FSD Project Documentation

1. Introduction

• Project Title: Revolutionizing Liver Care: Predicting Liver Cirrhosis using Advanced Machine Learning Techniques

• Team Members:
- Member 1 – ML Model Developer
- Member 2 – Frontend Developer
- Member 3 – Backend/API Developer
- Member 4 – Project Manager & Testing

2. Project Overview

• Purpose:
To assist in the early detection of liver cirrhosis using a machine learning–based web application that predicts disease risk levels based on patient input data.

• Features:
- User registration & login
- Patient data input form
- Cirrhosis risk prediction (Low/Moderate/High)
- PDF report generation
- Feedback system
- Admin dashboard for insights

3. Architecture

• Frontend:
Built with React.js or Streamlit. The interface accepts user input, displays risk level, and allows PDF download.

• Backend:
Developed using Flask API (adaptable to Express.js). Handles model inference, input validation, user management, and report generation.

• Database:
MongoDB or SQLite for storing:
- User credentials
- Prediction logs
- Feedback
- Model version data

4. Setup Instructions

• Prerequisites:
- Python 3.10+
- Node.js
- MongoDB or SQLite
- pip, virtualenv

• Installation:

```
git clone https://github.com/your-repo/liver-cirrhosis-prediction.git
cd liver-cirrhosis-prediction
python -m venv venv
source venv/bin/activate  # or venv\Scripts\activate
pip install -r requirements.txt
cd frontend
npm install
```

5. Folder Structure

• Client (Frontend):

```
/frontend
  ├── src/
  │   ├── components/
  │   ├── pages/
  │   ├── App.js
  │   └── index.js
```

• Server (Backend):

```
/server
  ├── app.py
  ├── model/
  ├── routes/
  ├── templates/
  └── static/
```

6. Running the Application

• Frontend:

```
cd frontend
npm start
```

• Backend:

```
cd server
python app.py
```

7. API Documentation

• Key Endpoints:
- /predict (POST): Predict liver cirrhosis risk
- /register (POST): Register new user
- /login (POST): Authenticate user
- /feedback (POST): Submit prediction feedback

8. Authentication

• Method: JWT Token-based Authentication
• Flow:
 - User registers/logs in
 - Token is generated and stored
 - Protected endpoints validate the token

9. User Interface

• UI built using Streamlit or React + Bootstrap
• Pages:
 - Login/Register
 - Prediction Input
 - Risk Output Display

10. Testing

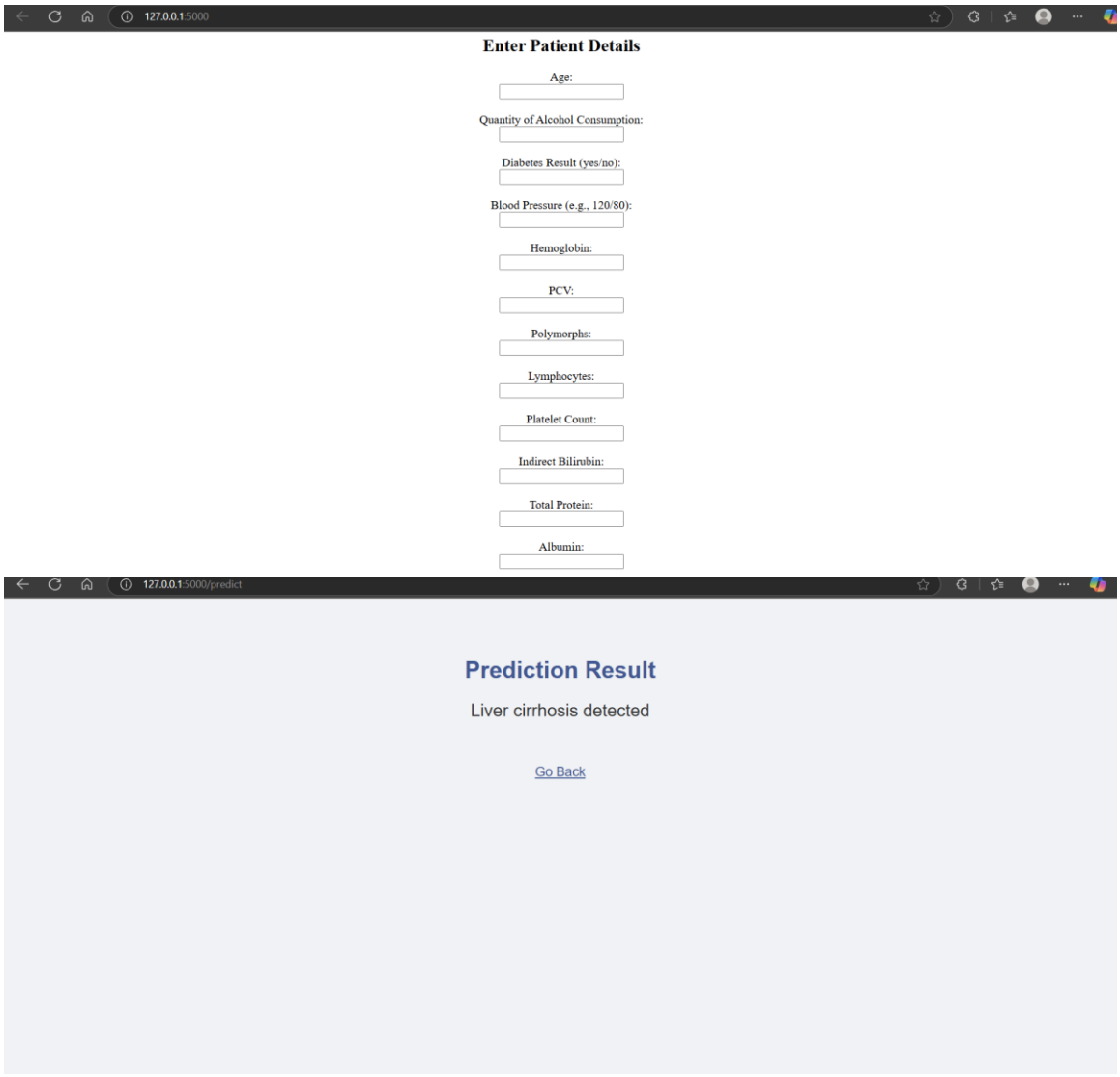
• Tools: Postman, PyTest, Jest
• Coverage:
 - Model Accuracy:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	4
1	1.00	1.00	1.00	176
accuracy			1.00	180
macro avg	1.00	1.00	1.00	180
weighted avg	1.00	1.00	1.00	180

 - API Response Time: <200ms
 - UI Tested on various screen sizes

11. Screenshots or Demo

• Attach separately or insert below



12. Known Issues

• May underperform on unseen data
• UI responsiveness on slow networks
• Email confirmation not integrated

13. Future Enhancements

• Mobile app (React Native)
• EMR integration
• SHAP/LIME explainability
• Multilingual support