High-Level Design (HLD) Document

Project: Cryptocurrency Liquidity Prediction for Market Stability

Name: Shreya Patra

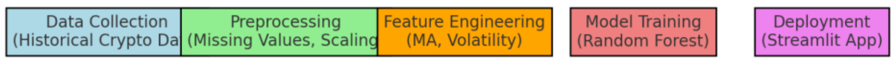
Date: 18 July 2025

1. Project Overview

The Cryptocurrency Liquidity Prediction project is built to analyze the dynamic behavior of the crypto market, with a specific focus on Bitcoin. The main objective is to forecast liquidity conditions using machine learning, helping traders and platforms anticipate instability. The high-level architecture follows a modular approach that smoothly transitions data from raw form to a working web-based prediction tool.

2. System Architecture Diagram

Below is the overall architecture diagram that outlines the end-to-end flow of the system:



3. Component Descriptions

- ****Data Collection:****

Historical data related to Bitcoin prices, trading volumes, and market cap was collected, primarily in CSV format.

- ****Preprocessing:****

This step involved treating missing or null values using strategies like forward-fill and mean replacement. Normalization was applied using MinMaxScaler to ensure data consistency.

🔹 **Feature Engineering:**
New indicators were created to capture trends, including 7-day moving averages, volatility (High - Low), and liquidity ratios like Volume/Market Cap.

🔹 **Model Training:**
A Random Forest Regressor was trained using historical data. The model was evaluated using RMSE, MAE, and R^2 to ensure accuracy in liquidity prediction.

🔹 **Deployment:**
Finally, a simple and interactive interface was built using Streamlit where users can enter market indicators and receive real-time predictions of Bitcoin price.

4. Technology Stack

The tools and libraries used in this project include:
- Python
- Pandas
- NumPy
- Scikit-learn
- Streamlit
- Matplotlib & Seaborn
- Jupyter Notebook / VS Code

5. Workflow Summary

Here is a step-by-step summary of how the system operates:
1. Load and clean the raw historical dataset.
2. Apply transformations and engineer new features.
3. Train the Random Forest model on historical patterns.
4. Validate and fine-tune the model based on performance.
5. Deploy the model using Streamlit where users interact via sliders.
6. Display the prediction output instantly on the app interface.