**Project Title:**

Revolutionizing Liver Care: Predicting Liver Cirrhosis Using Advanced Machine Learning Techniques

**Team Name:**

LTVIP2025TMID35624 - Tech Squad

**Team Members:**

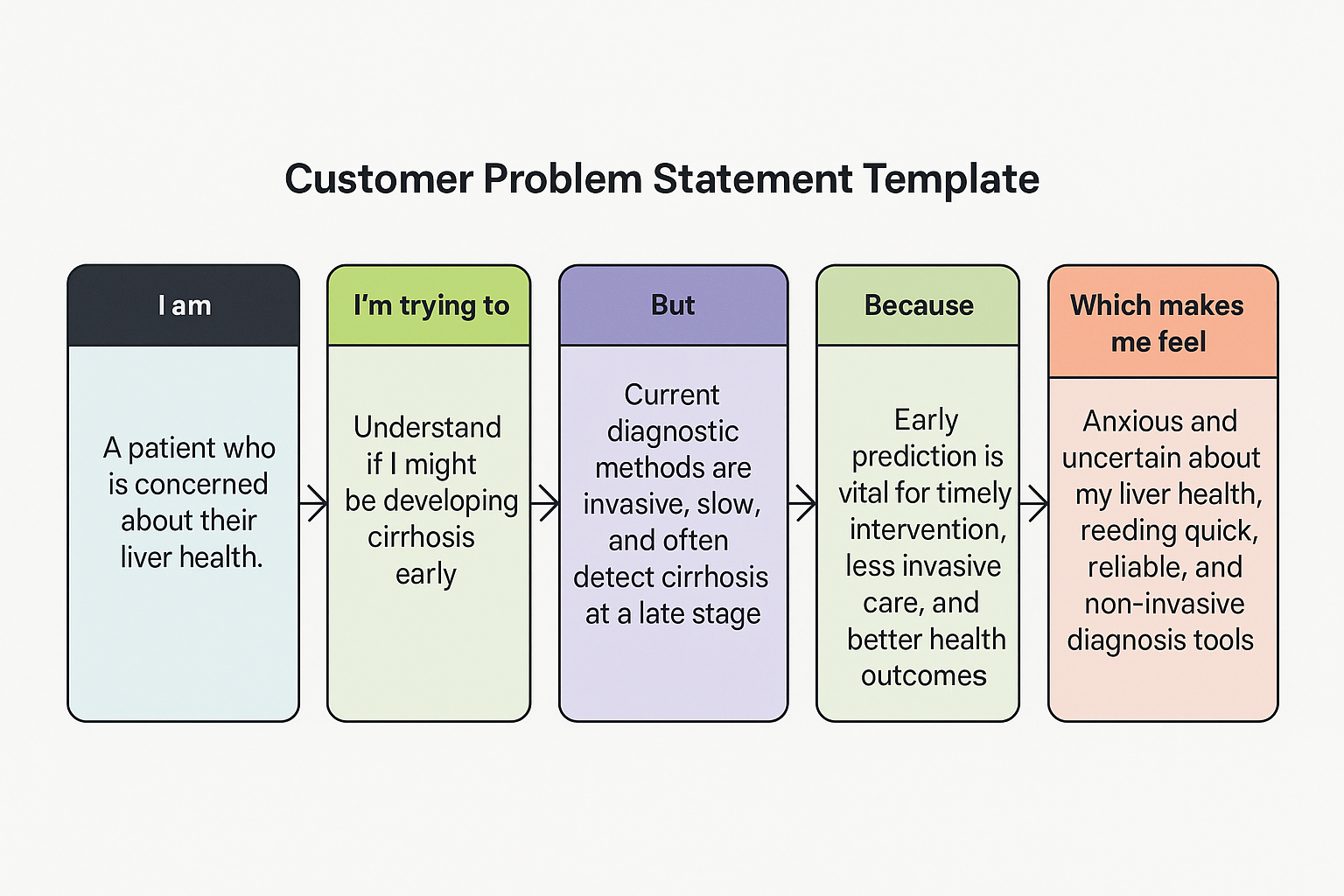
1. Thathireddy Eswar Reddy(Team Lead)
2. Dasari Gowthami
3. Golla Navitha
4. Baripireddy Sai Chandan Reddy
5. D Harsha Vardhan
6. **Phase 1: Brainstorming & Ideation**

**1.1 Objective:**

The primary objective is to enhance the early detection and management of liver cirrhosis by implementing advanced machine learning techniques, ensuring timely and accurate predictions.

**1.2 Problem Statement:**

Liver cirrhosis is a life-threatening condition that often goes undetected until it reaches an advanced stage. Early diagnosis is critical for effective treatment and improved survival rates, yet traditional diagnostic methods are invasive, time-consuming, and costly. This project aims to transform liver care by applying advanced machine learning techniques to predict liver cirrhosis from non-invasive clinical and laboratory data. By uncovering hidden patterns in patient data, the system provides accurate, early-stage predictions-enabling timely interventions and personalized healthcare solutions.

****

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Problem Statement (PS)** | **I am (Customer)** | **I’m trying to** | **But** | **Because** | **Which makes me feel** |
| PS-1 | A patient who is concerned about their liver health | Understand if I might be developing cirrhosis early | Current diagnostic methods are invasive, slow, and often detect cirrhosis at a late stage | Early prediction is vital for timely intervention, less invasive care, and better health outcomes | Anxious and uncertain about my liver health, needing quick, reliable, and non-invasive diagnosis tools |
| PS-2 | A middle-aged person with a family history of liver disease | Know my risk of developing liver cirrhosis early | I don’t get regular checkups because the tests are expensive and invasive | Knowing my risk early can help me make lifestyle changes and seek medical attention in time | Worried and helpless due to lack of affordable and accessible diagnosis |
| PS-3 | A healthcare provider managing multiple liver patients | Identify early-stage cirrhosis in patients efficiently | Traditional diagnosis takes time and often misses early signs | Early and accurate predictions can improve patient care and outcomes | Frustrated and overwhelmed due to delays and limited diagnostic precision |
| PS-4 | A patient with symptoms like fatigue and jaundice | Understand if it could be liver cirrhosis or something else | Symptoms are vague and doctors need invasive tests to confirm | A clear, data-driven prediction could reduce uncertainty and speed up diagnosis | Anxious and confused, needing clarity and fast results without invasive procedures |
| PS-5 | Someone recovering from alcohol use disorder | Monitor liver health and detect possible cirrhosis | Current tools don’t offer personalized monitoring or prediction | Early alerts could motivate me to stay healthy and seek timely help | Insecure and unsure about my progress and potential complications |
| PS-6 | A caregiver of an elderly person with chronic liver problems | Get early warnings about disease progression | We only get updates during scheduled visits and after symptoms worsen | Predictive tools can help us manage care better and avoid emergencies | Stressed and unprepared for sudden health deterioration, needing continuous support |

**1.3 Proposed Solution:**

We propose building a machine learning (ML)–based predictive model to detect liver cirrhosis in its early stages using non-invasive, routinely available clinical and laboratory data. The model will assist in identifying high-risk individuals and support early diagnosis and proactive intervention.

**1.3.1 Data Source:**

Source: <https://www.kaggle.com/datasets/bhavanipriya222/liver-cirrhosis-prediction>

Description: This dataset includes structured, anonymized patient records featuring:

* **Demographics**: Age, Gender, Location
* **Medical History**: Alcohol consumption, Hepatitis infections, Diabetes
* **Laboratory Tests**: Blood counts, liver function tests, lipid profiles
* **Clinical Indicators**: Blood pressure, obesity status, family history

**1.3.2 Solution Workflow**

1. Data Collection & Preprocessing

* Perform preprocessing steps such as:
  + Handling missing values
  + Encoding categorical variables (e.g., gender, yes/no lifestyle factors)
  + Normalizing continuous features
  + Feature selection to retain only the most relevant attributes
* Data Exploration and Preprocessing
* Univariate Analysis: Histograms were plotted for numerical features.
* Bivariate Analysis: Scatter plots and pair plots explored relationships between features.
* Outlier Handling: Outliers were detected and managed using the IQR method.

2. Model Development

* Train and evaluate multiple machine learning algorithms, including:
  + Random Forest
  + Naïve Bayes
  + XG Boost
  + Logistic Regression CV
  + Random Forest
  + Support Vector Classifier
  + Ridge Classifier
  + Logistic Regression
  + K-Nearest Neighbors (KNN)
* Apply cross-validation and hyperparameter tuning to optimize performance.
* Use evaluation metrics: Accuracy, Precision, Recall, F1-score, and Confusion Matrix to validate the models.

3.Model Selection  
 We will select and optimize the best model based on performance metrics to achieve the highest accuracy.

4.Prediction

* Generate predictions in the form of: Binary classification (Cirrhosis: Yes/No)

5. Interface Design

* Building a user-friendly web or mobile platform using tools like: HTML/ CSS +JavaScript(Frontend),Flask (Backend)
* Users can input lab results and receive the Prediction results.

**1.4 Target Users**

|  |  |
| --- | --- |
| **Target Group** | **Needs Addressed** |
| Patients | Want early awareness of liver damage risk through non-invasive tests |
| Clinicians | Need decision support tools to detect cirrhosis without complex procedures |
| Hospitals & Labs | Require cost-effective screening tools for large-scale use |
| Public Health Units | Can use the system for preventive screening and awareness programs |
| Healthcare Startups | Can integrate this tool into platforms for telemedicine or remote diagnosis |

* 1. **Expected Outcome:**

1. A trained machine learning model capable of predicting liver cirrhosis with high accuracy (above benchmark metrics like 85–90%).
2. A prototype application/interface for real-time predictions based on user-input health data.
3. Model explainability integrated using SHAP or similar tools to improve trust and usability in healthcare settings.
4. Improved early detection rates, enabling timely treatment and reducing hospital burden from advanced liver disease cases.
5. Better access to liver health assessments for under-resourced areas lacking specialized diagnostic tools.
6. Documentation, evaluation, and future scope roadmap for clinical integration and research expansion.
7. **Phase 2: Requirement Analysis**

**2.1 Prerequisites and System Requirements**

Before setting up the local development environment, ensure your system meets the following requirements:

|  |  |  |
| --- | --- | --- |
| **Component** | **Requirement** | **Purpose** |
| Python | Version 3.7+ | Runtime for Flask app and ML libraries |
| Git | Latest version | Version control and repository cloning |
| Operating System | Windows/macOS/Linux | Cross-platform compatibility |
| RAM | Minimum 4GB | Machine learning model operations |
| Storage | 1GB free space | Project files and virtual environment |

**2.2. Technical Requirements**

**2.2.1. Languages & Libraries:**

* Python: For machine learning model development and data preprocessing
* Flask : Synchronized, Lightweight Python backend for model integration and deployment
* Pandas, NumPy: Data analysis and manipulation
* Scikit-learn: ML model building, evaluation, and preprocessing
* XG Boost, KNN, etc.: For advanced ML algorithm support
* Matplotlib & Seaborn: For data visualization
* HTML/CSS + JavaScript: For building the frontend user interface

**2.2.2. Development Tools & Platforms:**

The system supports multiple development environments for model training and code development:

|  |  |  |
| --- | --- | --- |
| **Environment** | **Use Case** | **Setup Instructions** |
| Anaconda Navigator | Local Jupyter notebooks | Download from anaconda.com |
| Google Colab | Cloud-based training | Access via colab.research.google.com |
| VSCode | Integrated development | Install Jupyter and Python extensions |

**2.2.3** **Dataset Requirements**

The system requires the healthcare dataset from Kaggle:

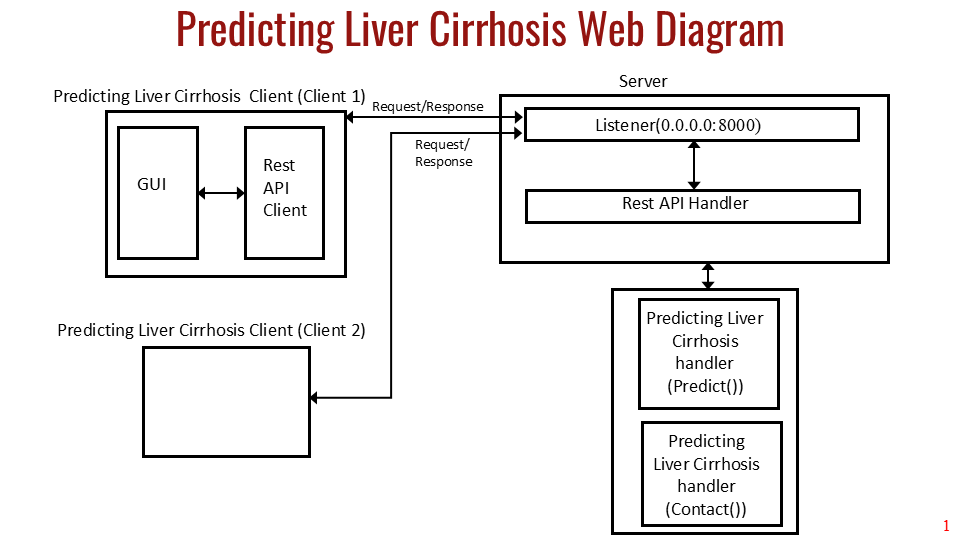
* Source: <https://www.kaggle.com/datasets/bhavanipriya222/liver-cirrhosis-prediction>
* File: HealthCareData.xlsx
* Purpose: Training data for the machine learning model

**2.3 Functional Requirements**

1. Upload or input patient data (Age,Gender,Place,Duration\_of\_alcohol\_consumption,Quantity\_of\_alcohol\_consumption,Type\_of\_alcohol\_consumed,Blood\_pressure,Obesity,Family\_history\_of\_cirrhosis\_hereditary,Hemoglobin,PCV,RBC,MCV,MCH,MCHC,Total\_Count,Polymorphs,Lymphocytes,Monocytes,Eosinophils,Basophils,Platelet\_Count,Direct,Indirect,Total\_Protein,Albumin,Globulin,AL\_Phosphatase,SGOT\_AST,USG\_Abdomen,Outcome etc.)
2. Run machine learning models to generate a prediction (Cirrhosis: Yes/No)
3. Deliver a clean, responsive web interface accessible on multiple devices

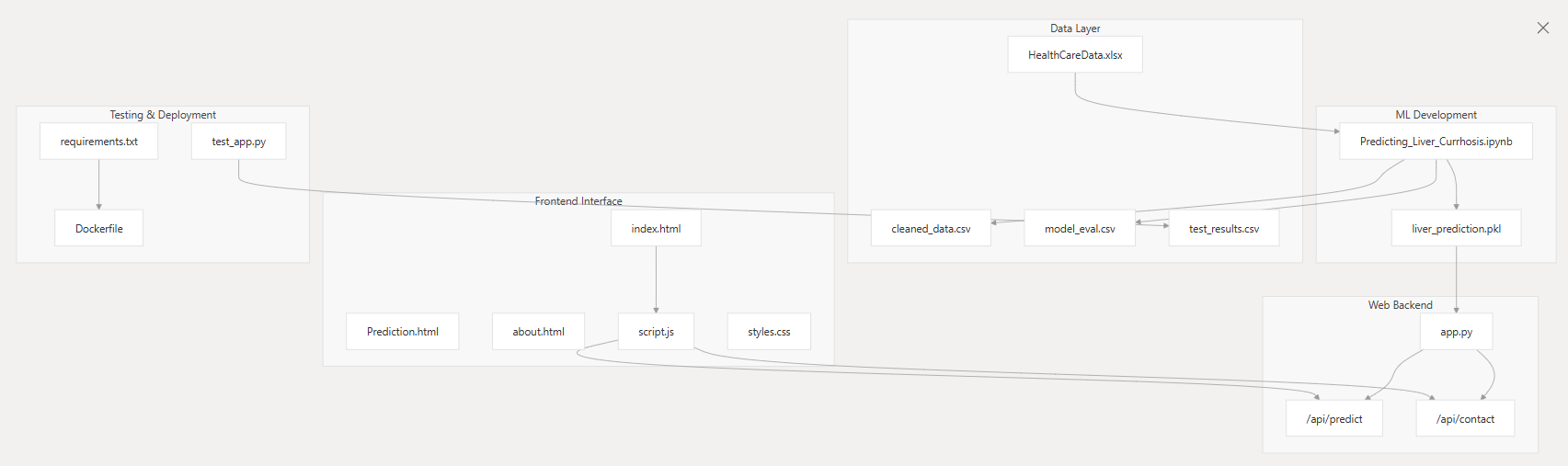
**2.4 Constraints & Challenges**

1. Data Quality: Missing or inconsistent values may reduce model reliability. Preprocessing is essential.
2. Model Bias: Overfitting due to small or imbalanced dataset; requires tuning and validation.
3. Interpretability: Medical predictions must be explainable; black-box models may not be trusted by clinicians.
4. Deployment Scalability: Real-time performance and scaling to many users may require cloud infrastructure.
5. Security & Privacy: Medical data is sensitive—ensuring data protection and compliance is crucial.
6. Limited Dataset Size: The Kaggle dataset may not cover all demographics or rare cases, limiting generalizability.
7. **Phase-3: Project Design**
   1. **System Architecture Diagram**





The application follows a layered architecture pattern with distinct data processing, machine learning, and web service components:

****

**3.1.2 Machine Learning Model**

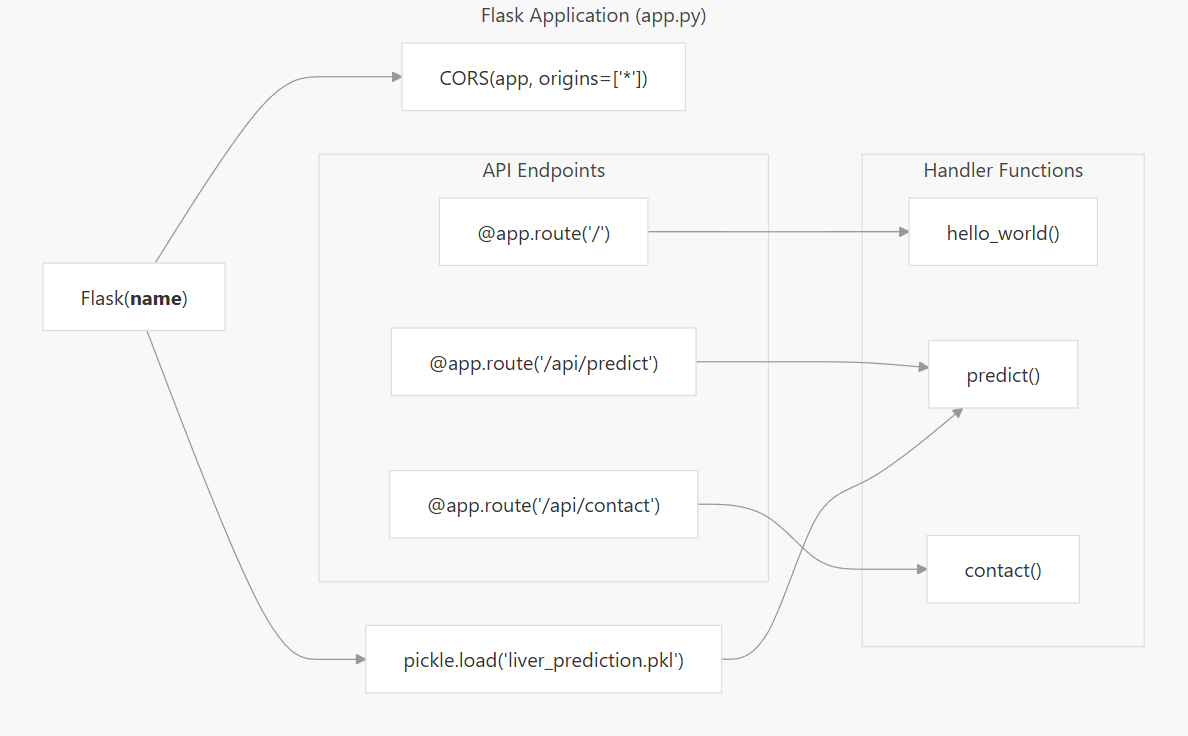
The ML system implements multiple algorithms for liver cirrhosis prediction, with model selection based on performance metrics:

****

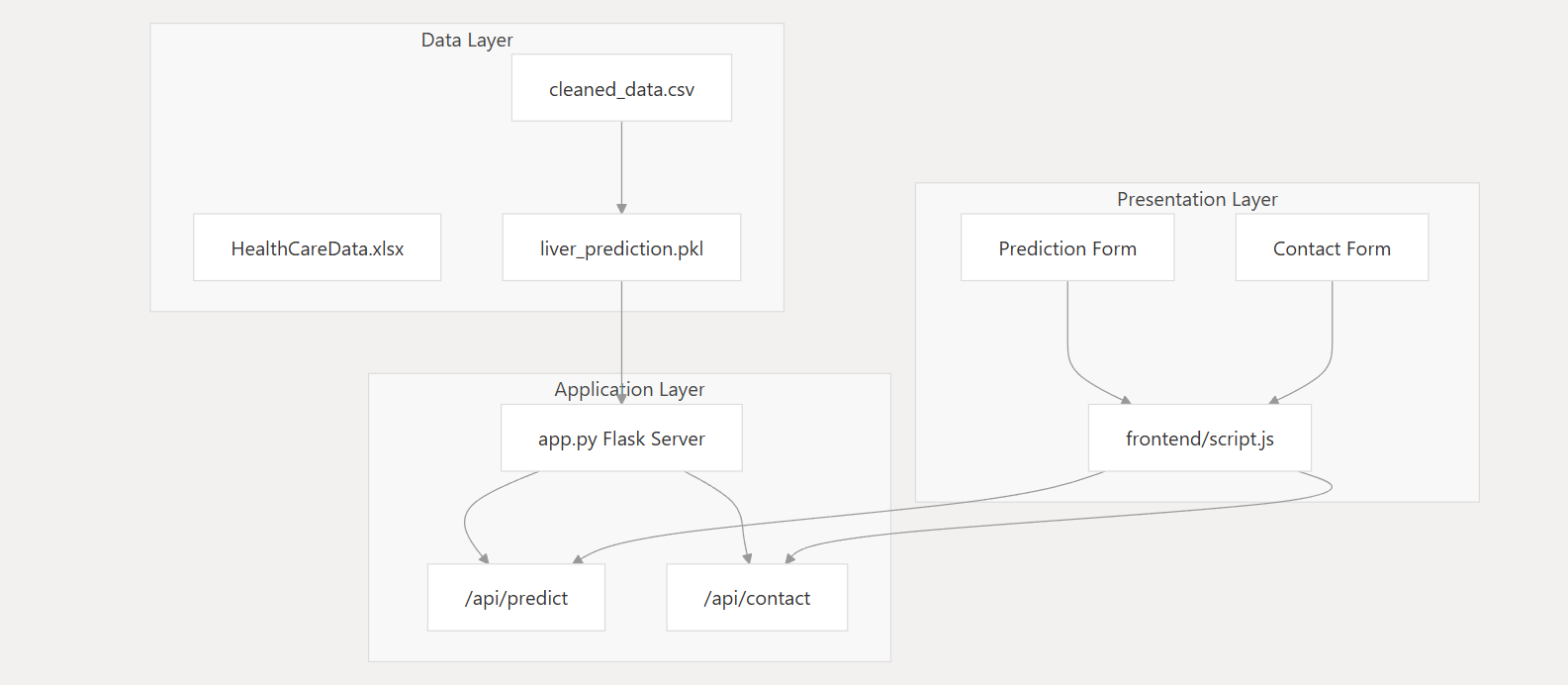
The model accepts numerical feature arrays and returns binary classification results (0 = No Liver Disease, 1 = Liver Disease).

**3.1.3 Flask Backend API**

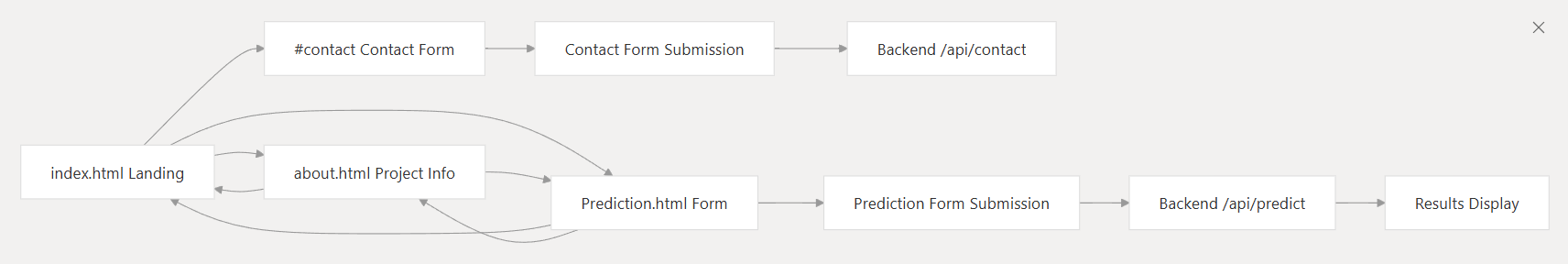
The core application server is implemented in app.py as a Flask application with CORS enabled for cross-origin requests. The server loads a pre-trained ML model and exposes two main API endpoints.

****

**3.1.4 High-Level Architecture**

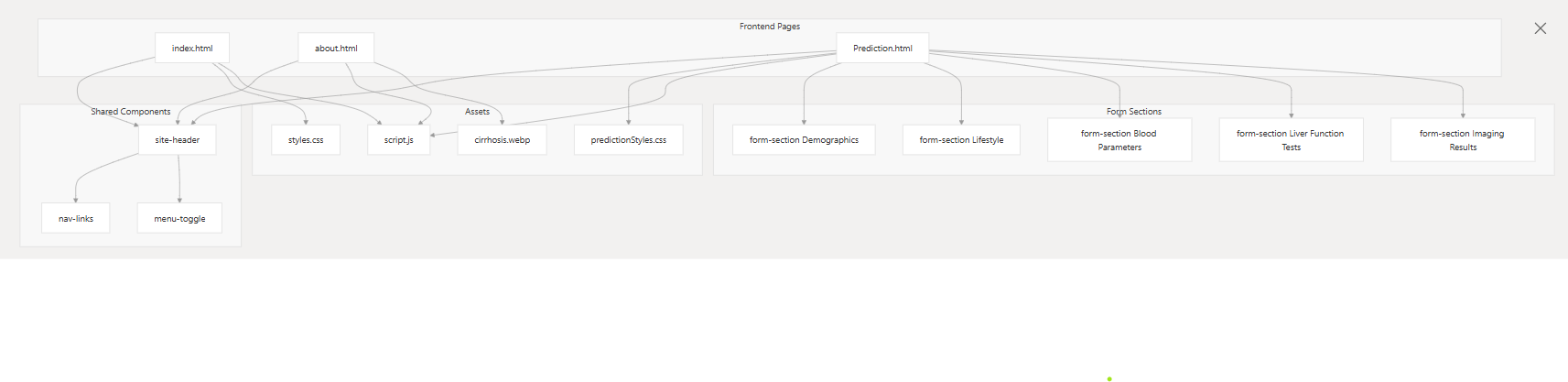
****

* 1. **User Flow**

****

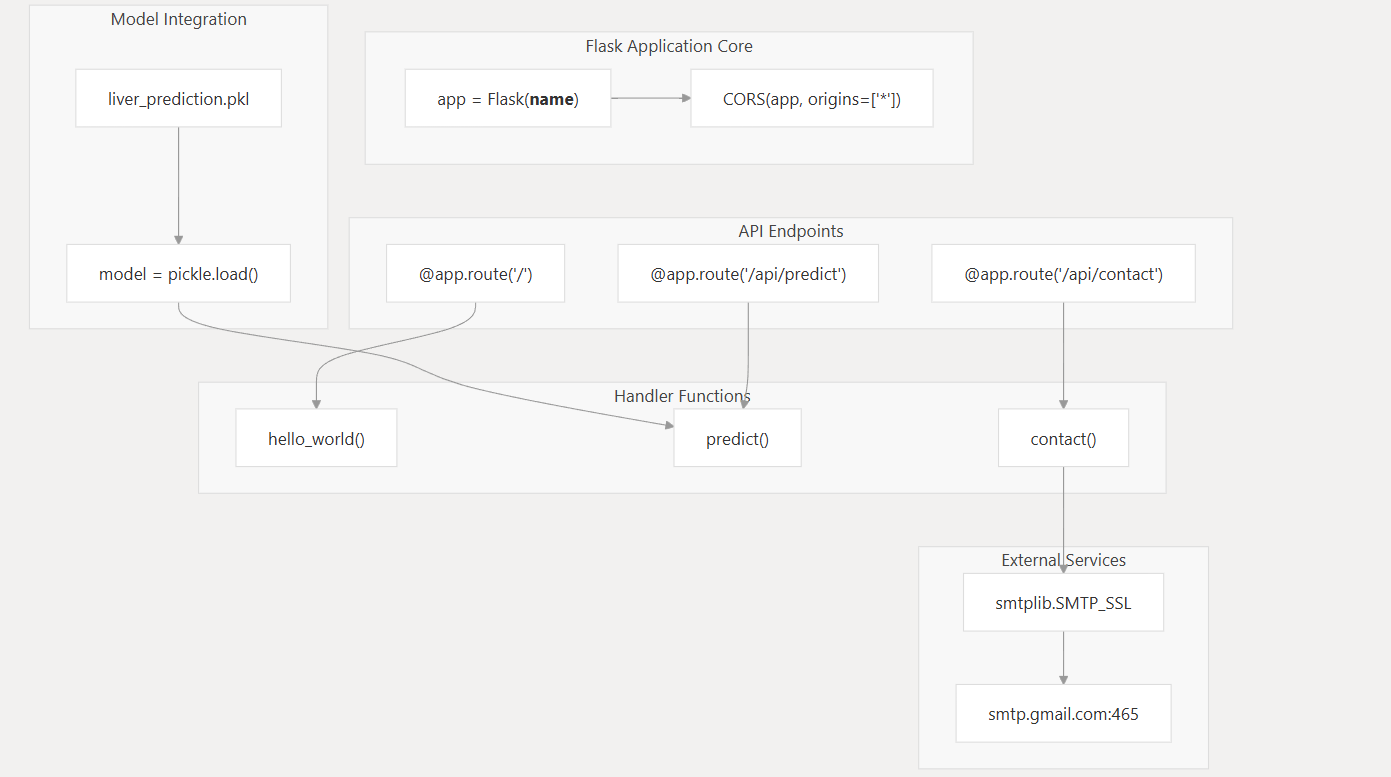
* 1. **UI/UX Considerations**

**3.3.1 Frontend Architecture Overview**



**3.3.2 Application Architecture**

The backend is implemented as a Flask web application with the following key components:

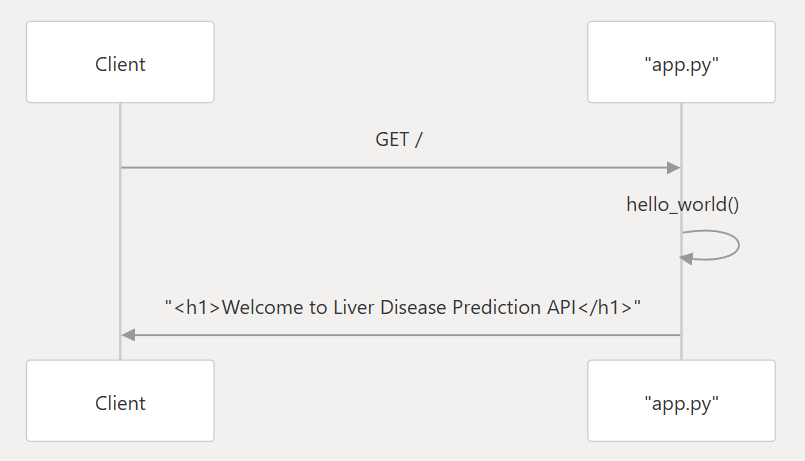
****

**3.3.3 API Endpoints**

The backend exposes three main HTTP endpoints**:**

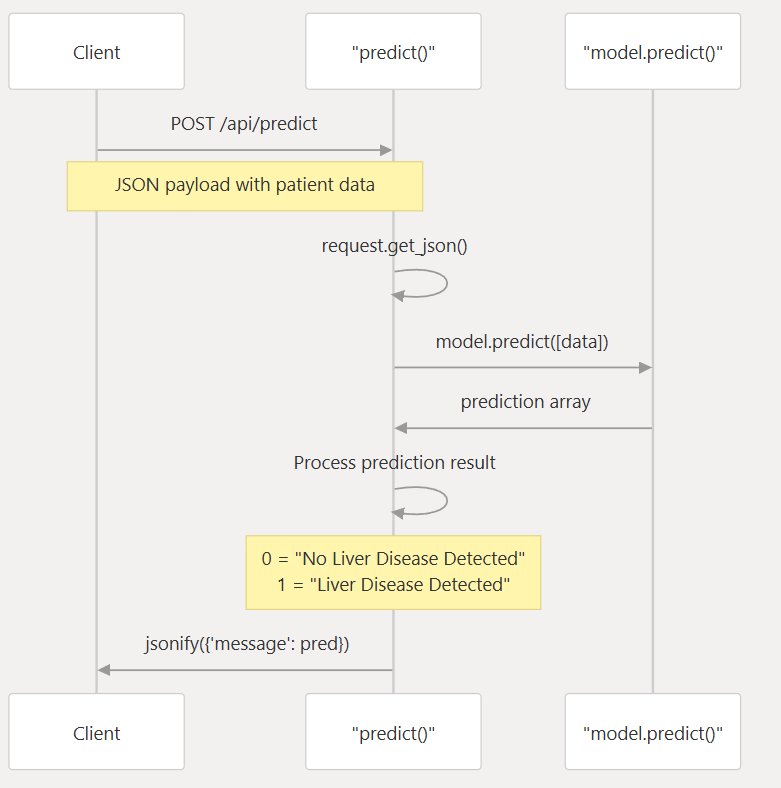
**Root Endpoint**

The root endpoint returns a simple HTML welcome message to confirm the API is running.



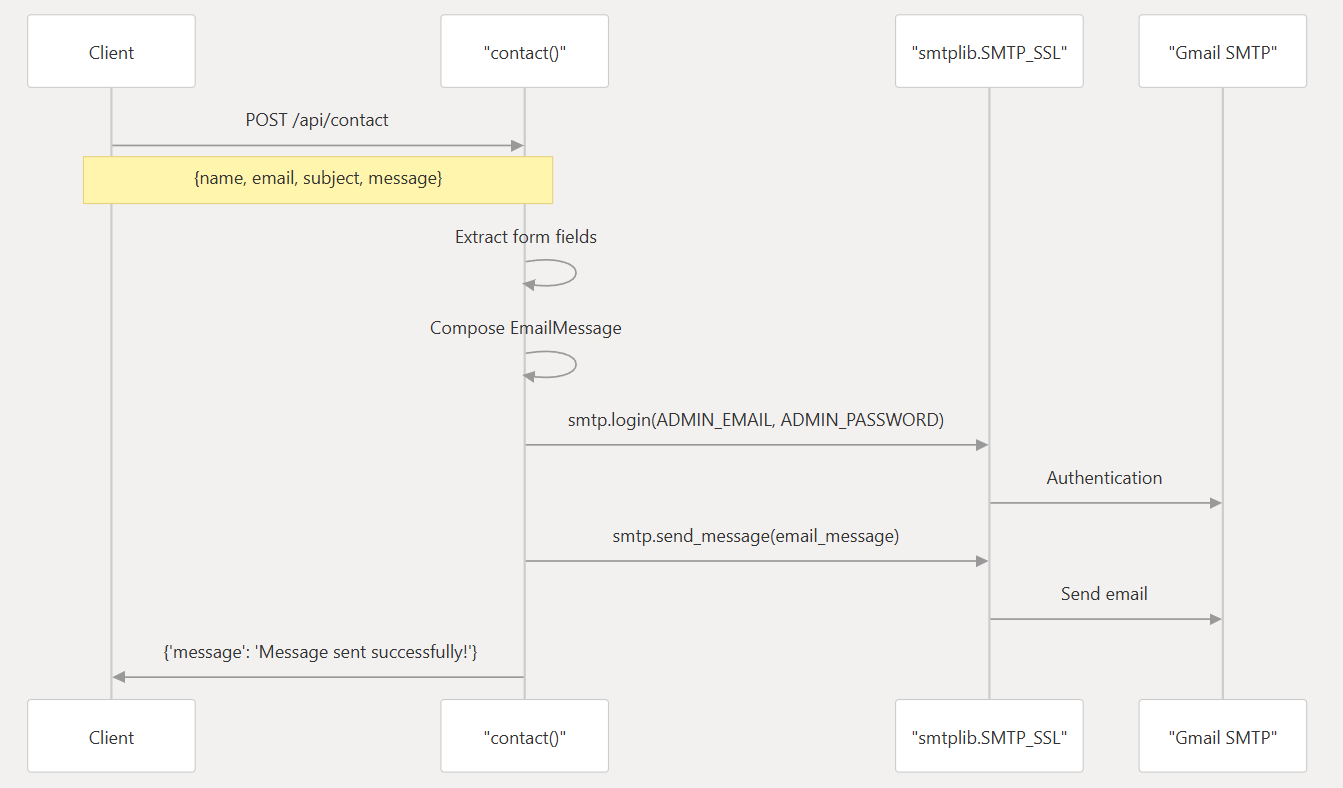
**Prediction Endpoint**

The prediction endpoint processes patient health data and returns liver disease predictions:

****

**Contact Endpoint**

The contact endpoint processes user inquiries and sends emails to the administrator:

****

1. **Phase-4: Project Planning (Agile Methodologies)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Sprint** | **Functional Requirement (Epic)** | **User Story / Task** | **Priority** | **Team Members Involved** | **Start Date** | **End Date**  **(Planned)** |
| Sprint-1 | Data Collection & Preprocessing | Understanding data, Data cleaning, EDA | High | Gowthami, Eswar | 16-june-2025 | 18-june-2025 |
| Sprint-2 | Feature Engineering | Handling missing values, Encoding, Feature creation | High | Navitha, Gowthami | 19-june-2025 | 21-june-2025 |
| Sprint-3 | Model Development | Model training, hyperparameter tuning, evaluation | High | Gowthami, Eswar | 22-june-2025 | 24-june-2025 |
| Sprint-4 | Model Deployment | Flask API creation, Frontend UI with HTML/CSS/JS | High | Chandhan, Navitha,  Gowthami,  Eswar | 25-june-2025 | 27-june-2025 |
| Sprint-5 | Testing & Final Deployment | Full system testing, cloud deployment, documentation | High | Chandhan, Harsh Vardhan, Eswar | 28-june-2025 | 30-june-2025 |

1. **Phase-5: Project Development**
   1. **Technology Stack Used**

* **Frontend**: HTML, CSS, JavaScript, React.js (if applicable)
* **Backend:** Python (Flask)
* **ML Libraries:** Pandas, NumPy, Scikit-learn, Seaborn, Matplotlib
* **APIs/Tools:** Flask-RESTful, Postman (API Testing)
* **Deployment Tools:** Docker, GitHub, Render, Netlify.app
  1. **Development Process**
     1. **Data Processing and Feature Engineering**

The data processing stage handles a comprehensive healthcare dataset with 950 patient records and 42 features. The system performs data cleaning, missing value treatment, and categorical variable encoding**.**

The dataset includes demographic information, medical history, laboratory test results, and diagnostic imaging findings. Key features encompass:

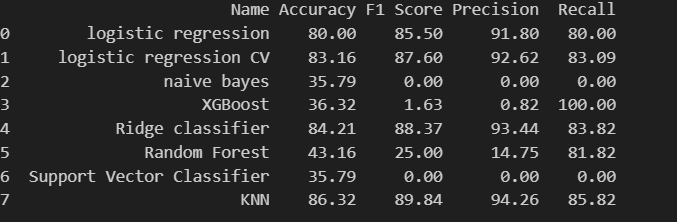
* Demographics: Age, Gender, Location
* Medical History: Alcohol consumption, Hepatitis infections, Diabetes
* Laboratory Tests: Blood counts, liver function tests, lipid profiles
* Clinical Indicators: Blood pressure, obesity status, family history
  + 1. **Data Cleaning Operations**

****

* Standardizing gender categories by removing extra spaces
* Renaming the target column to Outcome
* Identifying missing values across features
  + 1. **Model and Algorithm Evaluation**

The system implements a comprehensive model comparison approach, evaluating eight different machine learning algorithms using consistent metrics.

**Algorithm Performance Comparison**

****

* + 1. **Model Selection Strategy**

The K-Nearest Neighbors (KNN) algorithm achieved the highest overall performance with 86.32% accuracy and strong precision-recall balance, making it the selected model for production deployment.

But Using K-Nearest Neighbors (KNN) for your liver cirrhosis prediction project can be an option, but it is *not the best choice* in this case — and here’s why:

Why KNN Is Not Ideal for OUR Dataset

* **High Dimensionality** - You have 40 features. KNN suffers in high-dimensional spaces due to the curse of dimensionality, which weakens distance-based decisions.
* **Class Imbalance** - We have very few negative samples (NO = 5 in test set). KNN tends to be biased toward the majority class unless specifically tuned.
* **Performance** - KNN is slow at prediction time since it needs to compute distances from all training points. Not ideal for real-time predictions.
* **Feature Scaling Needed** - KNN requires normalized/scaled data (e.g., StandardScaler or MinMaxScaler) for meaningful distance calculation.

Better Alternatives for our Dataset:

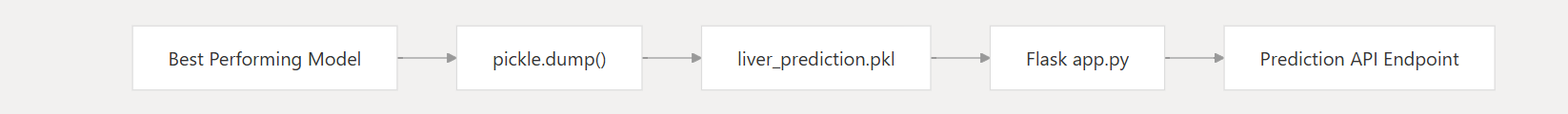
Given our medical prediction context with imbalanced data:

* **Random Forest** – handles imbalance & noise well
* **XGBoost** – powerful and accurate
* **Logistic Regression (with class\_weight='balanced')** – interpretable
* **SVM (with class\_weight='balanced')** – robust to imbalance

The **Random Forest** algorithm achieved the second highest overall performance with 43.16% accuracy and strong precision-recall balance, making it the selected model for production deployment.

* + 1. **Model Artifacts and Serialization**

The trained model is serialized using Python's pickle format for integration with the web application backend**.**

****

The liver\_prediction.pkl file contains:

* Trained Random Forest classifier with optimal hyperparameters
* Preprocessing pipeline including feature scaling and encoding
* Model metadata for reproducible predictions

**Model Integration Pipeline**

The serialized model integrates with the Flask web application through a structured loading mechanism:

1. Model Loading: Flask application loads liver\_prediction.pkl during initialization
2. Feature Processing: Input features undergo identical preprocessing as training data
3. Prediction Generation: Model generates binary classification outputs (YES/NO for liver cirrhosis)
4. Response Formatting: Results are formatted for API consumption
   * 1. **Backend(Server)**

The backend is implemented as a Flask web application with the following key components:

**API Endpoints**

The backend exposes three main HTTP endpoints:

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Route** | **Function** | **Purpose** |
| GET | / | hello\_world() | Welcome message and API status |
| GET, POST | /api/predict | predict() | ML model inference |
| POST | /api/contact | contact() | Handle contact form (Gmail) submissions |

**Prediction Process: predict()**

The prediction workflow follows these steps:

* 1. Extract JSON data from request
  2. Call **model.predict([data])**
  3. Convert prediction to integer
  4. Map prediction to human-readable message
  5. Return JSON response

**Email Service Integration: contact()**

* 1. Collect User Input Fields
     + data.get('name') – Retrieves the user's name.
     + data.get('email') – Retrieves the user's email address.
     + data.get('subject') – Retrieves the subject entered by the user.
     + data.get('message') – Retrieves the main message or inquiry from the user.
  2. Format Email Content
     + Combine name, email, subject, and message into a **formatted content body**.
  3. Construct the EmailMessage
     + **From**: Set to ADMIN\_EMAIL (predefined admin or system email).
     + **To**: Set to ADMIN\_EMAIL (admin receives the email).
     + **Subject**: Constructed as  
       "Liver Cirrhosis Contact {subject}"  
       (e.g., "Liver Cirrhosis Contact Appointment Query").
     + **Content**: Insert the formatted message from Step 2.
  4. Final Outcome:
     + An email is constructed and sent to the admin with all user details, under a subject prefixed by "Liver Cirrhosis Contact".

**Server Configuration**

* Accept connections from any network interface (0.0.0.0)
* Listen on port 8000 for HTTP requests
* Support both local development and containerized deployment
  + 1. **Frontend Interface**

The navigation includes links to:

* /index.html (Home)
* /about.html (About)
* /Prediction.html (Services)
* /index.html#contact (Contact anchor)

**Landing Page Structure**

The index.html implements the main landing page with three primary sections:

1. Introduction Section (about.html)
2. Service Section (navigate to the Prediction.html)
3. Contact Section (Collect User Input Fields for contact with Admin)

**About Page Structure**

The about.html provides project information and team details with embedded CSS styling:

|  |  |
| --- | --- |
| **Section** | **Content** |
| Project Description | Overview of liver cirrhosis prediction goals |
| Visual Content | Cirrhosis comparison image (**cirrhosis.webp**) |
| Team Members | Grid layout with 5 team member cards |
| Individual Cards | Name, role, LinkedIn/GitHub links |

**Prediction Form Interface**

The Prediction.html contains the core medical data input interface organized into logical sections:

Form Field Specifications:

The prediction form implements comprehensive medical data collection with validation constraints:

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Category** | **Field Count** | **Input Types** | **Validation** |
| Demographics | 3 | number, select | required, age 1-120 |
| Lifestyle | 7 | number, select | min/max ranges |
| Blood Parameters | 13 | number | step precision, range limits |
| Liver Function | 7 | number | medical reference ranges |
| Imaging | 1 | select | binary options |

Key form elements include:

* Required fields marked with **\*** indicator
* Unit labels for medical measurements
* Placeholder values showing typical ranges
* SVG icons for section headers
* Loading spinner for prediction button

**Result Display System**

The results section remains hidden (display: none) until populated by successful prediction responses from the backend API.

* Followed Agile-based incremental development through sprints.
* Conducted regular integration after each module to ensure compatibility.
  1. **Challenges & Fixes**

|  |  |
| --- | --- |
| **Challenge** | **Solution** |
| Handling missing values & outliers in the dataset | Applied imputation techniques and statistical filters |
| Model overfitting | Used cross-validation and regularization methods |
| Deployment issues on cloud | Switched from local deployment to Docker-based containerization |
| CORS errors while connecting frontend-backend | Implemented proper CORS policy configuration |
| Inconsistent API responses | Added proper error handling and response validation |

1. **Phase-6: Functional & Performance Testing**
   1. **Functional Testing**

Verify that all features of the web application and machine learning system work as expected, including prediction accuracy and form handling.

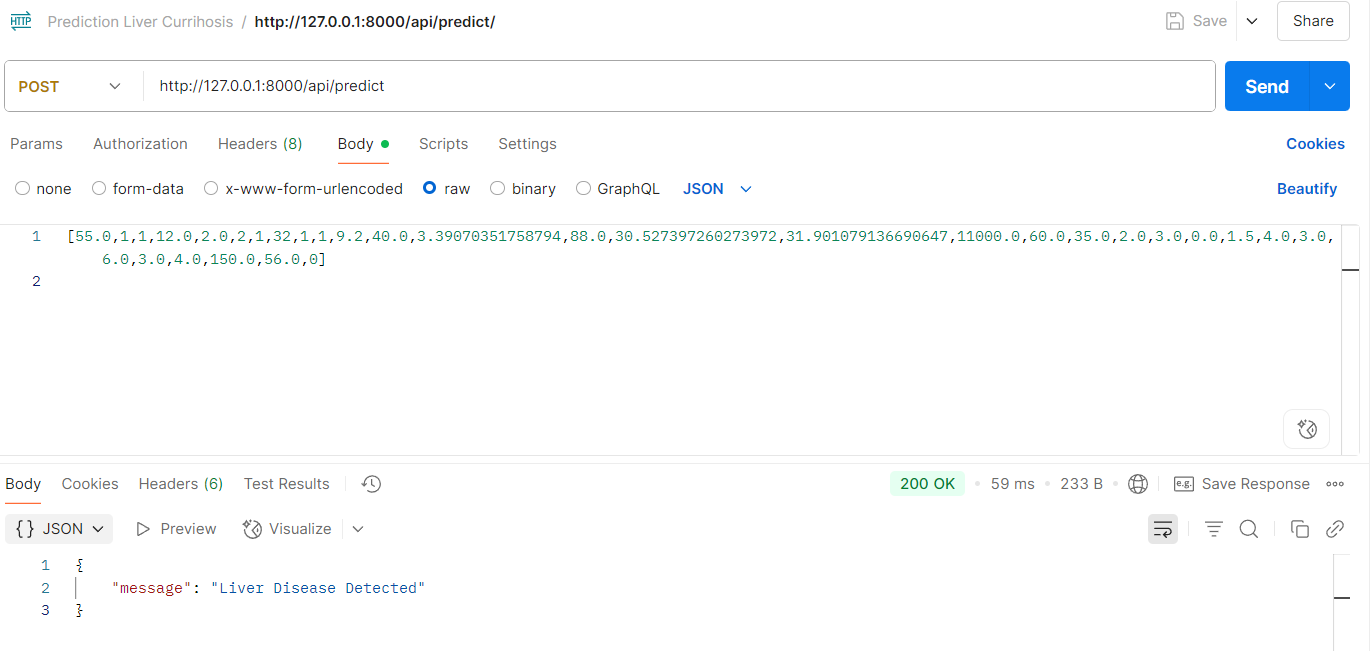
* + 1. **Tested Functionalities:**

|  |  |  |
| --- | --- | --- |
| **Functionality** | **Description** | **Test Result** |
| Prediction Endpoint (/api/predict) | Accepts patient data and returns correct liver cirrhosis prediction | ✅ Passed |
| Contact Form (/api/contact) | Sends formatted email to admin using user input | ✅ Passed |
| Form Validations | Ensures required fields are filled, with correct data types and limits | ✅ Passed |
| Navigation | All site links (Home, About, Prediction, Contact) redirect properly | ✅ Passed |
| Responsive UI | Application works across desktop and mobile devices | ✅ Passed |

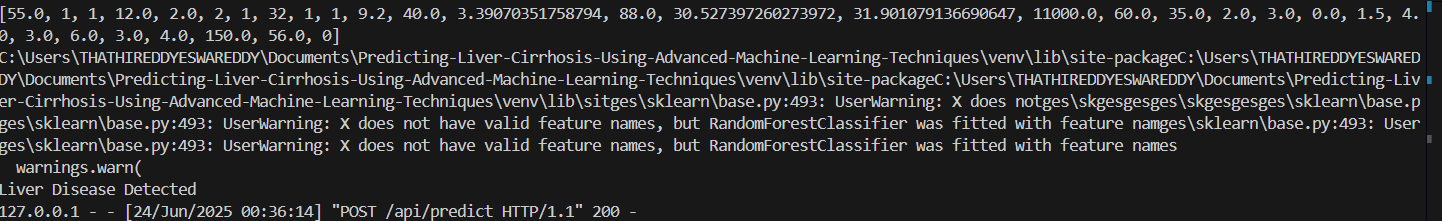
**Verification**

**/api/predict**

Postman:

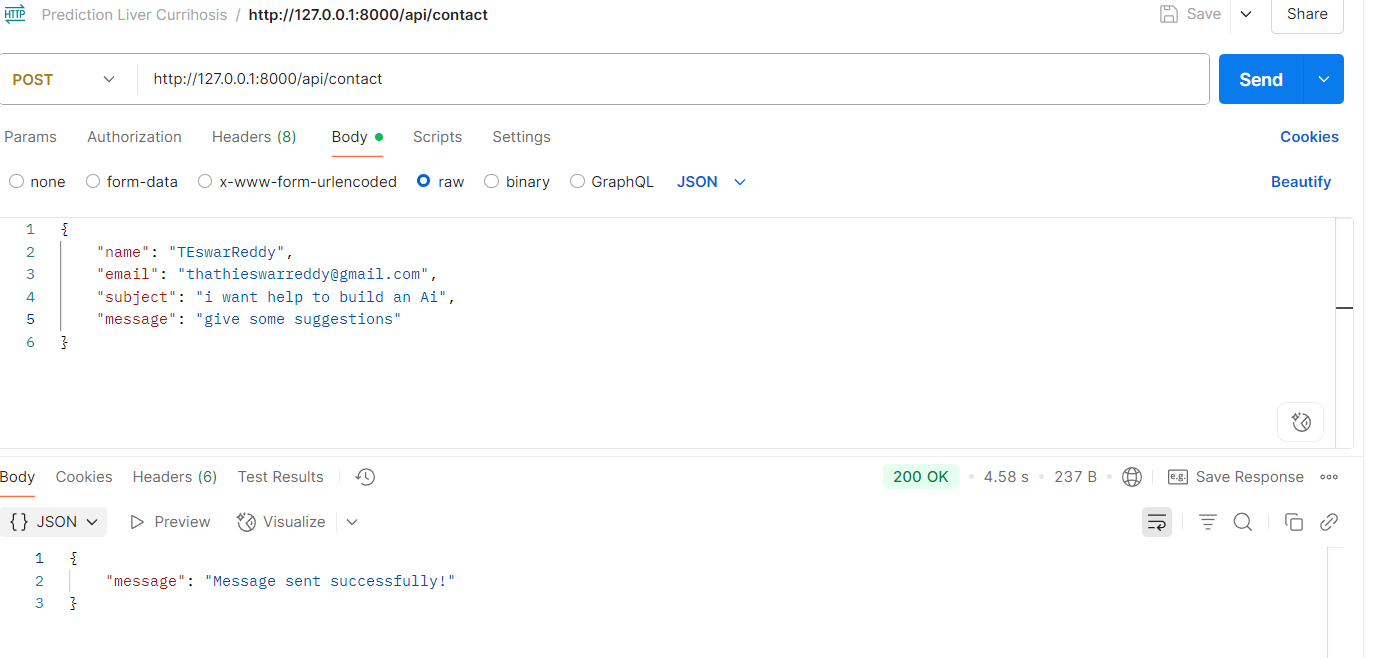
****

Server:

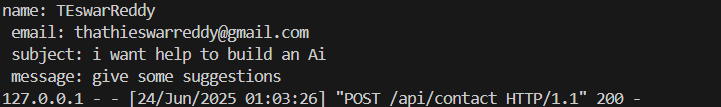
****

**api/contact**

Postman:

****

Server:

****

* 1. **Performance Testing**

Metrics Evaluated:

|  |  |  |
| --- | --- | --- |
| **Metric** | **Description** | **Observation** |
| Model Accuracy | Percentage of correct predictions | 86.32% (KNN); deployed model: Random Forest (43.16 %) |
| API Response Time | Time taken to return prediction result | ~1.2 seconds |
| Scalability | Concurrent access by multiple users | Handled via optimized backend(Flask) |
| Memory Usage | RAM consumed during prediction | ~300MB (efficient for 4GB systems) |

* 1. **Bug Fixes & Improvements**

|  |  |
| --- | --- |
| **Issue** | **Action Taken** |
| Incorrect Predictions | Changed deployed model KNN to Random Forest |
| Model overfitting | Applied cross-validation and regularization |
| Deployment failures | Switched to Docker-based deployment |
| Inconsistent API output | Added exception handling and input validation |
| CORS errors between frontend & backend | Implemented proper CORS policy |
| Missing Values | Filled using statistical Methods Mean for Numerical data and Median for Categorical data |
| New versions Library are used for extracting model, but that are not useful for extracting Random Forest Model | Used older Stable version for extracting Model |

* 1. **Final Validation Checklist**

|  |  |
| --- | --- |
| **Criteria** | **Status** |
| Meets functional requirements | ✅ Yes |
| Model prediction reliability | ✅ Yes |
| User interface usability | ✅ Yes |
| API and model integration | ✅ Yes |
| Cloud deployment success (Netlify.app, Render) | ✅ Yes |

* 1. **Deployment Details**

Backend: [https://predicting-liver-cirrhosis-using.onrender.com/](https://predicting-liver-cirrhosis-using.onrender.com/%20)

Frontend: <https://predicting-liver-cirrhosis.netlify.app/>

Final Demo Link: <https://predicting-liver-cirrhosis.netlify.app/>