X:Change

Tanish Seth
Vaidant Jain
Siddhi Murkute
Surab Sebait
Sophiya Ahmad

September 26, 2024

Abstract

This documentation provides a detailed overview of the Exchange Rate Visualization project, including its functionalities, API specifications, software development lifecycle (SDLC) methodologies, testing strategies, installation, and running instructions.

Contents

1 Introduction

The **Exchange Rate Visualization** project provides users with a powerful and dynamic interface to visualize historical exchange rates. Designed with user experience in mind, this application empowers individuals, businesses, and analysts to make informed decisions by offering insightful visualizations of currency trends.

Users can effortlessly select different currencies from a wide range of options and explore exchange rate trends over various time periods, including daily, weekly, monthly, and custom ranges. This flexibility allows for a comprehensive analysis of market movements and historical performance.

One of the standout features of this application is its ability to assess market volatility. By implementing risk indicators, users can easily gauge the stability of different currencies and understand the potential risks associated with their investments or transactions. The risk indicators categorize volatility into low, medium, and high levels, providing a clear understanding of market dynamics.

Whether you are a traveler seeking the best exchange rates, a business owner managing international transactions, or an investor analyzing market trends, the Exchange Rate Visualization project serves as a vital tool for navigating the complex world of currency exchange.

2 Functionalities

- **User Interface**: Interactive and responsive UI with light/dark mode toggle.
- **Currency Selection**: Allows users to select currencies for comparison.
- **Period Selection**: Options to view data over predefined periods (daily, weekly, monthly) or custom date ranges.
- **Risk Indicators**: Visual cues to indicate low, medium, and high volatility based on historical data.
- **Chart Visualization**: Graphical representation of exchange rates over selected periods.
- **Data Refresh**: Real-time updates of exchange rates when a user fetches new data.

3 API Specifications

The application interacts with a backend API to fetch exchange rate data. Below are the primary endpoints:

3.1 Fetch Exchange Rates

- **Endpoint**: `GET /api/exchangeRates/currency/period`
- **Parameters**:
  - `currency`: The currency code (e.g., USD, EUR).
  - `period`: Timeframe for which to fetch rates (daily, weekly, monthly, or custom).
- **Response**:

```
{
    "rates": [
        {
            "date": "YYYY-MM-DD",
            "rate": float
        },
        ...
    ]
}
```

3.2 Error Handling

In case of an error, the API returns a standard error response:

```
{
    "error": "Error message describing the issue"
}
```

3.3 Example API Testing

Here are examples of how to test the API using ‘curl‘:

- **Get Daily Rates for USD**:

  ```
  curl -X GET http://localhost:5000/api/exchangeRates/USD/daily
  ```

- **Get Weekly Rates for EUR**:

  ```
  curl -X GET http://localhost:5000/api/exchangeRates/EUR/weekly
  ```

- **Get Custom Rates for GBP** (from 2023-01-01 to 2023-01-10):

  ```
  curl -X GET http://localhost:5000/api/exchangeRates/GBP/custom/2023-01-01:2023-01-10
  ```

4 Software Development Lifecycle (SDLC)

The project follows the Agile SDLC methodology, which emphasizes iterative development and flexibility. The key phases include:

- **Planning**: Define the project scope, gather requirements, and outline the project's objectives.
- **Design**: Create wireframes and design the architecture of the application.
- **Development**: Implement the application using React for the frontend and Node.js/Express for the backend.
- **Testing**: Conduct unit, integration, and user acceptance testing to ensure quality.
- **Deployment**: Deploy the application on a cloud platform.
- **Maintenance**: Regular updates and improvements based on user feedback.

5 Testing Strategy

The project employs the following testing strategies:

5.1 Unit Testing

Each component and function is tested individually to ensure correctness. Jest is used as the testing framework.

5.2 Integration Testing

Tests are conducted to verify that different modules work together as expected.

5.3 User Acceptance Testing (UAT)

End-users test the application to validate its functionality and usability.

5.4 Running Tests

To run tests, execute the following command:

```
npm test
```

6 Installation

To install the project, use npm:

```
npm install my-project
cd my-project
```

7 Run Locally

To run the project locally, follow these steps:

1. Clone the project:

   ```
   git clone https://{link}
   ```

2. Go to the backend directory:

   ```
   cd northern/backend
   ```

3. Install dependencies:

   ```
   npm install
   ```

4. Start the server:

   ```
   npm run
   ```

5. Alternatively, you can run:

   ```
   node server.js
   ```

6. Open the front end by running the `index.html` file in your browser.

8 Tech Stack

Client: HTML, CSS, JavaScript, Chart.js
Server: Node.js **Database:** MongoDB

9 Environment Variables

To run this project, you will need to add the following environment variables to your .env file:

- API_KEY
- ANOTHER_API_KEY

10 Color Reference

Color	Hex
Low Volatility	#0a192f
Medium Volatility	#00b48a
High Volatility	#d10000
Background Color	#f8f8f8

11 Contributing

Contributions are always welcome! Here's how you can get involved:

11.0.1 Ways to Contribute

1. **Report Bugs**: If you find any bugs or issues while using the application, please report them in the issue tracker. Provide as much detail as possible to help us address them effectively.
2. **Feature Requests**: Have an idea for a new feature? We'd love to hear your suggestions! Open an issue and describe your idea, including why you think it would be beneficial for the project.
3. **Code Contributions**:
   - **Fork the Repository**: Start by forking the project to your own GitHub account.
   - **Create a Branch**: Create a new branch for your feature or bug fix (e.g., `feature/new-feature` or `bugfix/fix-issue`).
   - **Make Changes**: Implement your changes and ensure they are well-documented and tested.
   - **Open a Pull Request**: Once you're ready, submit a pull request detailing your changes and referencing any relevant issues.

11.0.2 Guidelines

- **Code Quality**: Please ensure that your code follows the existing style and conventions of the project. Use meaningful variable names and add comments where necessary.
- **Documentation**: If you add new features or make significant changes, please update the documentation accordingly to reflect these changes.
- **Testing**: Ensure that your contributions are well-tested. If you are adding new features, consider including unit tests to verify their functionality.

11.0.3 Code of Conduct

We expect all contributors to adhere to the project's code of conduct, which promotes a positive and respectful environment for everyone involved. Please be courteous, constructive, and welcoming to all participants.

12 Appendix

12.0.1 Documentation

- **API Documentation**: For detailed information on the API endpoints and their usage, please refer to the Documentation.
- **User Guide**: A comprehensive user guide is available, covering all functionalities and how to navigate the application. You can find it here.

12.0.2 Tools and Libraries

- **Chart.js**: Used for rendering interactive charts to visualize exchange rate data.
- **Axios**: Utilized for making HTTP requests to the backend API.
- **Node.js**: The server-side platform that powers the backend application.
- **Express**: A minimal and flexible Node.js web application framework used to build the API.

12.0.3 Testing

- **Postman Collection**: A Postman collection is available for testing the API endpoints.
- **Unit Tests**: Instructions for running unit tests are provided in the `tests` directory. Ensure you have the necessary setup to execute these tests.

12.0.4 Acknowledgments

We would like to thank all contributors and the open-source community for their invaluable support and resources that have made this project possible.