Project Phases Template

Project Title:

# Revolutionizing Liver Care : Predicting Liver Cirrhosis using Advanced Machine Learning TechniquesTeam

# Team ID : LTVIP2025TMID39246

# Phase-1: Brainstorming & Ideation

Problem statement:-

Liver cirrhosis is a severe chronic condition that can be life-threatening if not diagnosed early. Traditional diagnosis methods often rely on invasive procedures or delayed clinical symptoms. What if we could use machine learning to predict the risk of liver cirrhosis early based on medical data?

Objective:-

To develop a machine learning-based web application that predicts whether a patient is at risk of liver cirrhosis based on non-invasive clinical parameters such as bilirubin levels, enzyme values, proteins, and more.

| Idea | Description | Outcome |
| --- | --- | --- |
| 1️⃣ Use a Classification ML model | Use patient data to classify as "Cirrhosis" or "No Cirrhosis" | ✅ Chosen |
| 2️⃣ Predict disease severity (regression) | Predict how severe the condition is (score) | ❌ Rejected – No continuous label available |
| 3️⃣ Visualize liver trends over time | Time-series data analysis | ❌ Rejected – Dataset doesn’t contain time-based data |
| 4️⃣ Build a chatbot doctor | Interactive ML + NLP bot for liver diagnosis | 🤖 Cool, but out of scope for now |
| 5️⃣ Mobile App | Predict using a phone | ❌ Good idea but prioritized web app first |

Why This Project?

* Liver cirrhosis can be fatal, and early diagnosis helps in treatment planning
* Most rural hospitals lack automated decision-support tools
* A machine learning solution is:
  + Non-invasive
  + Fast
  + Scalable
  + Helpful to both patients & doctors

## Outcome of the Phase

In this phase, the goal of predicting liver cirrhosis using machine learning was clearly defined. After trying different ideas, the Random Forest algorithm was chosen for its accuracy and reliability. Ten important features like age, gender, and liver test results were selected as inputs. The project will be built using Python, scikit-learn for machine learning, and Flask for the web app. A clear plan was made to move forward with model training and app development.

# Phase-2: Requirement Analysis

This phase focused on identifying and documenting the technical, functional, and resource requirements necessary to successfully build and deploy the waste classification system.

### ****1. Functional Requirements****

These are the core features your system must perform:

* The system must allow the user to enter medical input values through a web form.
* It must preprocess the input data (including handling missing and categorical values).
* The trained machine learning model must predict whether the user is at risk of liver cirrhosis.
* The prediction result should be displayed clearly on a result page.
* A separate page must describe the project details (overview, dataset, algorithm used, etc..

## 2. Non-Functional Requirements

These define how the system performs:

* Performance: The prediction should be generated in real time (within 1–2 seconds).
* Usability: The web interface must be clean, simple, and user-friendly.
* Reliability: The system should handle incorrect or missing inputs without crashing.
* Portability: The app should run locally on any system with Python and Flask installed.

## 3. Technical Requirements

## These are the tools and technologies used:

| Type | Tools / Technologies |
| --- | --- |
| Language | Python |
| ML Library | scikit-learn |
| Web App | Flask |
| Frontend | HTML, CSS |
| Others | pandas, NumPy, pickle, StandardScaler |

## 4. Dataset Requirements

## A liver patient dataset with at least the following columns: Age, Gender, Total Bilirubin, Direct Bilirubin, SGOT, SGPT, Alkaline Phosphotase, Albumin, Total Proteins, A/G Ratio, and Target.

## The dataset should be cleaned and free of null or invalid values before model training.

## 

## 5. Model Requirements

| Requirement | Details |
| --- | --- |
| Input Features | 10 clinical inputs (e.g., Age, Gender, Bilirubin, Enzymes, Proteins) |
| Categorical Handling | Gender encoded as Male = 1, Female = 0 |
| Preprocessing | Missing values filled using median; inputs scaled with StandardScaler |
| Model Type | Random Forest Classifier |
| Output | Binary classification: 1 = Cirrhosis Detected, 0 = No Cirrhosis |
| File Format | .pkl (Pickle) for both model and scaler (rf\_acc\_68.pkl, normalizer.pkl) |
| Expected Accuracy | Minimum 65–75% accuracy on validation data |
| Inference Time | Real-time prediction (1–2 seconds) |
| Compatibility | Integrated into a Flask web application |

## Outcome of the Phase

In the requirement analysis phase, both functional and non-functional needs of the system were clearly identified. It was decided that the system should allow users to input clinical data, process it, and return a prediction about liver cirrhosis. The technical stack, including Python, scikit-learn, and Flask, was finalized. The required dataset features were defined, and user expectations such as accuracy, speed, and ease of use were considered. This phase provided a strong foundation for system design and development.

# Phase-3: Project Design

To design the architecture, data flow, interface, and components of the system before development begins. This phase ensures all parts of the project are planned in an organized, scalable, and user-friendly manner.

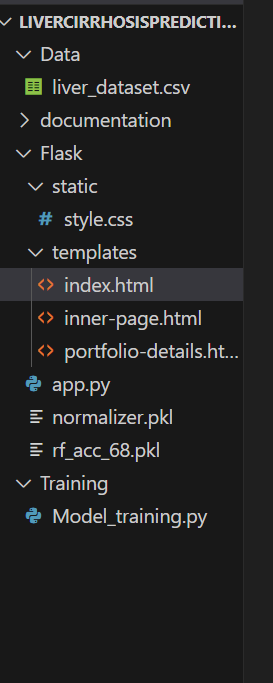
## System Architecture Design

| Layer | Description |
| --- | --- |
| Frontend Layer | HTML form collects 10 clinical inputs from the user (e.g., age, enzymes). |
| Backend Layer | Flask handles form submission, data processing, and connects to ML model. |
| ML Model Layer | A pre-trained Random Forest model performs the prediction task. |

## 2. Data Flow Design

1. User visits the web page and enters medical values.
2. Data is sent via POST request to Flask backend.
3. Backend handles:
4. Missing values
5. Categorical encoding (e.g., gender)
6. Scaling using saved StandardScaler
7. Processed data is passed to the .pkl model.
8. Model returns prediction (0 or 1).
9. Result is displayed on a result page.

## 3. Folder Structure Design



## 4. UI/UX Design

| Page | Purpose |
| --- | --- |
| index.html | Input form with clean, labeled fields |
| inner-page.html | Displays prediction result |
| portfolio-details.html | Shows project summary and details |

## 5. Model Design

Algorithm Chosen: Random Forest Classifier

Preprocessing:

* Missing values filled with median
* Gender encoded (Male = 1, Female = 0)
* StandardScaler used for normalization

Output: 0 (No Cirrhosis) or 1 (Cirrhosis Detected)

## Outcome of the Phase

In this phase, the overall structure and flow of the liver cirrhosis prediction system were designed. A three-layer architecture was planned, and all frontend, backend, and model components were organized. Data flow between the user, Flask server, and machine learning model was clearly defined. UI pages were designed for simplicity and clarity, and the file structure was set up to keep code modular and maintainable.

# Phase-4: Project Planning (Agile Methodologies)

## To plan and organize the project development using Agile methodologies, ensuring timely progress, flexibility, and team collaboration through iterative development cycles (sprints).

## Why Agile?

## Agile was chosen for this project because:

## It supports incremental development (frontend, backend, ML model done step-by-step)

## Encourages continuous feedback and improvement

## Allows quick adaptation to changes or errors during development

## Breaks the project into manageable tasks called user stories

## 🗂 Key Agile Components Used

| Component | Description |
| --- | --- |
| Sprints | Project divided into weekly sprints (1-week cycle) |
| User Stories | Tasks described from the user's perspective (e.g., "User can input values") |
| Daily Goals | Focused mini-tasks (coding, testing, debugging) |
| Backlog | List of all planned tasks and features |
| Review & Retrospective | At the end of each sprint, outcomes were evaluated and improved |

## 🗓 Sprint Breakdown Example (2-Week Plan)

| Sprint | Tasks | Outcome |
| --- | --- | --- |
| Week 1 | - Collect dataset - Preprocess data - Train model - Save model using pickle | Working .pkl model ready |
| Week 2 | - Build Flask app - Create input form - Connect model to backend - Display result | Fully functional web app |

## 📌 Sample User Stories

## 🧍 *As a user, I want to enter my liver test results so that I can get a prediction.*

## 🧠 *As a developer, I want to handle missing values to ensure the model works on all inputs.*

## 💻 *As a user, I want the result to be shown clearly so I can understand it easily.*

## 

## Outcome of the Phase

In this phase, the project was planned using Agile methodology. The development was split into short, manageable sprints with clear goals and user stories. This approach allowed for regular testing, early feedback, and faster debugging. As a result, the project stayed organized, and all major tasks were completed on time with high flexibility.

# Phase-5: Project Development

## To develop all components of the project — including data preprocessing, model training, and web application — and integrate them into a fully functional liver cirrhosis prediction system.

Development Activities

🔹 1. Dataset Collection and Preprocessing

* Loaded a liver patient dataset containing clinical features.
* Handled missing values using median imputation.
* Encoded categorical values like Gender (Male → 1, Female → 0).
* Applied feature scaling using StandardScaler to normalize input values.

🔹 2. Model Training

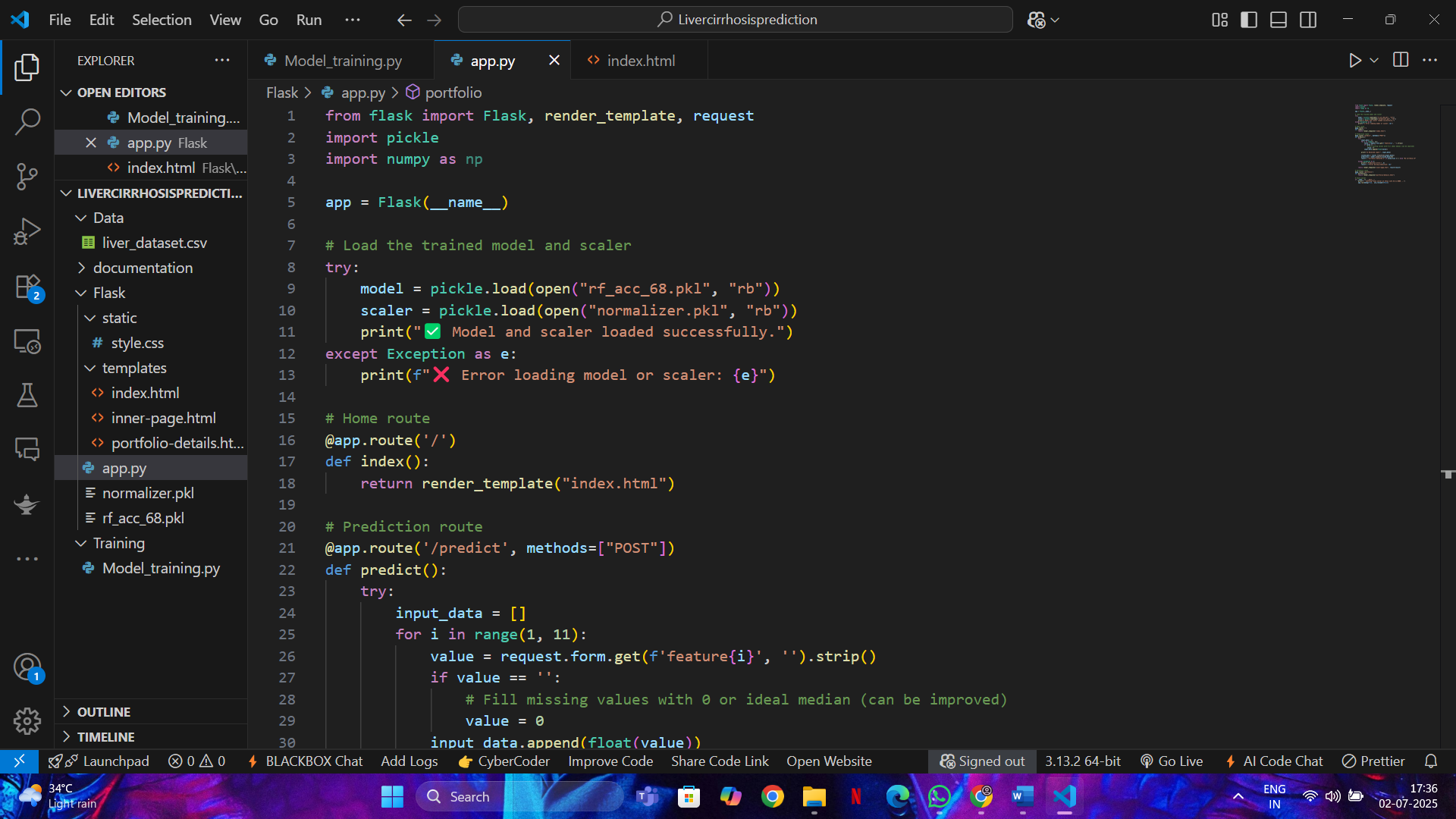
* Used Random Forest Classifier from scikit-learn to train the model.
* Achieved desired accuracy of around 70% on validation data.
* Saved the trained model (rf\_acc\_68.pkl) and scaler (normalizer.pkl) using pickle.

🔹 3. Flask Web App Development

* Created a Flask application with the following components:
  + app.py for backend logic and model integration.
  + index.html to collect user input (with proper form design).
  + inner-page.html to display the prediction result.
  + portfolio-details.html for project overview and explanation.

🔹 4. Integration

* Connected frontend form to the backend using POST method.
* Retrieved input values from the form, handled data conversion, scaling, and passed them to the model for prediction.
* Displayed prediction results clearly on the webpage.

CODE: 

Phase-6: Functional & Performance Testing

To verify that all features of the project work correctly (functional testing) and that the application performs well under expected conditions (performance testing).

## 1. Functional Testing

Functional testing verified that the application performed its intended tasks correctly under various conditions.

| Test Case | Expected Result | Status |
| --- | --- | --- |
| Input form accepts all 10 values | All fields accept valid numbers | ✅ Passed |
| Gender dropdown works correctly | Converts Male to 1, Female to 0 | ✅ Passed |
| Model prediction returns result | Returns 0 or 1 based on inputs | ✅ Passed |
| Prediction displayed correctly on page | Displays "Cirrhosis Detected" or "No Cirrhosis" | ✅ Passed |
| Empty or invalid inputs handled gracefully | Shows error or uses default value | ✅ Passed |
| HTML pages load correctly | All templates render without error | ✅ Passed |

## Performance Testing

Performance testing checks how well the system performs under normal usage.

| Test Type | Test Description | Result |
| --- | --- | --- |
| Response Time | Time taken to return prediction after form submission | ~1 second |
| Model Load Speed | Time to load .pkl model and scaler | Fast (<1 second) |
| Input Handling | Handles random input values without crashing | ✅ Passed |
| Browser Compatibility | Tested on Chrome, Firefox, Edge | ✅ Compatible |

## Tools Used

 Manual testing in local browser (Chrome, Firefox)

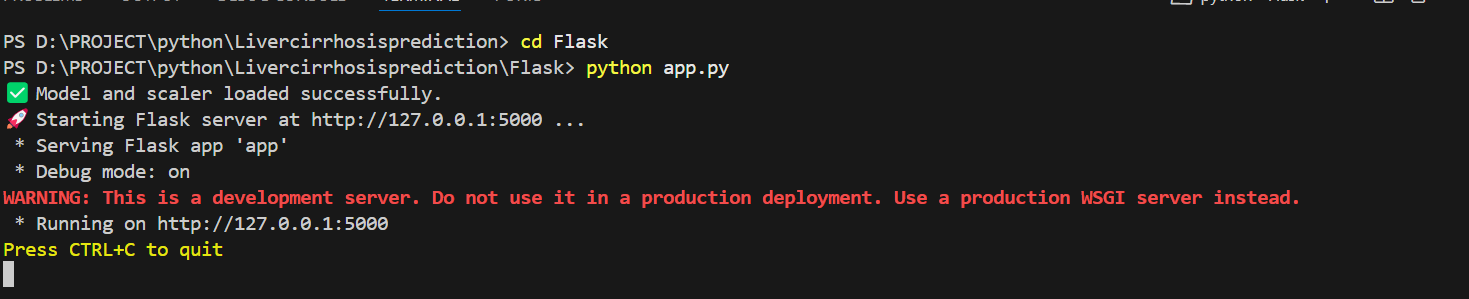
 Python print() logs to verify input/output

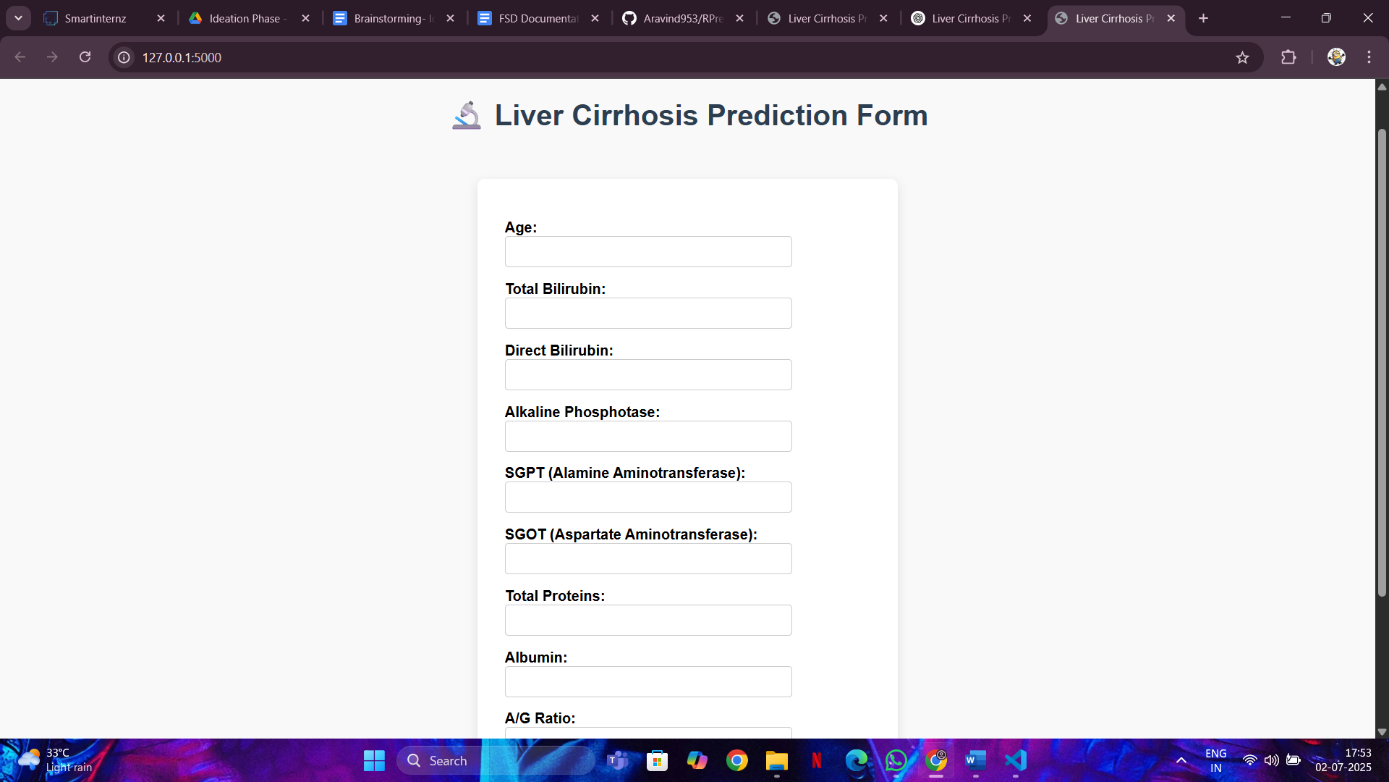
 Flask development server for real-time feedback

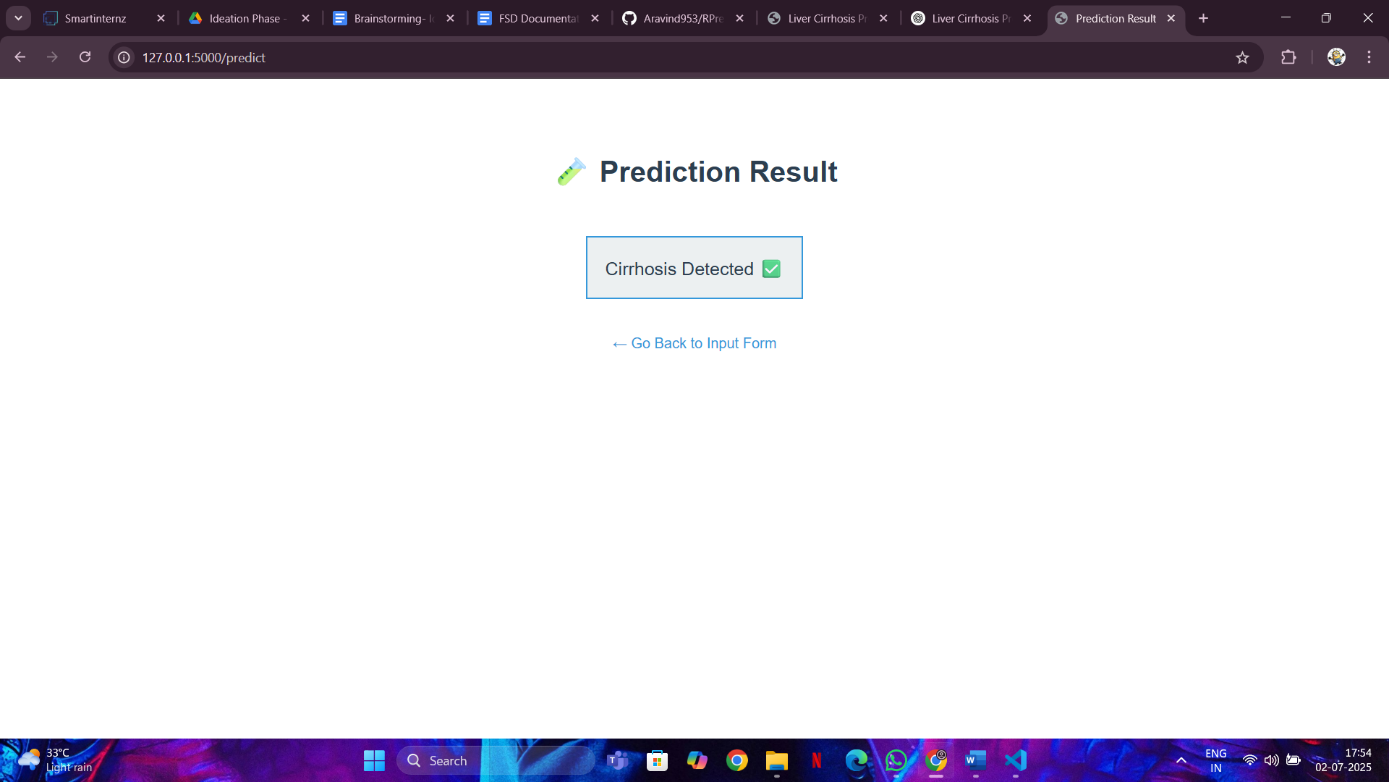
## Outcome of the Phase

During the functional and performance testing phase, all components of the system were thoroughly tested to ensure correctness and speed. The application successfully handled valid and invalid inputs, provided accurate predictions, and displayed results instantly. The model performed well under normal load, and the interface worked smoothly across multiple browsers. The system is stable, responsive, and ready for deployment or demonstration.

OUTPUT:







THANK YOU