High-Level Design (HLD) Document

Project Title: Cryptocurrency Liquidity Prediction for Market Stability

1. Objective: To predict liquidity levels of cryptocurrencies using machine learning techniques based on historical market indicators, enabling better decision-making for exchanges and investors.

---

2. System Architecture Overview:

Data Flow:

```
[Raw CSV Data]
    ↓
[Data Cleaning & Preprocessing Layer]
    ↓
[Feature Engineering Layer]
    ↓
[Model Training & Evaluation Layer]
    ↓
[Saved Model (.pkl)]
    ↓
[Flask Deployment API + Web UI]
```

---

3. Major Components:

- **Data Source:** CoinGecko (CSV historical data for March 2022)
- **Preprocessing Layer:**
  - Handling missing values using forward/backward fill
  - Dropping unnecessary columns (e.g., `symbol`)
  - Normalizing with `MinMaxScaler`
- **Feature Engineering:**
  - Liquidity Ratio: `volume / market_cap`
  - Rolling stats: Moving averages, volatility
- **Model Layer:**
  - Algorithm: `RandomForestRegressor`
  - Evaluation: R^2, MAE, RMSE
- **Deployment Layer:**
  - Flask-based API + HTML UI
  - User input form to predict liquidity ratio

---

4. Deliverables:

- Trained ML model (.pkl)
- Flask web application
- Reports: EDA, HLD, LLD, Final Summary

==========

Low-Level Design (LLD) Document

Project Title: Cryptocurrency Liquidity Prediction for Market Stability

1. Data Preprocessing Module

- Input: Raw CSV files (`coin_gecko_2022-03-16.csv`, `coin_gecko_2022-03-17.csv`)
- Merge Datasets using `pd.concat()`
- Drop column: `symbol`
- Handle missing values: `.ffill()` + `.bfill()`
- Convert features to numeric: `pd.to_numeric()`
- Normalize columns: `MinMaxScaler` on `price`, `volume`, `market_cap`

2. Feature Engineering Module

- Create:
  - `liquidity_ratio` = volume / (market_cap + epsilon)
  - `price_change_pct` = price_change_24h / price
  - Rolling windows: `price_ma_3`, `volume_ma_3`, `price_volatility_3`
- Fill NaNs post-rolling: `.bfill()`

3. Model Training Module

- Split data: `train_test_split` with 80/20
- Initialize: `RandomForestRegressor(n_estimators=100)`
- Train model on `X_train`, `y_train`
- Save model with `joblib.dump()`

4. Evaluation Module

- Predict on test set
- Metrics: `r2_score`, `mean_absolute_error`, `mean_squared_error`

5. Flask App Module

- `app.py` contains routes:
  - `/` = home page (HTML form)
  - `/predict` = prediction endpoint
- User enters:
  - price, volume, market cap, price_change_24h, price_change_7d
- Model returns predicted liquidity ratio

6. UI (HTML)

- Simple form in `index.html`

- Displays predicted result after form submission
