Low-Level Design (LLD) – Crypto Liquidity Prediction

Name: Shreya Patra
Date: 18 July 2025

1. Introduction

This document explains the internal code structure of the *Cryptocurrency Liquidity Prediction* project. It provides a breakdown of all the scripts and functions, describing how they work together to process data and deliver predictions through a web app.

2. Code Module Breakdown

- **data_preprocessing.py**
  Handles data cleaning including missing value handling and normalization.
- **feature_engineering.py**
  Adds technical indicators such as moving averages, volatility, and liquidity ratio.
- **train_model.py**
  Splits data, trains the Random Forest model, and evaluates it.
- **predictor.py**
  Loads the trained model and generates predictions on new input.
- **app.py (Streamlit)**
  Builds a web interface where users enter features and receive real-time predictions.

3. Function Descriptions

Function	Purpose	Input	Output
clean_data(df)	Handle missing values	Raw DataFrame	Cleaned DataFrame
scale_features(df)	Normalize features	DataFrame	Scaled DataFrame
create_features(df)	Add features (MA, Volatility)	DataFrame	Enhanced DataFrame
train_random_forest(X, y)	Train model	X, y	Trained RF Model
make_prediction(input)	Predict from user input	User Data	Predicted price

4. Dependencies Used

- pandas, numpy – for data handling
- scikit-learn – for model training and evaluation

- matplotlib, seaborn – for visualization
- streamlit – for building the app
- joblib – for saving and loading model

5. Execution Flow

1. Load and clean dataset
2. Scale and transform numerical features
3. Engineer new features
4. Train Random Forest model
5. Save model for reuse
6. Deploy model in Streamlit app
7. User inputs values → model gives prediction