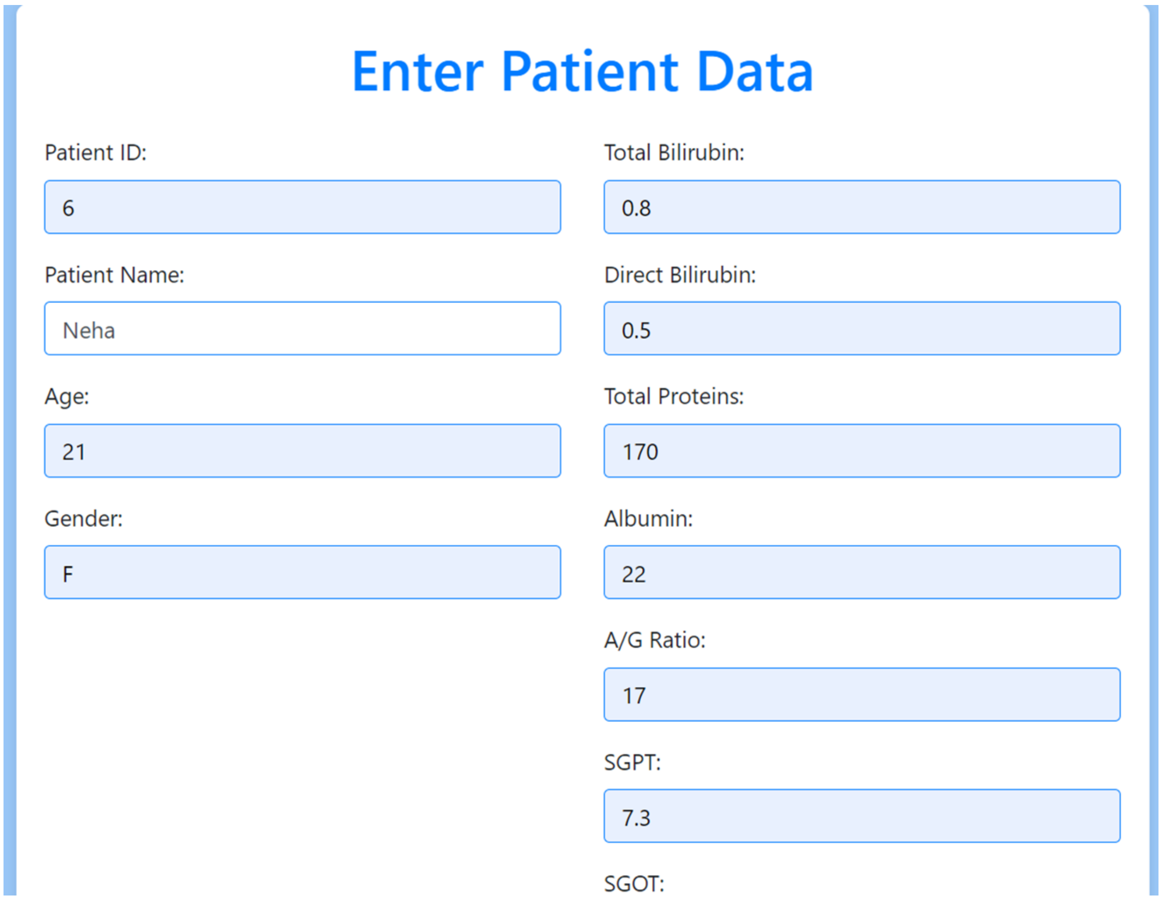
- **Frontend Framework:** HTML, CSS for building the interface.
- **Backend Framework:** Flask for server-side logic and MySQL for database management.
- **Data Extraction and Processing:** Python Libraries (Pandas, NumPy) for data handling, Pytesseract for extracting data and Machine Learning Models (Scikit-learn, XGBoost) for predictions.



4.6: LiverLens website Home page



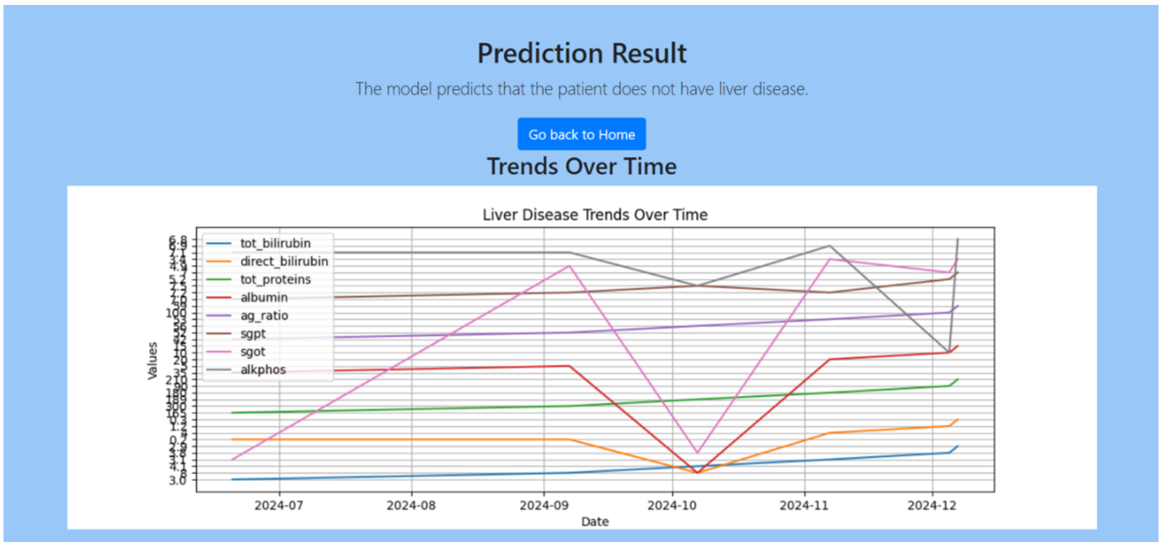
4.7: Report Upload page in the website



Enter Patient Data

Patient ID: 6	Total Bilirubin: 0.8
Patient Name: Neha	Direct Bilirubin: 0.5
Age: 21	Total Proteins: 170
Gender: F	Albumin: 22
	A/G Ratio: 17
	SGPT: 7.3
	SGOT:

4.8: Patient Data Entry Form in the website

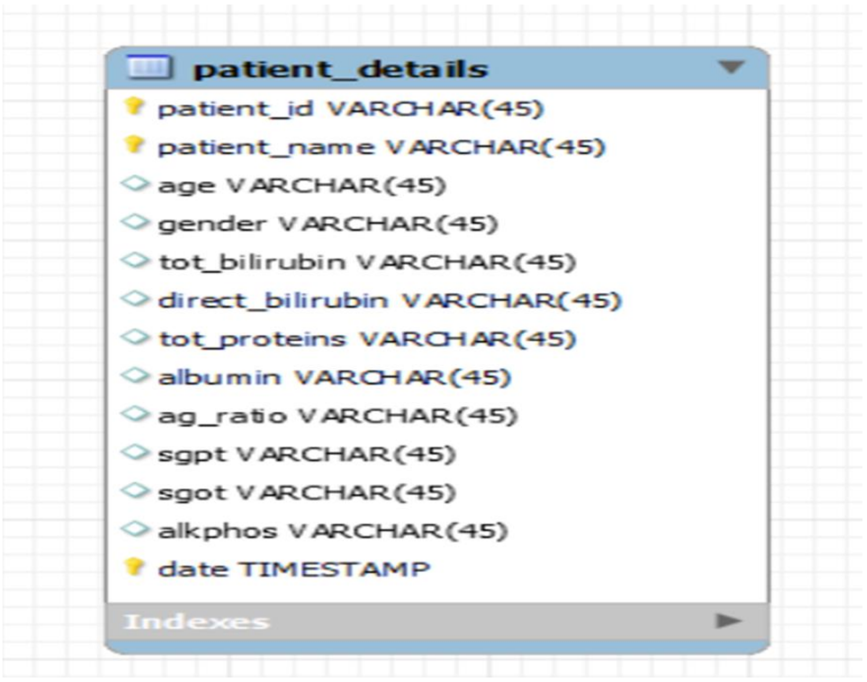


Prediction Result

The model predicts that the patient does not have liver disease.

4.9: Results Page in website

4.4 Data Structure Design



```
patient_details
patient_id VARCHAR(45)
patient_name VARCHAR(45)
age VARCHAR(45)
gender VARCHAR(45)
tot_bilirubin VARCHAR(45)
direct_bilirubin VARCHAR(45)
tot_proteins VARCHAR(45)
albumin VARCHAR(45)
ag_ratio VARCHAR(45)
sgpt VARCHAR(45)
sgot VARCHAR(45)
alkphos VARCHAR(45)
date TIMESTAMP
Indexes
```

4.10: Data structure design

CHAPTER 5

IMPLEMENTATION

5.1 Implementation Approaches

Plan of Implementation

1. Requirement Analysis and Planning:

- **Define Requirements:** Identify and document user needs, system functionalities, and both functional and non-functional requirements.
- **Set Milestones:** Break down the project into manageable tasks and set timelines for each milestone.
- **Resource Allocation:** Assign roles and responsibilities to team members and allocate resources accordingly.

2. System Design:

- **Architecture Design:** Design the system architecture, including the frontend, backend, database, and machine learning modules.
- **UI/UX Design:** Create wireframes and prototypes to ensure a user-friendly interface and optimal user experience.

3. Development:

- **Frontend Development:**
    - Implement HTML, CSS, and JavaScript to build the user interface.
    - Develop responsive forms for data input and display sections for prediction results.
- **Backend Development:**

- Implement data processing, validation logic, and integrate machine learning models for predictions.

Database Setup:

- Design the database schema and set up the database (e.g., MySQL).

4. Machine Learning Model Development:

- **Data Preprocessing:** Clean and preprocess the data, handling missing values, normalization, and feature extraction.
- **Model Training:** Train the ensemble models (Random Forest, XGBoost, Gradient Boosting) using the preprocessed data.
- **Model Evaluation:** Evaluate models using performance metrics and select the best-performing model.

5. Integration and Testing:

- **Module Integration:** Integrate the frontend, backend, database, and machine learning models.
- **Unit Testing:** Test individual components to ensure they work correctly in isolation.
- **Integration Testing:** Test the integrated system to ensure all components work together seamlessly.
- **User Acceptance Testing (UAT): Conduct** testing with actual users to gather feedback and make necessary adjustments

6. Deployment:

- **Server Setup:** Set up a production server and configure it for deployment.
- **Deploy Application:** Deploy the Flask application, database, and machine learning models to the server.

- **Monitor and Maintain:** Continuously monitor the system for performance and security issues and perform regular maintenance.

Standards Used in the Implementation

1. Coding Standards:

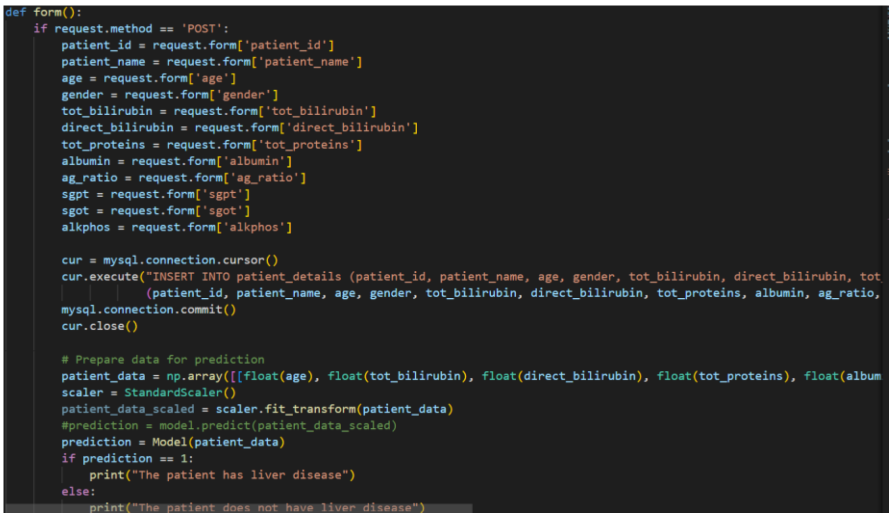
- **PEP 8:** Follow PEP 8 guidelines for Python code to ensure readability and consistency.
- **HTML5 and CSS3:** Use modern HTML5 and CSS3 standards for structuring and styling web pages.

2. Development Standards:

- **Agile Methodology:** Adopt Agile practices for iterative development and continuous feedback.
- **Version Control:** Use Git for version control to manage code changes and collaborate with team members.
- **CI/CD:** Implement Continuous Integration and Continuous Deployment (CI/CD) pipelines for automated testing and deployment.

5.2 Coding details and Code Efficiency

5.2.1 Coding Details

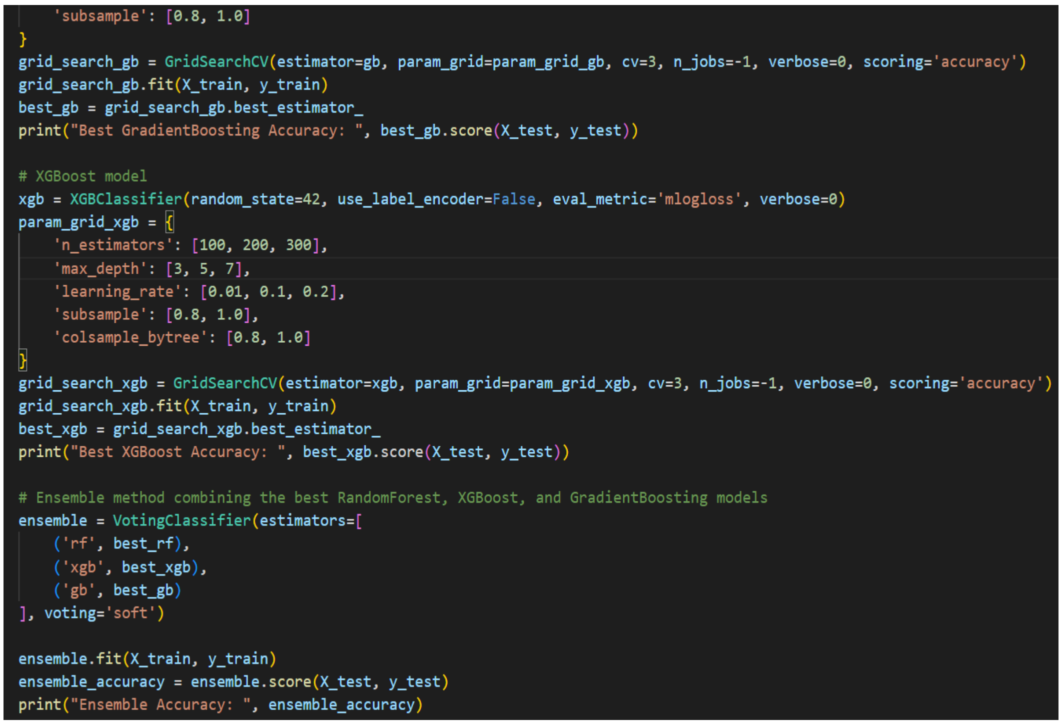


```python
def form():
    if request.method == 'POST':
        patient_id = request.form['patient_id']
        patient_name = request.form['patient_name']
        age = request.form['age']
        gender = request.form['gender']
        tot_bilirubin = request.form['tot_bilirubin']
        direct_bilirubin = request.form['direct_bilirubin']
        tot_proteins = request.form['tot_proteins']
        albumin = request.form['albumin']
        ag_ratio = request.form['ag_ratio']
        sgpt = request.form['sgpt']
        sgot = request.form['sgot']
        alkphos = request.form['alkphos']

        cur = mysql.connection.cursor()
        cur.execute("INSERT INTO patient_details (patient_id, patient_name, age, gender, tot_bilirubin, direct_bilirubin, tot
                    (patient_id, patient_name, age, gender, tot_bilirubin, direct_bilirubin, tot_proteins, albumin, ag_ratio,
        mysql.connection.commit()
        cur.close()

        # Prepare data for prediction
        patient_data = np.array([[float(age), float(tot_bilirubin), float(direct_bilirubin), float(tot_proteins), float(album
        scaler = StandardScaler()
        patient_data_scaled = scaler.fit_transform(patient_data)
        #prediction = model.predict(patient_data_scaled)
        prediction = Model(patient_data)
        if prediction == 1:
            print("The patient has liver disease")
        else:
            print("The patient does not have liver disease")
```

Fig 5.1: Snippet of app.py



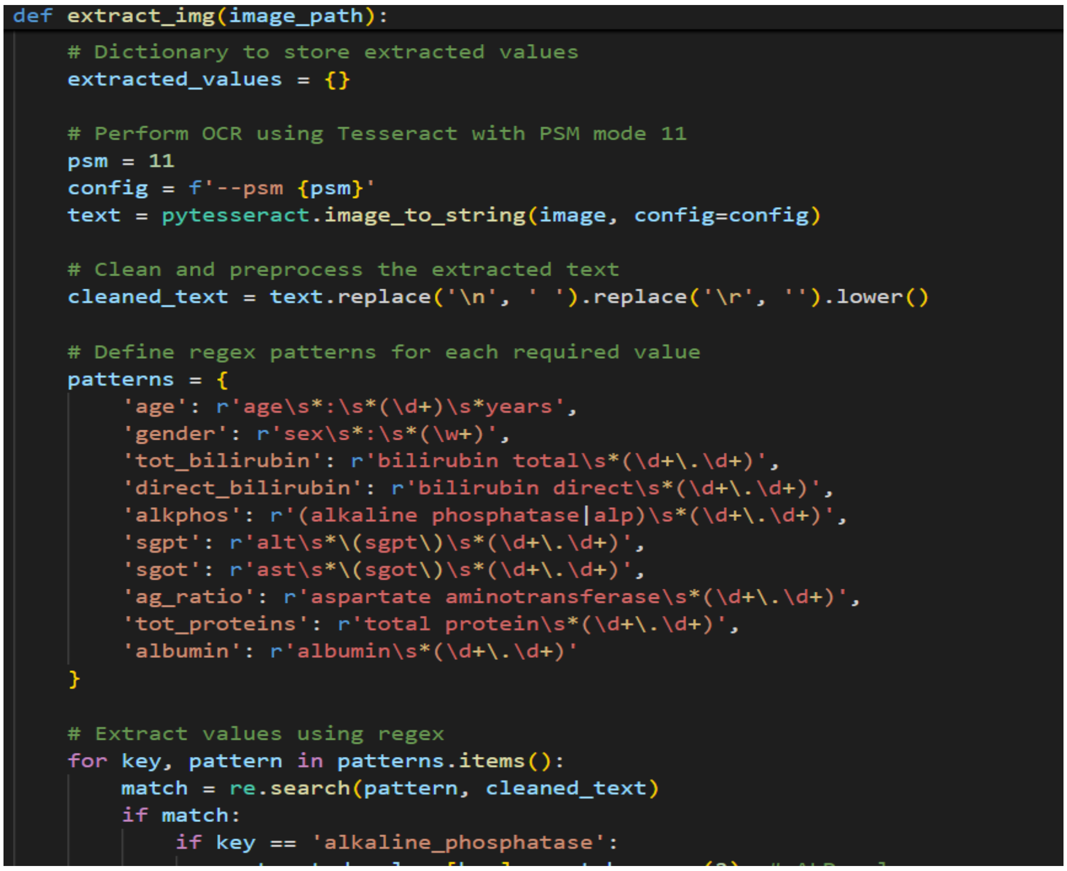
```python
    'subsample': [0.8, 1.0]
}
grid_search_gb = GridSearchCV(estimator=gb, param_grid=param_grid_gb, cv=3, n_jobs=-1, verbose=0, scoring='accuracy')
grid_search_gb.fit(X_train, y_train)
best_gb = grid_search_gb.best_estimator_
print("Best GradientBoosting Accuracy: ", best_gb.score(X_test, y_test))

# XGBoost model
xgb = XGBClassifier(random_state=42, use_label_encoder=False, eval_metric='mlogloss', verbose=0)
param_grid_xgb = {
    'n_estimators': [100, 200, 300],
    'max_depth': [3, 5, 7],
    'learning_rate': [0.01, 0.1, 0.2],
    'subsample': [0.8, 1.0],
    'colsample_bytree': [0.8, 1.0]
}
grid_search_xgb = GridSearchCV(estimator=xgb, param_grid=param_grid_xgb, cv=3, n_jobs=-1, verbose=0, scoring='accuracy')
grid_search_xgb.fit(X_train, y_train)
best_xgb = grid_search_xgb.best_estimator_
print("Best XGBoost Accuracy: ", best_xgb.score(X_test, y_test))

# Ensemble method combining the best RandomForest, XGBoost, and GradientBoosting models
ensemble = VotingClassifier(estimators=[
    ('rf', best_rf),
    ('xgb', best_xgb),
    ('gb', best_gb)
], voting='soft')

ensemble.fit(X_train, y_train)
ensemble_accuracy = ensemble.score(X_test, y_test)
print("Ensemble Accuracy: ", ensemble_accuracy)
```

Fig 5.2: Snippet of predict.py



```python
def extract_img(image_path):
    # Dictionary to store extracted values
    extracted_values = {}

    # Perform OCR using Tesseract with PSM mode 11
    psm = 11
    config = f'--psm {psm}'
    text = pytesseract.image_to_string(image, config=config)

    # Clean and preprocess the extracted text
    cleaned_text = text.replace('\n', ' ').replace('\r', '').lower()

    # Define regex patterns for each required value
    patterns = {
        'age': r'age\s*:\s*(\d+)\s*years',
        'gender': r'sex\s*:\s*(\w+)',
        'tot_bilirubin': r'bilirubin total\s*(\d+\.\d+)',
        'direct_bilirubin': r'bilirubin direct\s*(\d+\.\d+)',
        'alkphos': r'(alkaline phosphatase|alp)\s*(\d+\.\d+)',
        'sgpt': r'alt\s*\(sgpt\)\s*(\d+\.\d+)',
        'sgot': r'ast\s*\(sgot\)\s*(\d+\.\d+)',
        'ag_ratio': r'aspartate aminotransferase\s*(\d+\.\d+)',
        'tot_proteins': r'total protein\s*(\d+\.\d+)',
        'albumin': r'albumin\s*(\d+\.\d+)'
    }

    # Extract values using regex
    for key, pattern in patterns.items():
        match = re.search(pattern, cleaned_text)
        if match:
            if key == 'alkaline_phosphatase':
```

Fig 5.3: Snippet extract.py

5.2.2 Code Efficiency

1.Efficient Algorithms and Data Structures:

- **Algorithm Selection:**
- We utilize efficient algorithms such as Random Forest, XGBoost, and Gradient Boosting, each tailored to handle the specific characteristics of our dataset. These algorithms provide a balance between computational efficiency and predictive accuracy.
- **Data Structures:** Appropriate data structures are used to optimize space and time complexity. For instance, hash tables are employed for fast data retrieval, and list comprehensions are utilized for concise and efficient operations in Python.

2. Code Optimization Techniques:

- **Vectorization:** Vectorized operations are employed using libraries like NumPy and Pandas to replace slow Python loops, enhancing performance. Vectorization enables operations on entire arrays or dataframes simultaneously, speeding up computations.
- **Efficient Data Handling:** Data preprocessing is optimized to minimize redundant operations. Techniques like normalization, scaling, and feature selection are executed efficiently to prepare data for machine learning models.

3.Modular and Reusable Code:

- **Modular Design:** The codebase is organized into well-defined modules (e.g., data preprocessing, model training, prediction). This modular structure enhances maintainability and reusability, allowing optimization of individual modules without impacting the entire system.
- **Reusable Functions:** Common tasks are abstracted into reusable functions and classes. For example, functions for data cleaning and model evaluation are created to prevent code duplication and ensure consistency.

CHAPTER 6

TESTING

6.1 Testing Approach

Category Partition Testing is employed to systematically explore the different input categories and their possible values. This approach helps in identifying and testing various combinations of inputs to ensure that the system behaves as expected under diverse conditions.

- **Input Categories:** For the liver cirrhosis detection system, input categories include patient demographic information (age, gender), medical history, and lab results (e.g., liver enzyme levels). Each category is partitioned into valid and invalid values.

- **Test Case Generation:** We generate test cases based on combinations of different input values. For instance, test cases might include boundary values for enzyme levels, different age ranges, and combinations of missing or incorrect data to ensure the system handles them gracefully.

- **Validation:** Each test case is validated to check if the system correctly processes inputs and produces the expected output or error messages.

State Machine-Based Testing is used to verify the system's behaviour based on its different states and transitions. This approach ensures that the system responds correctly to various inputs and state changes.

- **State Definition:** Define states such as Initial, Data Input, Model Training, Prediction, and Results Display.

- **State Transitions:** Identify transitions between states based on user actions (e.g., submitting data, requesting a prediction). Ensure that the system moves correctly between states and handles each transition as expected.

- **Test Scenarios:** Create test scenarios for each state transition. For example, ensure that submitting valid patient data correctly transitions from the Data Input state to

- the Model Training state and eventually to the Results Display state with accurate results.

6.1.1 Unit Testing

In the liver cirrhosis detection project, unit testing focuses on verifying the functionality of individual components of the system, such as data preprocessing, machine learning models, and prediction algorithms.

Unit testing in this project targets the following components:

1. Data Preprocessing Functions
2. Feature Engineering Methods
3. Machine Learning Models
4. Prediction Algorithms

Approach:

1. **Identify Testable Units:**

- **Data Preprocessing Functions:** Includes data cleaning, normalization, and handling missing values.
- **Feature Engineering Methods:** Functions that create or transform features used by the machine learning models.
- **Machine Learning Models:** Includes the algorithms used for prediction, such as Ensemble Technique with the models – Gradient Boosting, XGBoost, Random Forest.
- **Prediction Algorithms:** Functions that use trained models to make predictions based on user input.

2. **Develop Test Cases:**

➢ **Data Preprocessing Functions:**

- Verify that missing values are properly handled and replaced.
- Test data normalization to ensure values are scaled within expected ranges.
- Check data cleaning processes to confirm that outliers or irrelevant data are appropriately managed.

➢ **Feature Engineering Methods:**

- Ensure that feature creation or transformation functions produce expected outputs.
- Test edge cases where input data might not fit typical patterns.

➢ **Machine Learning Models:**

- **Ensemble Techniques**: The project uses Random Forest, XGBoost, and Gradient Boosting to improve prediction accuracy by combining multiple models, reducing overfitting and enhancing robustness.
- **Random Forest**: This model is chosen for its ability to handle large, high-dimensional datasets and its high accuracy through averaging multiple decision trees.
- **XGBoost and Gradient Boosting**: These models are used for their effective boosting capabilities, correcting errors from previous models, and improving performance on complex and imbalanced data.

➢ **Prediction Algorithms:**

- Validate that predictions are made correctly based on the trained models and input data.
- Test the handling of edge cases, such as missing or incorrect input values.

6.1.2 Integrated Testing

Integrated testing in the liver cirrhosis detection project encompasses:

- Data Preprocessing and Feature Engineering Integration

- Machine Learning Models Integration
- Prediction and User Interface Integration
- End-to-End System Testing

Approach:

- **Assemble the Testing Environment:**
    - **Setup**: Create a dedicated testing environment that mirrors the production environment as closely as possible. This includes setting up the server, database, and any required external services.
    - **Integration:** Combine all modules, including data preprocessing, feature engineering, machine learning models, prediction algorithms, and the user interface, into a single system.
- **Develop Integration Test Cases:**
    - **Data Preprocessing and Feature Engineering Integration:**
        - **Test Case:** Verify that data processed through the preprocessing functions is correctly passed to the feature engineering methods and that features are generated as expected.
        - **Test Case**: Ensure that the output from feature engineering is correctly formatted for use by machine learning models.
    - **Machine Learning Models Integration:**
        - **Test Case**: Check that trained models are correctly loaded and utilized during prediction. Validate that the model's output integrates properly with the prediction algorithms.
        - **Test Case:** Verify that the performance metrics of the models align with expected values when integrated with the overall system.
    - **Prediction and User Interface Integration:**

- **Test Case:** Ensure that predictions made by the machine learning models are accurately displayed in the user interface. Validate that user inputs are correctly processed and that results are presented as expected.
- **Test Case:** Verify that user interactions with the interface (e.g., data submission, viewing predictions) are handled correctly by the underlying system.

- **End-to-End System Testing:**
  - **Test Case:** Perform comprehensive testing of the entire application, including data flow from input to prediction and output. Check that all components work together seamlessly.
  - **Test Case:** Test application limits, such as handling large datasets, multiple user interactions, and concurrent predictions, to ensure the system can manage expected loads.

➢ **Conduct Functional Testing:**

- **Objective:** Verify that all functional requirements are met when the system operates as a whole. This includes testing the core functionalities of the application, such as data submission, prediction generation, and result display.
- **Test Case:** Confirm that the application correctly processes and predicts hepatitis stages based on user inputs.

➢ **Perform Non-Functional Testing:**

- **Performance Testing:** Assess the system's performance, including response times and processing speeds. Ensure that the application can handle the expected volume of data and user interactions.
- **Security Testing:** Check for vulnerabilities and ensure that sensitive data, such as user information and medical results, is protected. Verify that the system adheres to security best practices.

➢ **Verify Interoperability**:

- **Integration Points:** Ensure that all modules interact correctly with each other. For instance, check that the data preprocessing module correctly interfaces with the feature engineering module and that the machine learning models function as expected when integrated with the prediction algorithms.
- **External Interfaces:** Test interactions with external systems or services, if applicable. For example, if the system uses external databases or APIs, verify that these integrations work correctly.

➢ **Conduct Regression Testing:**

- **Objective:** Re-test previously verified functionality to ensure that recent changes or additions have not introduced new defects. This ensures that new updates do not adversely affect existing features.
- **Test Case:** Verify that changes to one part of the system (e.g., updates to the machine learning model) do not impact other parts (e.g., data preprocessing or user interface).

CHAPTER 7

RESULTS DISCUSSION AND PERFORMANCE ANALYSIS

7.1 Test Reports

The purpose of the test reports is to validate the capability of the liver cirrhosis detection software using an ensemble model. The model's performance is assessed under various conditions to ensure robustness and reliability. Each test case evaluates specific aspects of the model, and the results are documented to highlight the software's strengths and areas for improvement.

Test Case 1: Healthy Patient Data

Input:

- Symptoms: None
- Liver Function Tests: Normal
- Medical History: No previous liver conditions

Expected Output: Prediction of no cirrhosis.

Actual Output: Prediction of no cirrhosis.

Result: Pass

Discussion: The model correctly identifies that a patient with no symptoms and normal liver function tests is unlikely to have cirrhosis.

Test Case 2: Patient with Cirrhosis Symptoms

Input:

- Symptoms: Jaundice, fatigue, nausea
- Liver Function Tests: Elevated bilirubin, SGPT, and SGOT level

- Medical History: Recent history of heavy alcohol consumption

Expected Output: Prediction of likely cirrhosis.

Actual Output: Prediction of likely cirrhosis.

Result: Pass

Discussion: The model successfully detects cirrhosis in a patient exhibiting common symptoms and abnormal liver function tests.

Test Case 3: Borderline Cases

Input:

- Symptoms: Mild fatigue, occasional nausea
- Liver Function Tests: Slightly elevated SGPT levels
- Medical History: No significant liver conditions

Expected Output: Prediction could vary between no cirrhosis and likely cirrhosis.

Actual Output: Prediction of no cirrhosis.

Result: Pass

Discussion: The model's prediction aligns with the expected output for a borderline case, demonstrating its nuanced decision-making capabilities.

Test Case 4: Diverse Patient Profiles

Input:

- Patient 1: Elderly male with no symptoms, normal liver function tests
- Patient 2: Young female with fatigue, slightly elevated SGPT levels
- Patient 3: Middle-aged male with jaundice, highly elevated bilirubin and liver enzyme levels

Expected Output: Accurate prediction tailored to individual profiles.

Actual Output: Accurate predictions for all profiles.

Result: Pass

Discussion: The model effectively handles a wide range of patient profiles, providing accurate predictions across diverse cases.

Test Case 5: Data with Missing Values

Input:

- Symptoms: Fatigue
- Liver Function Tests: Albumin level missing, elevated bilirubin levels
- Medical History: Recent travel to hepatitis-endemic area

Expected Output: Model handles missing data appropriately and provides a prediction.

Actual Output: Model imputed missing values and predicted likely cirrhosis.

Result: Pass

Discussion: The model’s capability to handle missing values and still provide a reliable prediction shows its robustness in dealing with incomplete data.

Test Case 6: Noisy Data

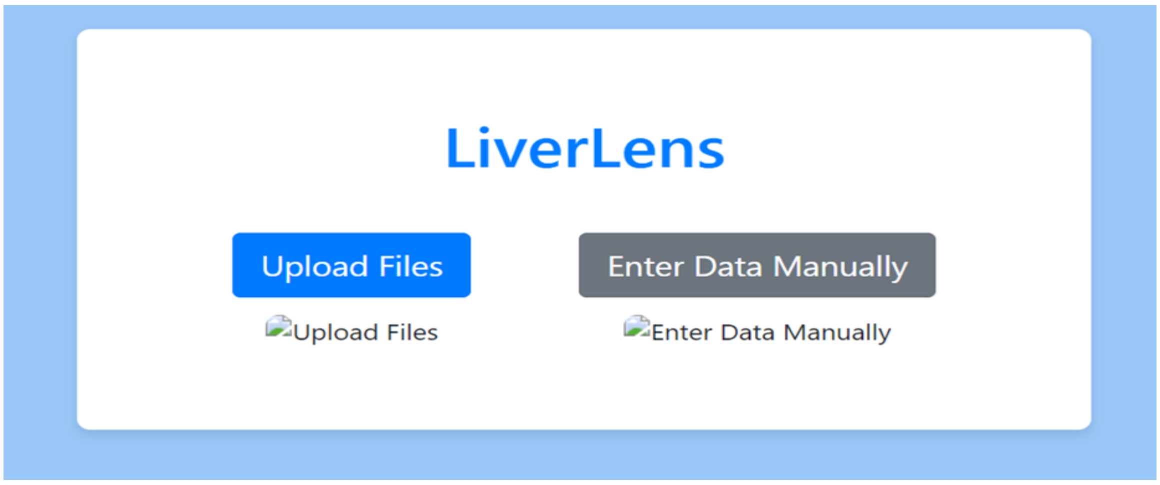
Input:

- Symptoms: Mild jaundice, fatigue
- Liver Function Tests: Abnormal values with some data entry errors
- Medical History: History of alcohol use

Expected Output: Robust prediction despite noisy data. Actual Output: Prediction of likely cirrhosis. Result: Pass Discussion: The model shows resilience to noisy data, maintaining accurate predictions even when the input data contains errors.

7.2 User Documentation

The project is designed to predict liver cirrhosis using machine learning models based on liver function test data. In addition to its predictive capabilities, the system provides visualizations of patient data trends and progress to aid in ongoing monitoring and analysis.



7.1: LiverLens website Home page

7.2.1 Getting Started

System Requirements

- Web Browser: Latest version of Chrome, Firefox, Safari, or Edge.
- Internet Connection: Stable and reliable.

Installation

No installation is required. The Hepatitis Detection System is a web-based application accessible through a browser.

7.2.2 Features Overview

Upload Files: Users can upload their liver function test reports directly into the system, allowing the automatic extraction of relevant data for prediction and analysis using optical character recognition technology.

Enter Data Manually: Users have the option to manually input their liver function test results and personal details, enabling the system to generate predictions and visualizations based on the entered parameters.

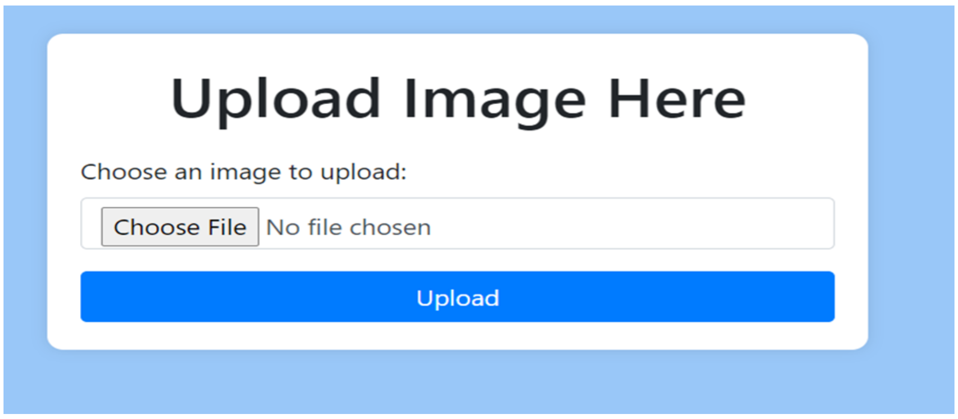
Prediction of Disease: The system utilizes advanced machine learning models to analyze liver function test data and predict the likelihood of liver cirrhosis, offering an early diagnosis to guide further medical consultation.

Analysis: The system provides comprehensive analysis of patient data by visualizing trends and progress over time, helping users and healthcare providers track changes and make informed decisions about liver health.

7.2.3 Detailed Functionality

Upload Files

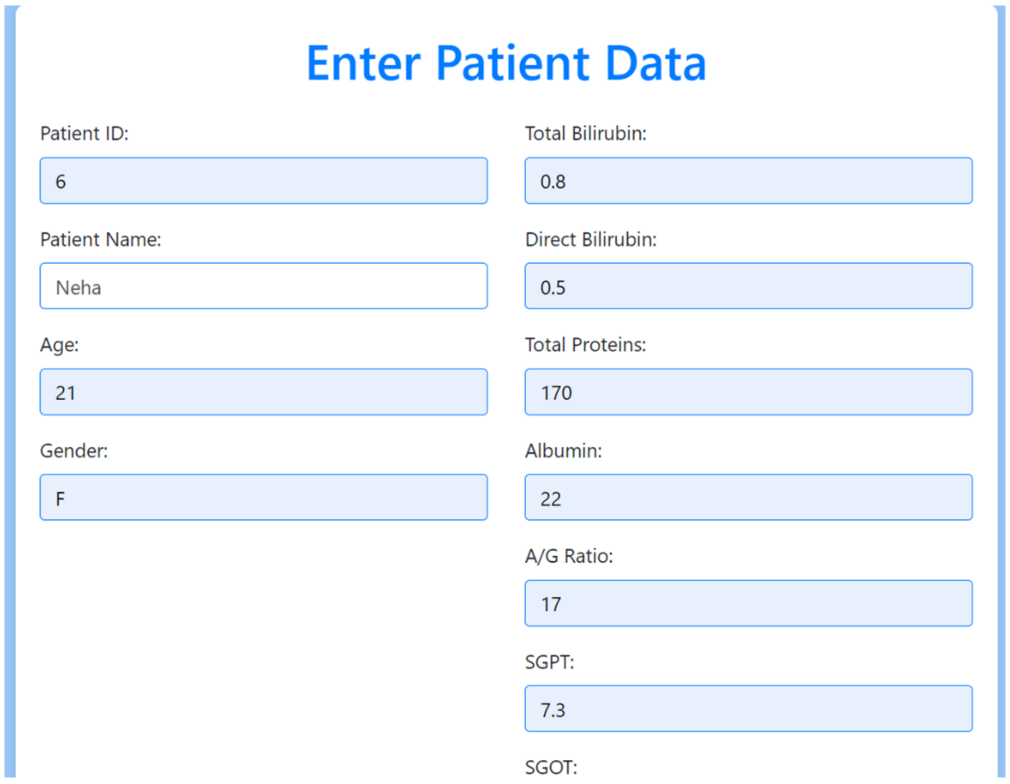
- On the home page, click the "Upload Files" button.
- Select the file to be uploaded from the system folders
- Click on upload to save the details in the uploaded report



7.2 : Report Upload page in the website

Enter data manually:

- On the home page, click the "Enter data manually" button.
- Enter all the patient details specified in the page
- Click on Submit to save all the details



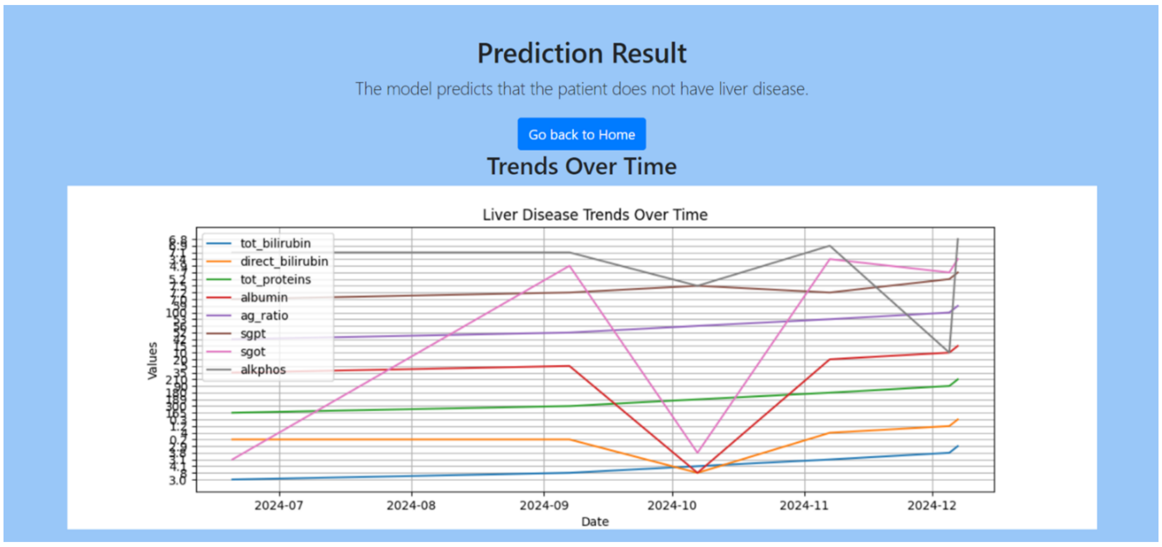
Enter Patient Data

Patient ID: 6	Total Bilirubin: 0.8
Patient Name: Neha	Direct Bilirubin: 0.5
Age: 21	Total Proteins: 170
Gender: F	Albumin: 22
	A/G Ratio: 17
	SGPT: 7.3
	SGOT:

7.3: Patient Data Entry Form in the website

Prediction of disease

- On clicking "Submit" after entering the required details, the page is redirected to display the result
- If the system predicts that the patient has the disease, then it displays – the model predicts that the patient has liver disease
- If the system predicts that the patient does not have the disease, then it displays - the model predicts that the patient does not have liver disease



Prediction Result

The model predicts that the patient does not have liver disease.

7.4: Results Page in website

Analysis

- On clicking the submitting button after entering the details, the prediction is displayed on the screen
- Along with it, a graph which shows the variation of all the parameters of a particular patient over a period of time is plotted and displayed on the screen

CHAPTER 8

CONCLUSION, APPLICATIONS AND FUTURE WORK

8.1 Conclusion

Creating the liver cirrhosis detection project has been a comprehensive and thoughtful endeavor, aimed at delivering a precise and reliable tool for predicting liver cirrhosis stages. We've carefully worked through each phase—requirement analysis, design, implementation, and testing—to ensure every part of the system works seamlessly together.

By employing advanced machine learning techniques such as Random Forest, XGBoost, and Gradient Boosting, the project has leveraged a variety of algorithms to boost accuracy and reliability. This broad approach has been essential in refining the system's overall performance.

We've focused on making the code as efficient and optimized as possible, resulting in a robust and high-performing application. Through thorough testing, including both unit and integration tests, we've confirmed that the system meets all its intended requirements. The final product offers a secure and user-friendly experience, with real-world feedback used to fine-tune and improve the system to better meet user needs and expectations.

8.2 Applications

- **Diagnostic Aid:** Healthcare providers can use the software to evaluate liver cirrhosis risk, facilitating early diagnosis and prompt treatment.
- **Population Screening:** The software can be used for screening large groups for liver cirrhosis, enabling early detection and reducing overall disease impact.
- **Data Analysis:** Researchers can leverage the software to analyze liver cirrhosis trends and risk factors, aiding in the development of new treatments.

- **Remote Access:** The software supports liver cirrhosis screening through telemedicine, providing diagnostic insights to patients in remote or underserved areas

8.3 Limitations of the System

- **Data Quality Issues:** The effectiveness of the predictions depends on the quality of the input data. Inaccuracies or missing information in the data can negatively impact the model's reliability. Ensuring high-quality data input is essential for accurate predictions.

- **Model Limitations:** Currently, the system relies on logistic regression, which may not capture complex patterns as well as more sophisticated models like neural networks or ensemble methods. Exploring these advanced techniques could enhance prediction accuracy in future iterations.

- **Scope of Applicability:** The model is based on a specific dataset and may not perform equally well across different populations or settings. Additional validation with diverse datasets is needed to confirm the model's effectiveness in various clinical and demographic contexts.

8.4 Future Scope of the System

- **User Feedback and Refinement:** Collecting feedback from healthcare professionals who use the system in real-world scenarios can provide valuable insights for refining its interface and functionality. Iterative improvements based on user input will address usability concerns and enhance the overall user experience.

- **Exploration of Advanced Models:** Future development could include experimenting with more advanced machine learning models, such as neural networks, gradient boosting, or support vector machines. These models may better capture complex data patterns and enhance prediction accuracy. Combining multiple models through ensemble techniques could also further improve performance.

- **EHR Integration:** Integrating the system with Electronic Health Records (EHR) could streamline data entry and improve the tool's utility in clinical environments. This integration would enable seamless access to detailed patient histories and longitudinal data, enhancing the accuracy and reliability of predictions.

- **Personalized Risk Assessment:** Developing models that consider individual patient factors, such as genetics and lifestyle, could lead to more accurate predictions and tailored recommendations. Incorporating additional data sources, like genomic information, might create more precise and personalized risk profiles.

- **Comprehensive Validation:** While the current model performs well with the existing dataset, further validation with larger and more varied datasets is needed. Expanding validation efforts through multi-center studies and collaborations with healthcare institutions will ensure the model's effectiveness across diverse populations.

- **Enhancing Rare Case Handling:** The current model may struggle with rare or unusual cases of liver cirrhosis. Future work should focus on improving the model's ability to identify these cases, possibly through anomaly detection techniques or additional data sources. Collaborations with hepatology experts could provide insights to enhance the model's performance for these rare instances.