1. **Problem Statement**

The project involves the development, testing, and deployment of a service to retrieve and display the historical Bitcoin price. The service should support user-defined start and end dates, as well as an output currency (including default USD). Additionally, the service should calculate and display the highest and lowest Bitcoin prices within the provided date range, alongside the price data in the selected currency.

Features and Requirements:

a. User Input:

* Start Date
* End Date
* Output Currency (Default is USD)
* Offline Data Flag (Toggle to enable offline functionality)

b. Output***:***

* A UI (e.g., React app) that displays historical Bitcoin prices for each day within the given date range, including:
  + - Date (DD-MM-YYYY)
    - Bitcoin Price with text indicating whether it's the highest or lowest price in the range
    - Output Currency value (e.g., INR, USD)

c. Service***:***

* The service should fetch Bitcoin historical prices and convert them into the specified currency using live rates.
* The service should also handle cases where dependencies (e.g., public API) are unavailable, ensuring the service continues to work in offline mode.
* Accessible via a web browser or Postman for testing.

d. Deployment:

* Ready for production/live environment deployment.

1. **API Data Sources**

* Historical Bitcoin Price Data:
  + URL: https://api.coindesk.com/v1/bpi/historical/close.json
* Supported Currencies:
  + URL: https://api.coindesk.com/v1/bpi/supported-currencies.json
* Documentation:
  + Coindesk API Docs (<https://www.coindesk.com/coindesk-api>)

1. **Expected Output**

* The following data should be presented to the user:
* Date (DD-MM-YYYY): e.g., 01-01-2018
* Price: Display the Bitcoin price along with tags:
  + (high) if it's the highest in the date range.
  + (low) if it's the lowest in the date range.
* Currency Conversion: Support both INR and USD, with the ability to switch between them.

1. **Non-Functional Requirements (NFRs)**

* Architecture Principles:
  + Demonstrate SOLID principles.
  + Implement 12 Factor App principles.
  + HATEOAS (Hypermedia as the engine of application state) for REST API.
* Design Patterns:
  + Utilize relevant design patterns in the solution.
  + Performance, Optimization, & Security:
  + Ensure the service performs well with large datasets and handles currency conversions efficiently.
  + Implement basic security practices, especially in handling sensitive data like API keys.
* Production Readiness:
  + The code should be deployable in a production environment with appropriate configurations.
  + Test-Driven Development (TDD) & Behavior-Driven Development (BDD):
  + Demonstrate unit tests and behavior-driven testing to ensure quality.
* Data Protection:
  + Sensitive information like API keys should be encrypted or protected.

1. **Documentation**

* API Documentation:
  + Include OpenAPI spec/Swagger documentation to allow users to understand the API endpoints, expected inputs, and possible error codes.
* Project Documentation:
  + Create a detailed README.md file explaining:
  + Purpose of the project
  + Design and implementation approach
  + Technologies used
  + Setup instructions
  + API usage and examples
  + Sequence diagram or flowchart (created using draw.io)

1. **Build and Deployment**

* Continuous Integration (CI)
  + Build a CI/CD pipeline using Jenkins for automating builds and deployments.
  + Ensure that all pipeline scripts and Jenkins job configurations are part of the codebase.
* Continuous Deployment (CD)
  + Dockerization: Deploy the service in a Docker container.
    - Create a Docker image for local deployment.
    - Ensure that the Dockerfile is part of the project sources.
* Instructions for IDE Setup:
  + Configure the application to run on port 8080 as that is the only available port for previewing.
  + If encountering network or bandwidth issues, the following actions should be taken:
  + Share a screenshot of the issue.
  + Develop and test locally using your preferred IDE and tools.
  + Upload the project files to QuizME or share a zip file containing the executable uber jar with all dependencies.

1. **User Stories**

* User Story 1: As a user, I want to input a date range and currency so that I can view the historical Bitcoin prices for that period in my preferred currency.
* User Story 2: As a user, I want to see the highest and lowest Bitcoin price for the given date range so that I can easily identify the price extremes.
* User Story 3: As a user, I want to switch between currencies (INR, USD) so that I can view Bitcoin prices in my preferred currency.
* User Story 4: As a user, I want to use the service in offline mode, so I can still access the last available price data when there is no internet connection.

1. **Acceptance Criteria**

* The UI should allow the user to select a date range and currency.
* The Bitcoin price should be displayed for each date in the range, with the high/low markers.
* The service should handle failures of external dependencies gracefully and operate in offline mode.
* The service should be deployable via Docker and be available via both a web interface and Postman.
* The code should follow industry best practices for design, testing, and security.