WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Real-Time Forex Calculator: A Cloud-Based Currency Conversion System Using API Integration and DevOps Practices

Keerthana R

Dayananda Sagar college Of engineering

A B S T R A C T :

This paper presents the design, development, and deployment of a cloud-based Forex Calculator application that enables real-time currency conversion using live exchange rate APIs. Developed during an internship at Tata Consultancy Services (TCS), the project integrates modern frontend technologies, secure backend processing, and robust cloud infrastructure. GitHub was used for source control and continuous integration/deployment pipelines to ensure efficient, versioned, and collaborative development. Emphasis was placed on data security, system scalability, and responsive design. The application demonstrates industry practices such as Infrastructure as Code (IaC), API key protection, monitoring, and compliance with data protection regulations.

Keywords: Forex Calculator, Real-Time Currency Conversion, GitHub, DevOps, Cloud Deployment, API Integration, Flask, Node.js, CI/CD, TCS, Infrastructure as Code, Secure Web Applications.

Introduction

Foreign exchange plays a pivotal role in global trade, investment, and travel. Accurate and real-time currency conversion is essential for individuals and enterprises alike. This project introduces a Forex Calculator—a real-time, cloud-hosted application that integrates foreign exchange APIs to provide reliable currency conversion data. The project was conceptualized as part of an internship at TCS, aligning with the company's digital transformation and cloud computing vision. The **Forex Calculator** project was developed to address this gap by building a secure, scalable, and real-time currency conversion tool as part of an academic internship with **Tata Consultancy Services (TCS)**. This web-based application enables users to convert between global currencies using live exchange rates fetched from third-party APIs such as CurrencyLayer or OpenExchangeRates. The project aligns with TCS's vision of cloud-first digital transformation and demonstrates the practical application of cloud-native architectures, API integration, DevOps practices, and modern UI/UX principles.

From a technical standpoint, the system features a responsive frontend built using HTML5, CSS3, and JavaScript, supported by a backend developed in Python using Flask (or alternatively Node.js). It incorporates robust security practices like HTTPS communication, API key protection via environment variables, and role-based access control. Deployment on cloud platforms such as AWS or Microsoft Azure ensures high availability and scalability.

Furthermore, **GitHub** was used for source control, with CI/CD pipelines implemented through GitHub Actions to enable automated testing and deployment.

Beyond its functional capabilities, the project serves as a live case study for integrating industry-standard software development workflows into academic learning. It exposes students to critical skills such as cloud deployment, version control, agile development, and cybersecurity—preparing them for real- world careers in software engineering and fintech. Additionally, the calculator is designed with extensibility in mind, allowing for future enhancements such as cryptocurrency support, historical rate tracking, and multilingual localization.

This paper outlines the system's architecture, technology stack, deployment model, and security considerations. It also highlights the contributions of TCS's Cloud Infrastructure Unit (CIU) in enabling infrastructure provisioning, automation, and compliance—demonstrating how professional cloud practices can be applied to student-led development initiatives.

1.1. Structure

The Forex Calculator is designed with a modular and scalable architecture that separates concerns across different system layers. This structure facilitates independent development, testing, deployment, and maintenance, ensuring the system remains flexible and extensible over time. The architecture follows a **client-server model** and includes components such as the frontend, backend, third-party APIs, database (optional), and cloud infrastructure.

1.1.1 Frontend (Client Interface)

The frontend is the user-facing component of the application. It is built using HTML5, CSS3, and JavaScript, ensuring responsive design across desktops, tablets, and mobile devices. Users can input the amount to be converted, select source and target currencies, and receive real-time conversion results. The interface includes input validation and error handling to ensure a smooth user experience.

1.1.2 Backend (Server Logic)

The backend, developed in Python using Flask (or alternatively Node.js with Express), handles request processing, API communication, and response formatting. It performs the currency conversion logic, fetches data from external APIs, and ensures secure storage and handling of API keys. The backend also manages business rules and formats the data sent back to the frontend.

1.1.3 API Integration Layer

This layer is responsible for communicating with external foreign exchange rate providers such as CurrencyLayer or OpenExchangeRates. The backend sends HTTPS requests to these APIs, retrieves live exchange rates in JSON format, and extracts the required data for computation. Authentication is managed using secure API keys stored in environment variables.

1.1.4 Database (Optional)

Although the application can function without persistent storage, a database such as MySQL or MongoDB may be used to store user preferences, conversion history, and logs. The use of a database supports advanced features like session management, analytics, and history tracking.

1.1.5 Cloud Infrastructure

The system is deployed on cloud platforms like AWS or Microsoft Azure. Cloud services are used for hosting the frontend and backend, provisioning virtual machines (VMs), managing storage, configuring auto-scaling, and monitoring resource usage. Security measures include HTTPS encryption, IAM policies, firewalls, and secure API key vaults.

Illustrations

Visual illustrations provide a deeper understanding of how the Forex Calculator system functions by depicting its architecture, data flow, user interface, and deployment model. The following diagrams help visualize the logical structure, technical workflow, and user interaction within the application.

System Architecture Diagram

The **System Architecture Diagram** outlines the core components and their interactions:

- **Frontend:** Built with HTML, CSS, and JavaScript. Takes user input and displays output.
- **Backend:** Flask/Node.js server handles logic and external API requests.
- **External APIs:** Provides real-time exchange rates (e.g., CurrencyLayer).
- **Database (optional):** Stores user preferences and historical conversions.
- **Cloud Infrastructure:** Hosts and scales the application (AWS, Azure).

Data Flow Diagram (DFD)

The **Data Flow Diagram** represents how data travels through the system:

1. User inputs amount and selects currencies.
2. Frontend sends a request to the backend.
3. Backend calls the external API.
4. Exchange rate is retrieved and conversion is calculated.

5. Backend returns the result to the frontend.

6. (Optionally) data is logged in the database.

UI Mockup / Wireframe

The **User Interface (UI)** is designed for clarity and responsiveness. A simple wireframe includes:

- **Dropdowns** for selecting base and target currencies.
- **Input field** for the amount.
- **Convert button** to trigger calculation.
- **Output display** for the converted amount and exchange rate.

+--+

| Forex Calculator (Responsive UI) |

+--+

| Amount: [100.00] |

| From: [USD ▼] To: [INR ▼] |

| |

| [Convert] |

| |

| Result: ₹ 8,322.00 |

| Rate: 1 USD = 83.22 INR

Deployment Diagram

Shows how the application is hosted and accessed in the cloud:

- Application is containerized with Docker.
- Hosted on a VM or App Service in AWS/Azure.
- Load balancer distributes traffic.
- HTTPS secures

communication. [User Device]

|

v

[Load Balancer] --> [Web App Container]

v

[Exchange Rate API]

v

[Database (Optional)]

Equations

The Forex Calculator relies on fundamental arithmetic operations to convert currency values using real-time exchange rates. Though the calculations are simple, they are critical to ensuring accuracy, consistency, and transparency for end-users. This section outlines the core equations used in the backend logic of the application.

3.1 Currency Conversion Formula

The basic formula used to calculate the target amount based on live exchange rates is:

Converted Amount=Base Amount×Exchange Rate

Where:

Base Amount is the amount entered by the user in the source currency Exchange Rate Exchange Rate is the live rate retrieved from the API (e.g., USD to INR = 83.22) Converted Amount is the output displayed in the target currency

Example:

If a user wants to convert **100 USD to INR**, and the current exchange rate is **83.22**, then:

Converted Amount=100×83.22=8322

Converted Amount=100×83.22=8322

DevOps and GitHub Integration

GitHub was used for version control and collaboration. CI/CD pipelines were implemented using GitHub Actions, enabling continuous integration, automated testing, and seamless deployment. Infrastructure as Code (IaC) was achieved using Terraform for consistency across development environments.

Security and Compliance

The application implements:

- IAM policies
- Data encryption (AES-256 at rest, HTTPS in transit)
- GDPR and ISO/IEC 27001 compliance
- Intrusion Detection Systems (IDS)

Future Work

- Support for cryptocurrency conversions
- Historical and trend-based visualization with Chart.js
- Multilingual and localization support

- Integration with payment gateways

REFERENCES

CurrencyLayer, “Real-time & historical exchange rates API,” [Online]. Available: https://currencylayer.com[2] Open Exchange Rates, “Forex data API for developers,” [Online]. Available: https://openexchangerates.org[3] Flask Documentation, “Flask web development framework,” [Online]. Available: https:// flask.palletsprojects.com/[4] Node.js, “JavaScript runtime built on Chrome's V8 engine,” [Online]. Available: https://nodejs.org[5] GitHub Docs, “GitHub Actions for CI/CD,” [Online]. Available: https://docs.github.com/en/actions[6] Amazon Web Services, “AWS Cloud Services Overview,” [Online]. Available: https://aws.amazon.com/[7] Microsoft Azure, “Cloud Computing Services,” [Online]. Available: https://azure.microsoft.com/[8] M. Fowler, “Continuous Integration,” martinfowler.com, [Online]. Available: https://martinfowler.com/articles/continuousIntegration.html[9] Terraform by HashiCorp, “Infrastructure as Code tool,” [Online]. Available: https://www.terraform.io/[10] B. Burns, B. Grant, D. Oppenheimer, E. Brewer, and J. Wilkes, "Borg, Omega, and Kubernetes," Communications of the ACM, vol. 59, no. 5, pp. 50–57, May 2016.