**3. System Architecture & Deployment Flow – Liver Cirrhosis Prediction**

**🔹 Overview**

The system is designed as a **machine learning-based web application** for predicting liver cirrhosis. It integrates a trained model into a Flask web framework and allows users (healthcare professionals or patients) to input data and receive instant predictions.

**🔹 End-to-End Workflow**

text

CopyEdit

+---------+ +----------+ +-------------+ +-------------+ +--------+

| User | ---> | Web UI | ---> | Flask Server| ---> | ML Model | ---> | Output |

+---------+ +----------+ +-------------+ +-------------+ +--------+

**Flow Description:**

1. User visits the web page and enters medical test results.
2. The form submits the data to the backend Flask application.
3. Flask receives and normalizes the input using normalizer.pkl.
4. Normalized data is passed to the trained model (rf\_acc\_68.pkl) for prediction.
5. The result is displayed on the webpage (cirrhosis or not).

**🔹 File Structure and Roles**

| **Folder/File Name** | **Purpose** |
| --- | --- |
| app.py | Main Python Flask file. Loads model and handles prediction logic. |
| templates/index.html | HTML form for collecting user input. Frontend interface. |
| static/ | CSS, JavaScript, and image files (for styling and visuals). |
| rf\_acc\_68.pkl | Serialized Random Forest model trained on liver dataset. |
| normalizer.pkl | Pre-fitted Standard scaler object used for feature normalization. |
| requirements.txt | Contains list of required Python packages (Flask, scikit-learn, etc.). |
| training\_notebook.ipynb | Jupiter notebook for data analysis, visualization, and model training. |

**🔹 Backend Logic (app.py) Highlights**

* Loads the model and scaler at startup using pickle.
* Defines Flask route / for the home page (index.html).
* Accepts form data via POST, processes it, and makes predictions.
* Displays result using Jinja templating in HTML.

**🔹 Technologies Used**

| **Technology** | **Role** |
| --- | --- |
| Python | Core language for model, preprocessing, and app logic |
| Flask | Lightweight web framework for deployment |
| Pandas/NumPy | Data manipulation and preparation |
| scikit-learn | Model training, evaluation, and export |
| HTML/CSS | Web interface for user interaction |
| Jupiter Notebook | For EDA, feature engineering, and model training |

**🔹 Deployment Instructions**

**To Run Locally:**

1. Ensure Python is installed
2. Install dependencies: pip install -r requirements.txt
3. Run the Flask app: python app.py
4. Open http://127.0.0.1:5000/ in browser

**For Online Hosting (optional):**

* Push code to GitHub
* Use Render, Heroku, or Streamlit to host
* Add runtime, Procfile, and environment settings as needed

**🔹 Conclusion**

This modular architecture ensures:

* Easy maintainability and updates
* Secure handling of inputs
* Quick and interactive user experience
* Readiness for cloud deployment