High-Level Design (HLD) & Low-Level Design (LLD)

Cryptocurrency Volatility Prediction System

---

1. Introduction

This document describes the **High-Level Design (HLD)** and **Low-Level Design (LLD)** of the *Cryptocurrency Volatility Prediction System*. The system is designed to predict future volatility levels of cryptocurrencies using historical market data such as OHLC prices, trading volume, and market capitalization.

The design ensures scalability, modularity, and ease of deployment while maintaining accuracy and interpretability.

PART A: HIGH-LEVEL DESIGN (HLD)

2. System Overview

The system follows a **machine learning pipeline architecture** consisting of data ingestion, preprocessing, feature engineering, model training, evaluation, and deployment. It enables users to input market data and receive predicted volatility levels.

2.1 Objectives

- Predict short-term cryptocurrency volatility
- Assist in risk management and trading decisions
- Provide interpretable insights into market behavior

---

3. High-Level Architecture

3.1 System Components

1. **Data Source**
2. Historical cryptocurrency dataset (CSV format)

3. Contains OHLC, volume, and market capitalization data

4. **Data Preprocessing Module**

5. Handles missing values

6. Sorts and cleans time-series data
7. Normalizes numerical features

8. **Feature Engineering Module**

9. Generates volatility and liquidity-related features
10. Computes rolling statistics and technical indicators

11. **EDA & Analysis Module**

12. Statistical analysis
13. Visualization of trends and correlations

14. **Model Training Module**

15. Trains machine learning models (Random Forest)
16. Performs hyperparameter tuning

17. **Model Evaluation Module**

18. Evaluates model using RMSE, MAE, and R^2

19. **Deployment Module**

20. Local deployment using Streamlit
21. Accepts user input and returns predictions

---

4. High-Level Data Flow

```
Raw Dataset
     ↓
Data Preprocessing
     ↓
Feature Engineering
     ↓
EDA & Insights
     ↓
Model Training
     ↓
Model Evaluation
```

```
     ↓
Deployment (Streamlit App)
```

---

5. Technology Stack (HLD)

Layer	Technology
Programming Language	Python
Data Processing	Pandas, NumPy
Visualization	Matplotlib, Seaborn
Machine Learning	Scikit-learn
Model Persistence	Joblib
Deployment	Streamlit

---

6. Assumptions & Constraints

- Dataset is historical and static
- Internet access is not required for prediction
- System is designed for local deployment

PART B: LOW-LEVEL DESIGN (LLD)

7. Module-Wise Detailed Design

7.1 Data Ingestion Module

Input: CSV file containing cryptocurrency market data

Process: - Load dataset using Pandas - Parse date column - Sort records by symbol and date

Output: Cleaned raw DataFrame

7.2 Data Preprocessing Module

Responsibilities: - Handle missing values using forward fill - Remove invalid or incomplete rows - Ensure data consistency

Key Functions: - `fillna(method='ffill')` - `dropna()`

Output: Preprocessed DataFrame

7.3 Feature Engineering Module

Engineered Features:

Feature	Description
Log Return	Natural log of price change
Rolling Volatility (14 days)	Target variable
SMA (7, 14)	Trend indicators
Price Range	High − Low
Volume/Market Cap Ratio	Liquidity indicator

Logic: - Group data by cryptocurrency symbol - Apply rolling window calculations

Output: Feature-enhanced DataFrame

7.4 EDA Module

Responsibilities: - Generate statistical summaries - Plot volatility distribution - Analyze correlations

Tools Used: - Matplotlib - Seaborn

7.5 Model Training Module

Algorithm: Random Forest Regressor

Input Features: - OHLC prices - Volume - Market capitalization - Engineered features

Training Strategy: - Train-test split (80:20) - Time-series aware splitting (no shuffling)

Output: Trained ML model

7.6 Model Evaluation Module

Evaluation Metrics: - RMSE (Root Mean Squared Error) - MAE (Mean Absolute Error) - R^2 Score

Purpose: - Measure prediction accuracy - Validate generalization ability

---

7.7 Hyperparameter Tuning Module

Method: GridSearchCV

Parameters Tuned: - Number of estimators - Maximum depth

Output: Optimized model

7.8 Deployment Module

Framework: Streamlit

Workflow: 1. User enters market parameters 2. Input data is formatted 3. Model predicts volatility 4. Result is displayed

Model Storage: - Saved using Joblib (`model.pkl`)

---

8. Error Handling & Logging

- Input validation at deployment layer
- Exception handling during data loading
- Logging of prediction errors

---

9. Security Considerations

- Local deployment only
- No sensitive user data stored
- Read-only dataset access

---

10. Conclusion

The HLD and LLD together define a robust, modular, and scalable architecture for cryptocurrency volatility prediction. The system design ensures ease of maintenance, reproducibility, and alignment with real-world financial analytics use cases.

End of HLD & LLD Document