MAJOR PROJECT SRS

On

Bitcoin Price Predictor

Submitted in the partial fulfilment of the requirement for the award of degree

Bachelor of Technology

In

Computer science & Artificial Intelligence

BATCH (2022-2026)

Submitted to: -
Dr Ridhi kapoor

Submitted by: -
Manoj rana
12200520

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
DAV UNIVERSITY
JALANDHAR-PATHANKOT NATIONAL HIGHWAY, NH 44,
SARMASTPUR PUNJAB
144012

TABLE OF CONTENT

Introduction

1.1Purpose

The primary objective of this Software Requirements Specification (SRS) is to precisely docu- ment all the functional, non-functional, interface, and data requirements for the Bitcoin Price Predictor with Visualizations (BPPV) (BPPV) application. This document serves as the founda- tional contract between stakeholders, developers, and testers, establishing a clear scope and roadmap for the system's development. It specifies the necessary components for data acquisi- tion, time-series modeling, predictive forecasting, and interactive data presentation, ensuring the final product meets the high expectations of financial enthusiasts and data analysts. This SRS ensures that the resulting system is robust, accurate, secure, and highly maintainable.

1.2Document Conventions

This document uses the following conventions for clarity and compliance:

- Shall: Denotes a mandatory, non-negotiable requirement that must be implemented.
- Should: Denotes a highly desired or recommended feature that may be deferred or ex- cluded if constraints require it.
- May/Could: Denotes an optional feature or a future enhancement possibility.
- BPPV: Shorthand for the Bitcoin Price Predictor with Visualizations system.
- BTC/USD: The Bitcoin price paired against the US Dollar, which is the system's focus.
- API: Application Programming Interface, referring to external data sources.
- OHLCV: Stands for Open, High, Low, Close, and Volume—the standard metrics for candle- stick charting.

1.3Intended Audience and Reading Suggestions

This SRS is prepared for a diverse audience involved in the BPPV project:

Project Managers: Sections 1, 2, 5, and 7 are critical for understanding the business con- text, overall scope, quality assurance measures, and project verification plan.

- Developers/Engineers: Sections 3, 4, and 6 provide the technical blueprint, defining re- quired interfaces, algorithms (LSTM, ARIMA), specific functional logic, and the database schema.
- Testers/Quality Assurance (QA) Staff: Sections 4, 5, and 7 are essential for creating test cases, defining performance benchmarks, and verifying compliance with all functional and non-functional requirements.
- Technical Writers: Sections 1 and 2 provide the context needed for developing user man- uals and internal system documentation.

1.4 Product Scope

The Bitcoin Price Predictor with Visualizations (BPPV) is a data intelligence platform focused exclusively on forecasting the price of Bitcoin (BTC) against the U.S. Dollar.

1. Core Capability: Deploying and managing advanced time-series models, specifically Long Short-Term Memory (LSTM) and ARIMA, for BTC price prediction.
2. Prediction Horizons: Generating forecasts for two fixed horizons: a short-term 24-hour window (hourly granularity) and a medium-term 7-day window (daily granularity).
3. Data Sources: The system scope covers ingestion of historical BTC price and volume data from a third-party API. Initially, the scope explicitly excludes alternative data sources such as sentiment analysis, social media trends, or macro-economic indicators.

4. Deliverable: The final system shall be a secure, responsive, web-based application pro- viding interactive data visualizations, model performance metrics, and a controlled envi- ronment for model selection (for Analyst Users).

The system is intended to provide informational insights and shall not be construed as offering financial advice.

1.5References

- IEEE Std. 830-1998, Recommended Practice for Software Requirements Specifications.
- Box-Jenkins Methodology for ARIMA Modeling and Time Series Analysis (Core methodol- ogy for FR-2.3).
- Documentation for the selected backend language (Python) and deep learning library (e.g., TensorFlow/Keras).

Overall Description

2.1Product Perspective

The BPPV system is designed as a sophisticated, independent prediction engine situated within the broader financial data ecosystem. It is a standalone application, meaning it does not inte- grate into a larger enterprise platform, but it has critical external dependencies. The core archi- tecture comprises a data pipeline that autonomously extracts, transforms, loads, and models data, reporting results to a visual front-end layer.

2.2User Classes and Characteristics

The system identifies three distinct user groups with varied permissions and requirements:

1Standard Users (Level 1):

Goal: Quickly assess short-term and medium-term price trends for general informa- tional purposes.

Skills: Familiarity with basic cryptocurrency concepts and chart reading. Minimal tech- nical expertise required.

Key Requirements: Clear, easy-to-read dashboard (UI-1), reliable real-time updates (NFR-5.3), and simple visualization exports (FR-3.3).

2Analyst Users (Level 2):

Goal: Evaluate the prediction model's effectiveness, compare methodologies, and fine- tune parameters for specific research or trading strategies.

Skills: Strong understanding of statistics, time-series modeling (LSTM, ARIMA), and machine learning metrics (MAE, RMSE).

Key Requirements: Access to Configuration Panel (UI-3), model selection/retraining control (FR-2.3), and detailed backtesting reports (FR-2.4). Requires authenticated access (FR-4.1).

Administrator:

Goal: Maintain system health, ensure data pipeline reliability, and manage infrastruc- ture.

Skills: Expertise in containerization (Docker), cloud deployment, database manage- ment, and system monitoring.

Key Requirements: System logs, pipeline health checks, and database management tools (outside the scope of this SRS but assumed).

2.3 Operating Environment

The BPPV is designed for a modern, scalable cloud environment.

Operating Environment Specifications

Component	Specification
Deployment Platform	Containerization (Docker) for consistent environment

	across development and production. Orchestrated via Kubernetes or similar service for high availability (NFR- 5.7).
Operating System com-	Linux (e.g., Ubuntu, Alpine) on the server, ensuring patibility with Python data science libraries.
Frontend Client CSS3,	Modern web browsers supporting ES6+, HTML5, and WebSockets. Must be fully responsive for mobile.
Backend Framework development,	Python 3.10+ using FastAPI or Flask for API leveraging asynchronous capabilities for high through- put.
Modeling Libraries	TensorFlow/Keras or PyTorch for LSTM, and the Statsmodels library for ARIMA. Pandas for data ma- nipulation.
Database series data	PostgreSQL 14+ (or MongoDB for flexible time- modeling) to provide scalable storage and indexing for time-series data (DM- 6.1).
Communication Protocol WebSockets	HTTPS (TLS 1.2+) for all RESTful API calls and for low-latency real-time updates (NFR-5.4).

2.4 Design and Implementation Constraints

- Time-to-Data Constraint: The full data acquisition and preprocessing pipeline (FR-1.1, FR-1.3) triggered by a request or scheduled job shall complete within 5 seconds to ensure near real-time relevance of the data.
- Retraining Constraint: The model retraining process (FR-2.1) is

computationally inten- sive; it should be scheduled during off-peak hours (e.g., 00:00 UTC) and must not impact the system's prediction serving latency (NFR-5.1).

- Library Licensing Constraint: All third-party libraries, frameworks, and tools employed must be open-source with licenses (e.g., MIT, Apache 2.0) that permit commercial use without restrictions.
- Prediction Window Constraint: The system shall not generate predictions beyond the 7-day forecast window, as the model's accuracy degrades significantly past this point.

2.5 Assumptions and Dependencies

- Data Integrity: It is assumed that the chosen third-party data API provides data that is reasonably accurate, free of major gaps, and delivered with correct timestamps. The sys- tem design includes minimal error-checking for API data (FR-1.3).
- API Reliability: The external data API must provide an uptime of at least 99.5% and shall consistently adhere to its documented rate limits. Changes to API endpoints or data schemas are considered a major risk dependency.
- Hardware Availability: Adequate computational resources (specifically GPU access or multi-core CPU instances) for training and running complex LSTM models must be con- tinuously available in the deployment environment.
- Model Predictive Power: It is assumed, for the purposes of system design, that the cho- sen models (LSTM, ARIMA) can achieve a minimum baseline accuracy (e.g., RMSE below a defined threshold) over the short-term horizon, although this is validated by FR-2.4, not guaranteed by the SRS.
- Security Infrastructure: It is assumed that the deployment environment provides the necessary infrastructure for TLS termination, firewalling, and denial-of-service protectionExternal Interface Requirements

3.1 User Interfaces

The user interface design focuses on an elegant, functional, and data-centric experience.

UI-1: Dashboard (Primary View)

- Requirement: The dashboard shall serve as the landing page and principal point of in- teraction, providing a high-level summary of BTC performance and immediate prediction results.
- Key Elements:
  1. Current Price Card: A large, prominent display showing the current BTC/USD price, its 24-hour change (%), and the price update timestamp (FR-1.2).
  2. Forecast Summary: A text panel summarizing the key forecast value (e.g.,” BTC is forecasted to reach $X in 7 days”), including the 90% confidence range.
  3. Chart Embed: An embedded, full-width instance of the Interactive Charting view (UI- 2), set by default to the last 30 days of history plus the 7-day prediction.
- Responsiveness: The dashboard shall use a mobile-first design approach, ensuring all key metrics and the primary chart scale correctly on devices down to 360px width.

UI-2: Interactive Charting

- Requirement: The charting component shall provide a high degree of interactivity, allow- ing users to analyze historical trends and prediction fidelity.
- Data Representation: The chart shall display historical OHLCV data using a standard Candlestick or OHLC bar format, including volume data in a

sub-panel (FR-3.1).

- Interaction: Users shall be able to:

  1. Pan across the historical timeline.
  2. Zoom in and out using mouse/touch gestures.
  3. Select and view data points to see specific OHLCV values.

- Prediction Overlay (FR-3.2): The future prediction (24-hour or 7-day, based on user selec- tion) shall be overlaid as a separate line plot, clearly delineated from historical data. The 90% confidence interval shall be rendered as a shaded area surrounding the prediction line.
- Technical Indicators: The chart shall support the activation of at least two widely used technical indicators: a Simple Moving Average (SMA, configurable periods) and the Rela- tive Strength Index (RSI) displayed in a separate pane.

UI-3: Configuration Panel (Analyst Access Only)

- Requirement: This secured interface (FR-4.2) shall provide Analyst Users with control over prediction model settings and system diagnosis.
- Model Selection (FR-2.3): The panel shall include a radio button or drop-down selection allowing the Analyst to switch the active prediction model between LSTM and ARIMA. The selection must trigger the use of the selected model's last known prediction.
- Model Retraining Control: A button labeled "Retrain Model Now" shall be provided, which triggers the asynchronous retraining process (FR-2.1) with the current look-back window setting. The system shall display a status indicator (e.g., "Training: 75% Com- plete") during this process.
- Look-Back Window Adjustment: An input field shall allow the Analyst to set the number of historical days used for model training (default 90 days).
- Backtesting View (FR-2.4): A table/chart interface shall display the recent performance metrics (MAE, RMSE) for the currently selected model,

detailing the results of the last 30 retraining cycles.

3.2 Software Interfaces

SI-1: Data Feed API (External)

- Protocol: The system shall use HTTPS GET requests to retrieve data from the third-party API.
- Endpoint (Example): /api/v3/exchanges/coindata?symbol=BTCUSD&start=YYYY-MM-DD&end=YYYY- MM-DD&interval=D
- Required Fields: The JSON response must contain, at minimum, the following fields for each historical period:
  1. time (Timestamp/Date)
  2. open (Opening Price)
  3. high (Highest Price)
  4. low (Lowest Price)
  5. close (Closing Price)
  6. volume (Trading Volume)
- Error Handling: The system shall implement exponential backoff and retry logic for API calls that return status codes 429 (Rate Limit) or 5xx (Server Error).

SI-2: Database Interface (Internal)

- Technology: The backend shall interface with the PostgreSQL database using a standard Python ORM (e.g., SQLAlchemy) or a direct client library (e.g., Psycopg2).
- Read/Write Operations:
  1. Write: Insertion of new real-time data (FR-1.2), saving new model parameters (FR-2.1), and logging backtesting results (FR-2.4).
  2. Read: Retrieval of historical data for charting (FR-3.1) and prediction data for overlay (FR-3.2).

•Connection Pool: The system shall maintain a database connection pool to optimize re- source usage and adhere to the Prediction Latency NFR (NFR-5.1).

3.3 Communications Interfaces

The communication architecture is hybrid, utilizing both REST and real-time protocols.

- Standard Requests (RESTful HTTPS): All non-real-time data (historical charts, configura- tion changes, user authentication, backtesting reports) shall be handled via RESTful API endpoints, all secured via TLS (NFR-5.4). Data payload formats shall be JSON.
- Real-Time Updates (WebSockets): A dedicated WebSocket endpoint shall be used exclu- sively for pushing the latest BTC price and volume data to the connected clients (NFR-5.2). This avoids the overhead of frequent polling for dynamic data.
- Authentication Token Management: Successful user authentication (FR-4.1) shall issue a short-lived JSON Web Token (JWT) that must accompany all subsequent RESTful and Web- Socket requests for secured endpoints**System Features (Functional Requirements)**

4.1 Historical Data Fetch (Initial Load)

- Input: API access credentials, BTC/USD symbol, initial timestamp (e.g., January 1, 2018).
- Process: The data ingestion service shall make sequential API calls to retrieve all historical daily OHLCV data up to the current date. The process must respect API rate limits (SI-1).
- Output: Data is inserted into the PriceData entity (DM-6.1). A success log shall be gener- ated.
- Validation: Before insertion, the system shall check for and log any

duplicate timestamps or periods of missing data.

Real-Time Polling

- Process: The system shall initiate a polling job every 60 seconds to fetch the latest avail- able minute-level or hourly data point from the API.
- Data Handling: If the fetched price is newer than the latest price in the database, the system shall update the PriceData entity and immediately push the new price data to all active WebSocket clients (NFR-5.2).
- Timestamp Integrity: The system shall ensure that the displayed price is synchronized within the 90-second delay tolerance (NFR-5.3).

Data Preprocessing

- Trigger: Executed immediately prior to any model training (FR-2.1).
- Steps: The pipeline shall perform the following transformations on the look-back window data:
  1. Feature Engineering: Calculate required features like moving averages, percentage changes, and lag features based on the closing price.
  2. Missing Value Imputation: Use forward-fill or linear interpolation to handle small gaps (up to 3 consecutive data points). Gaps larger than 3 points shall trigger a warn- ing log.
  3. Normalization: Apply Min-Max Scaling to transform the data into a range (0, 1) as required by the LSTM model.
- Output: The scaling parameters (min/max values) shall be stored in the ModelConfiguration entity (DM-6.3) for later inverse transformation.

4.2 Prediction Modeling

FR-2.1: Model Training (LSTM)

- Algorithm: A multi-layer, stacked LSTM network shall be implemented for the primary prediction model.

- Schedule: The training job shall be automatically scheduled to execute daily, 24 hours after the last completion time.
- Training Data: The model shall train exclusively on the latest 90 days of preprocessed historical data (FR-1.3).
- Output: The fully trained model weights and architecture shall be serialized and saved as a new version in the database. The new model performance metrics (MAE/RMSE) shall be updated in DM-6.3.

FR-2.2: Prediction Generation

- Trigger: Executed immediately following successful model training (FR-2.1) and upon ex- plicit request from the UI.
- Forecasts: The system shall use the latest trained model to generate:
  1. A 24-hour prediction sequence with hourly granularity.
  2. A 7-day prediction sequence with daily granularity.
- Confidence Interval (90%): The prediction process shall generate the upper and lower bounds of the 90% confidence interval, likely using Monte Carlo Dropout or similar statis- tical techniques to quantify uncertainty.
- Storage: The final forecasted price points and their associated confidence bounds shall

be stored in the PredictionResults entity (DM-6.2).

FR-2.3: Analyst Model Selection

- Access: Only authenticated Analyst Users (Level 2) shall be able to access this feature via UI-3.
- Functionality: The user shall be able to select the prediction backend, switching between the LSTM model and the secondary ARIMA model.
- Impact: Upon selection, the system shall immediately load the latest prediction results (FR-2.2) from the selected model and update the UI-2 chart overlay within the latency con- straint (NFR-5.1).

FR-2.4: Backtesting Report

- Access: Only authenticated Analyst Users (Level 2) shall be able to view this report via UI-3.
- Content: The report shall display a historical log detailing the Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) achieved by the model after each of its last 30 daily training cycles.
- Visualization: The system should provide a line chart of MAE over the 30 cycles to visualize prediction performance stability.

4.3 Visualization and Reporting

FR-3.1: Historical Visualization

- Data Source: Fetches the last 90 days of OHLCV data from the PriceData entity (DM-6.1).
- Indicators: The chart library shall support calculating and rendering user-configurable technical indicators, defaulting to a 20-day Simple Moving Average (SMA) and the 14- period Relative Strength Index (RSI).

FR-3.2: Prediction Overlay

- Display Logic: The system shall retrieve the prediction data (FR-2.2) corresponding to the currently selected model (FR-2.3) and visually overlay the forecasted price path and the 90% confidence band (UI-2).
- Clarity: The predicted price line and the confidence band boundary lines must be ren- dered with high contrast and distinct colors to differentiate them clearly from historical data.

FR-3.3: Export Functionality

- Chart Export: The system shall provide a “Download Chart” button enabling the user to export the current view (including historical data, indicators, and predictions) as a high- resolution PNG or SVG image file.

- Data Export: The system shall allow the user to export the raw prediction data (times- tamp, forecasted price, lower bound, upper bound) as a universally readable CSV file.

4.4 User Management (for Level 2 Access)

FR-4.1: User Authentication

- Method: Authentication shall be performed via standard username and password login.
- Credential Handling: Passwords shall be hashed and salted using a modern, robust al- gorithm before storage (NFR-5.5).
- Token Issuance: Upon successful login, the system shall issue a secured JWT for session management.

Non-functional Requirements

5.1 Performance

Performance is critical for a high-frequency data system.

NFR-5.1: Prediction Latency

- Metric: The time taken for the backend to retrieve pre-calculated prediction data from the database and transmit it to the client.
- Target: The latency shall not exceed 500 milliseconds for 95% of all data retrieval re- quests.
- Measurement: Performance stress testing (VP-7.3) using tools like Apache JMeter or Lo- cust.

NFR-5.2: Real-Time Throughput

- Metric: The capacity of the WebSocket server to maintain stable, low-latency connections under load.
- Target: The WebSocket connection shall handle up to 1,000 concurrent users

receiving real-time price updates without the message delay exceeding 1 second.

- Measurement: Concurrent connection testing, monitoring CPU/memory usage of the WebSocket server.

NFR-5.3: Data Synchronization

- Metric: The time difference between the latest price reported by the external API and the price displayed on the user's Dashboard.
- Target: The displayed" current price" shall be no more than 90 seconds delayed from the source API's current reported price, accounting for the 60-second polling interval (FR-1.2) and processing time.
- Measurement: End-to-end timing checks on the data polling, ingestion, and WebSocket broadcast sequence.

5.2 Security

NFR-5.4: Encryption

- Requirement: All data transmission, including RESTful API calls and WebSocket traffic,

shall be encrypted end-to-end using TLS 1.2 or higher.

- Implementation: Requires SSL/TLS certificates on the server and forced redirection from HTTP to HTTPS.

NFR-5.5: Data Protection

- Requirement: User authentication credentials shall be stored securely.
- Implementation: Passwords shall be hashed using Argon2 or bcrypt with a computa- tionally expensive work factor, ensuring protection against rainbow table and brute-force attacks.

NFR-5.6: Input Validation

- Requirement: The system shall prevent common injection and scripting vulnerabilities.

- Implementation: All user input fields (e.g., look-back window setting, login fields) must be validated on both client-side and server-side. Server-side validation shall sanitize and cast inputs to expected types (e.g., integer for look-back window) to mitigate SQL Injection or Cross-Site Scripting (XSS) risks.

5.3 Reliability

NFR-5.7: Availability

- Target: The system shall maintain an uptime of 99.9% (equivalent to approximately 43 minutes of unscheduled downtime per month).
- Implementation: Achieved through redundancy in deployment (container orchestration) and automated health checks.

NFR-5.8: Disaster Recovery

- Requirement: The data tier shall be protected against data loss.
- Implementation: The production database shall be backed up incrementally every 6 hours. The Recovery Point Objective (RPO) shall not exceed 6 hours and the Recovery Time Objective (RTO) shall not exceed 4 hours.

5.4 Maintainability and Extensibility

NFR-5.9: Code Standards

- Requirement: The codebase shall be easily understood and modified by new developers.
- Implementation: All Python code shall adhere strictly to PEP 8 style guidelines. All func- tions, classes, and modules must include detailed docstrings explaining purpose, param- eters, and return values.

NFR-5.10: Model Modularization

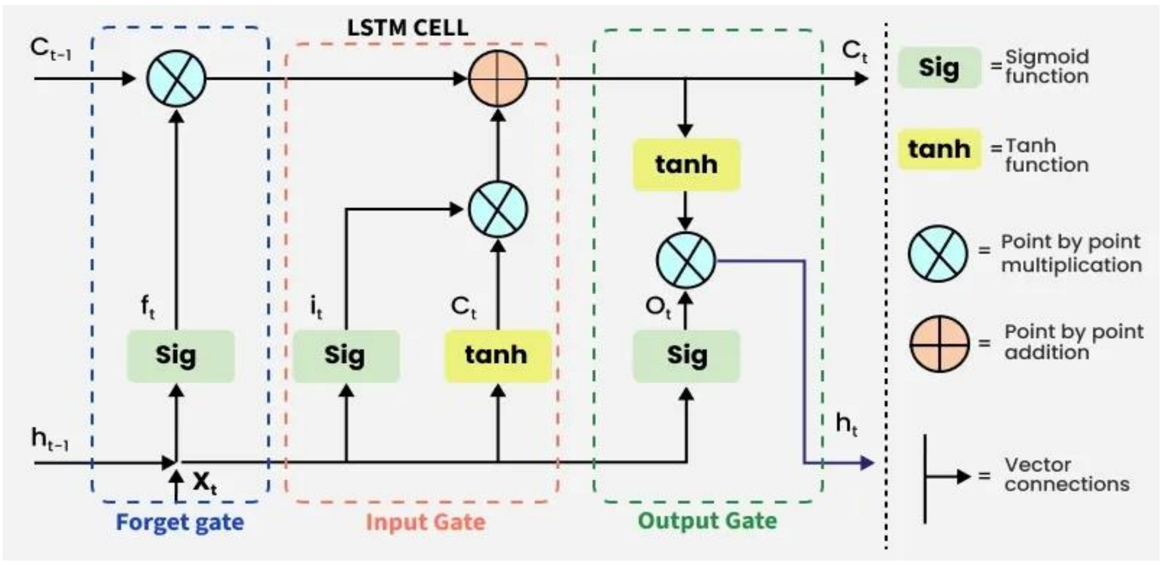
- Requirement: The core prediction logic must be adaptable for future model integration.

- Implementation: The prediction service shall use a Strategy Pattern or similar design pat- tern, abstracting the model loading, training, and prediction interfaces. This allows new models (e.g., Prophet, other RNN architectures) to be added as plugins without changing the core data pipeline or API endpoints.

Data Model Requirements

6.1Logical Data Model Overview

The system's data is organized around three primary entities: PriceData for time-series inputs, PredictionResults for model outputs, and ModelConfiguration for system metadata and model diagnostics.



Verification Plan

Testing Strategy

The verification plan ensures that the BPPV meets all documented requirements using a multi- stage testing approach. The strategy focuses heavily on data quality, model integrity, and user experience under load.

VP-7.1: Unit Testing

- Scope: All individual, non-integrated components (functions, classes) in the backend shall be tested. This includes data cleaning routines, scaler transformation/inverse transforma- tion logic, and utility functions for calculating technical indicators.

- Coverage: Target 90% statement coverage for all core Python prediction modules, uti- lizing Python’s pytest framework.

VP-7.2: Integration Testing

- Scope: Testing the seamless data flow between the three tiers: Data Acquisition → Pre- processing → Model → Database → API Endpoint → Frontend.
- Scenarios:
  1. Verify that data fetched from SI-1 is successfully stored in DM-6.1.
  2. Verify that a newly trained model (FR-2.1) successfully writes its prediction sequence into DM-6.2.
  3. Verify that retrieving historical data (FR-3.1) and prediction data (FR-3.2) on the UI simultaneously loads and overlays correctly.

VP-7.3: Performance Testing

- Scope: Stress-testing critical NFRs related to speed and capacity.
- Scenarios:
  1. Latency Test (NFR-5.1): Simulate 100 concurrent requests for prediction results and measure the 95th percentile response time.
  2. Throughput Test (NFR-5.2): Simulate 1,000 active WebSocket connections for 1 hour and confirm that the system maintains the real-time update rate and stability.