Currency Exchange Tracker

Complete Project Documentation

Project Overview

The Currency Exchange Tracker is a real-time monitoring system designed to track USD, EUR, and GBP exchange rates against the Sri Lankan Rupee (LKR). The system provides automated alerts and a comprehensive dashboard for monitoring currency fluctuations.

Key Features

Real-time Monitoring
Live tracking of LKR to USD/EUR/GBP exchange rates with automatic updates.

Visual & Audio Alerts
Color-coded indicators and sound notifications when rates exceed thresholds.

Historical Data
View the last 7 exchange rate records with timestamps.

Scheduled Updates
Automatic daily rate fetching at 9 AM with manual refresh option.

Responsive Dashboard
Mobile-friendly web interface accessible from any device.

Technology Stack

Node.js · Express.js · Firebase Firestore · HTML/CSS/JavaScript · FastForex API · node-cron · Server-Sent Events (SSE)

Deployment Guide

Prerequisites

- Node.js v14 or higher
- Firebase Project with Firestore enabled
- FastForex API Account with valid API key
- Git for version control

Step-by-Step Deployment

1. **Clone the Repository**

   ```
   git clone https://github.com/udithanayanajith/currency-tracker-uditha.git cd currency-tracker-uditha
   ```

2. **Install Dependencies**

   ```
   npm install
   ```

3. **Environment Configuration**
   Create a .env file in the project root with the following variables:

   ```
   FASTFOREX_API_KEY=your_api_key_here FASTFOREX_API_URL=https://api.fastforex.io/convert PORT=3000
   ```

4. **Firebase Setup**
   - Create a Firebase project at Firebase Console
   - Enable Firestore Database
   - Generate a service account key and save as serviceAccountKey.json in the project root

5. **Run the Application**

   ```
   npm start
   ```

   Access the dashboard at: http://localhost:3000

API Reference

External API

FastForex API - Used for fetching real-time exchange rates

- **Endpoint:** https://api.fastforex.io/convert
- **Method:** GET
- **Authentication:** API Key required
- **Sample Request:**

```
const response = await axios.get( `https://api.fastforex.io/convert?
from=LKR&to=USD,EUR,GBP&api_key=${API_KEY}` );
```

Internal APIs

Endpoint	Method	Description
/api/current	GET	Returns latest exchange rates with threshold status
/api/rates/history	GET	Returns last 7 rate records with timestamps
/api/fetch-realtime	POST	Manually triggers rate fetch from FastForex API
/api/events	GET	Establishes SSE connection for real-time updates

Configuration

Threshold Settings

Update threshold values in src/services/alertService.js:

```
const thresholds = { USD: 330, // Alert when USD exceeds 330 LKR EUR: 370, // Alert when
EUR exceeds 370 LKR GBP: 420 // Alert when GBP exceeds 420 LKR };
```

Scheduler Configuration

Update scheduling in src/app.js:

```
// Development/testing mode (every 5 minutes) cron.schedule("*/5 * * * *", fetchRates); //
Production mode (daily at 9 AM) cron.schedule("0 9 * * *", fetchRates);
```

Challenges Faced & Solutions

Firebase Cloud Functions & Scheduling Issues

Problem: Initially attempted to use Firebase Cloud Functions with Cloud Scheduler but encountered:

- Node.js version conflicts between development environment and Firebase runtime
- Complex configuration for Cloud Scheduler triggers
- GCP subscription requirements for Firebase Functions Pub/Sub

Solution Implemented

Switched to node-cron library for scheduling within the Node.js application:

- Eliminated Node.js version conflicts
- Simplified scheduling configuration
- No additional cloud service dependencies or billing requirements
- More control over the scheduling logic

Firebase Hosting Configuration

Problem: Deployed site showed default Firebase page instead of the application due to incorrect public directory configuration in firebase.json.

Solution Implemented

Corrected the firebase.json configuration to point to the actual directory containing the front-end build files.

Real-time Data Synchronization

Problem: Implementing reliable real-time updates to all connected clients with proper connection management.

Solution Implemented

Implemented Server-Sent Events (SSE) with proper connection lifecycle management and automatic reconnection handling.