Express Exchange

Express Exchange is a distributed mobile-cloud application that lets users compare live currency exchange rates on the go. The Android app interacts with a Java servlet hosted on the cloud, which fetches real-time data from CoinAPI and logs request analytics to MongoDB. A web-based dashboard provides insights into usage trends and system activity.

1. Native Android Application

My app is called **Express Exchange**.

It contains 3+ views:

1. ‘EditText’ for base currency input
2. ‘EditText’ for target currency input
3. ‘Button’ to trigger the API call
4. ‘TextView’ to display exchange rate
5. ‘ImageView’ to show the Express Exchange logo

It requires user input for currency symbols (e.g., USD → EUR).

It makes an HTTP POST request to the cloud-hosted servlet using ‘HttpURLConnection’, wrapped inside a background thread, ensuring the UI thread is not blocked.

The response from the servlet is a JSON object ‘{ "rate": <number> }’, which is parsed and displayed on screen.

The application is repeatable – users can make multiple conversions without restarting the app.

2. Web Service

Hosted using Java Servlet on GitHub Codespaces with Apache TomEE.

URL: ‘https://<public-codespace-url>/express’

The web service receives POST requests with ‘base’ and ‘target’ parameters, calls the CoinAPI for exchange rates, parses JSON, and responds with the selected value only.

It uses only the required fields from CoinAPI and sends back just what’s needed, avoiding extra data on mobile.

No banned APIs or scraping were used – CoinAPI is a clean, authenticated, JSON-based API.

3. Web Service Logging & Analysis Dashboard

A separate dashboard accessible at ‘/dashboard’ shows usage and analytics.

Logging is done using a MongoDB Atlas cloud database, and the following 6 fields are logged:

1. ‘timestamp’
2. ‘base_currency’
3. ‘target_currency’
4. ‘exchange_rate’ (from API)
5. ‘client_ip’
6. ‘device_info’ (passed from Android header)

Logs are persistently stored using MongoDB Java Sync Driver.

Dashboard Servlet (‘DashboardServlet.java’) fetches this data and sends it to a JSP file (‘dashboard.jsp’) for rendering.

4. Log Useful Information

Each request stores all 6 fields mentioned above.

Server prints request logs and API responses to console for debugging.

5. Store Logs in Cloud MongoDB

Connected to MongoDB Atlas cluster with a 3-node replica set.

Logs are inserted as BSON documents in the ‘logs’ collection.

6. Display Analytics and Logs

The dashboard shows:

- Total number of requests
- Top 5 currency pairs used
- Formatted logs in a table (with timestamp, currencies, rate, device info, IP)

It uses JSP and JSTL with styling inspired by Studio Ghibli aesthetic.

The dashboard avoids showing JSON/XML and is rendered in a human-readable, centered format.

7. Cloud Deployment

The project was deployed using GitHub Codespaces, configured with:

- Dockerfile and ‘.devcontainer.json’
- Public port 8080 enabled
- Project URL made public and tested with Android app

The ‘.war’ file includes both servlet and JSPs, and all dependencies are bundled using Maven.

8. About CoinAPI

For this project, I used the CoinAPI service (https://docs.coinapi.io), which provides real-time and historical data on exchange rates for digital and fiat currencies.

I accessed the Exchange Rate Endpoint using a free API key.

My web service constructs a URL of the form:

‘https://rest.coinapi.io/v1/exchangerate/{base}/{target}’

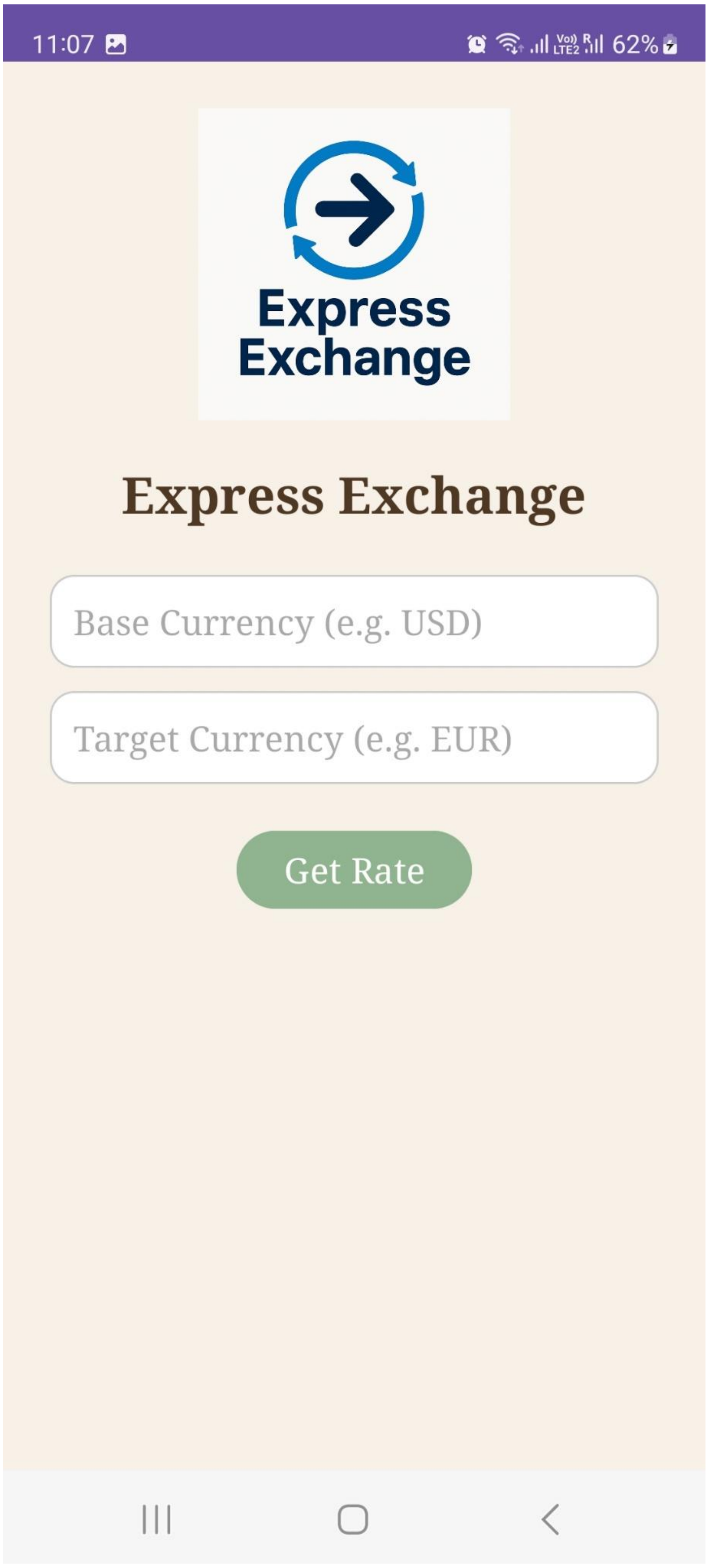
It sends an HTTP GET request with a required header:

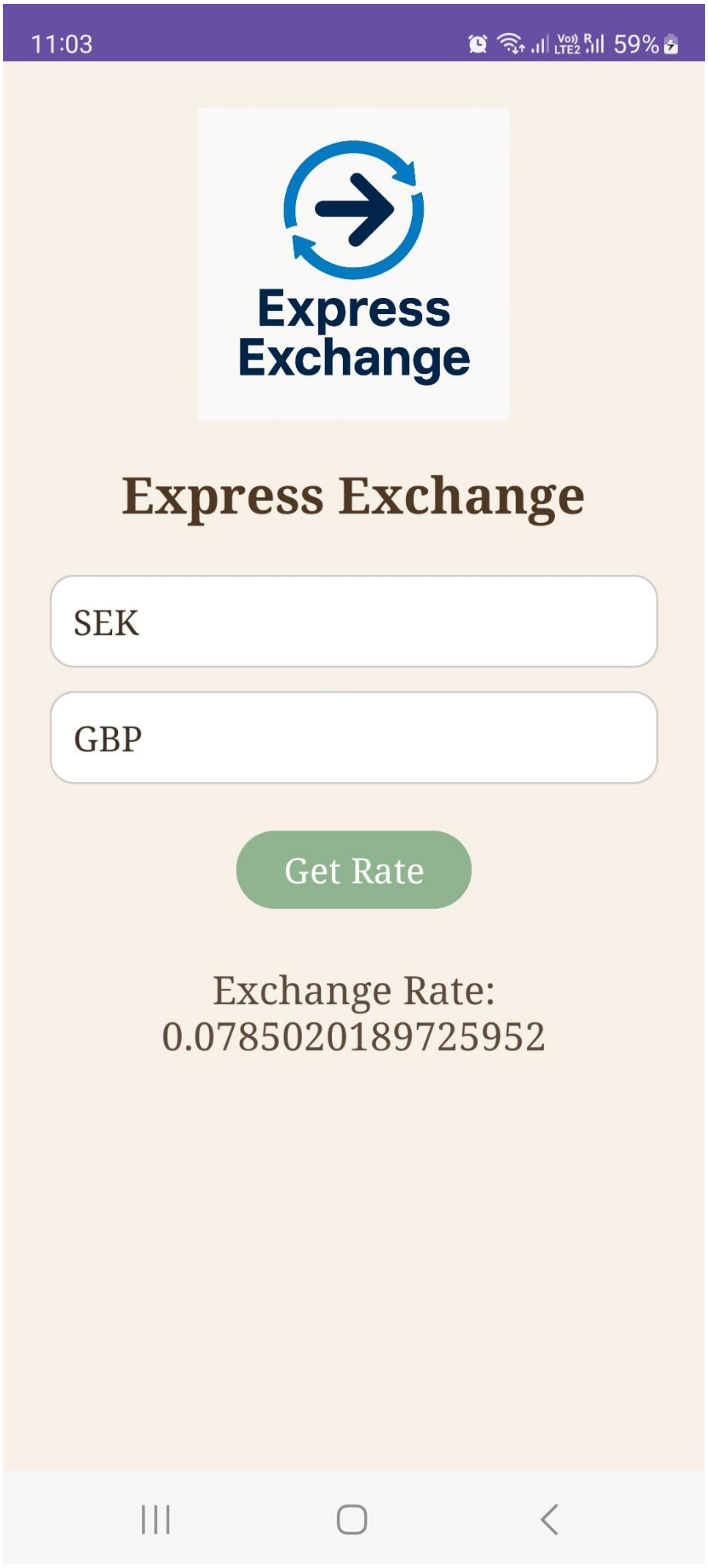
‘X-CoinAPI-Key: <api-key>‘

The response includes the real-time conversion rate in JSON format. I extract and return only the "rate" value to the Android app to reduce data transfer.

I chose CoinAPI because it is well-documented, reliable, fast, and meets all the project requirements.

Android App


11:07
62%
Express Exchange
Express Exchange
Base Currency (e.g. USD)
Target Currency (e.g. EUR)
Get Rate



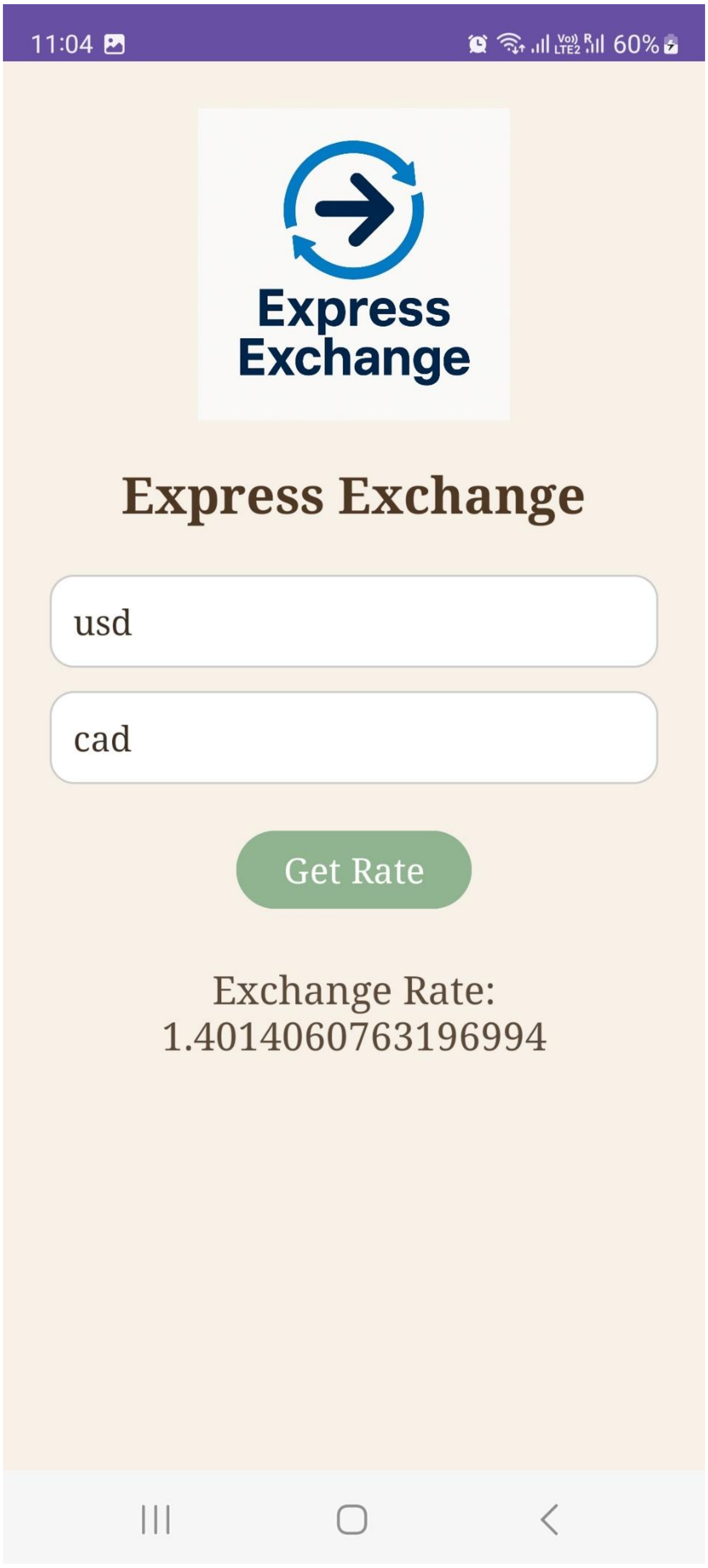
Express Exchange

SEK

GBP

Get Rate

Exchange Rate:
0.0785020189725952



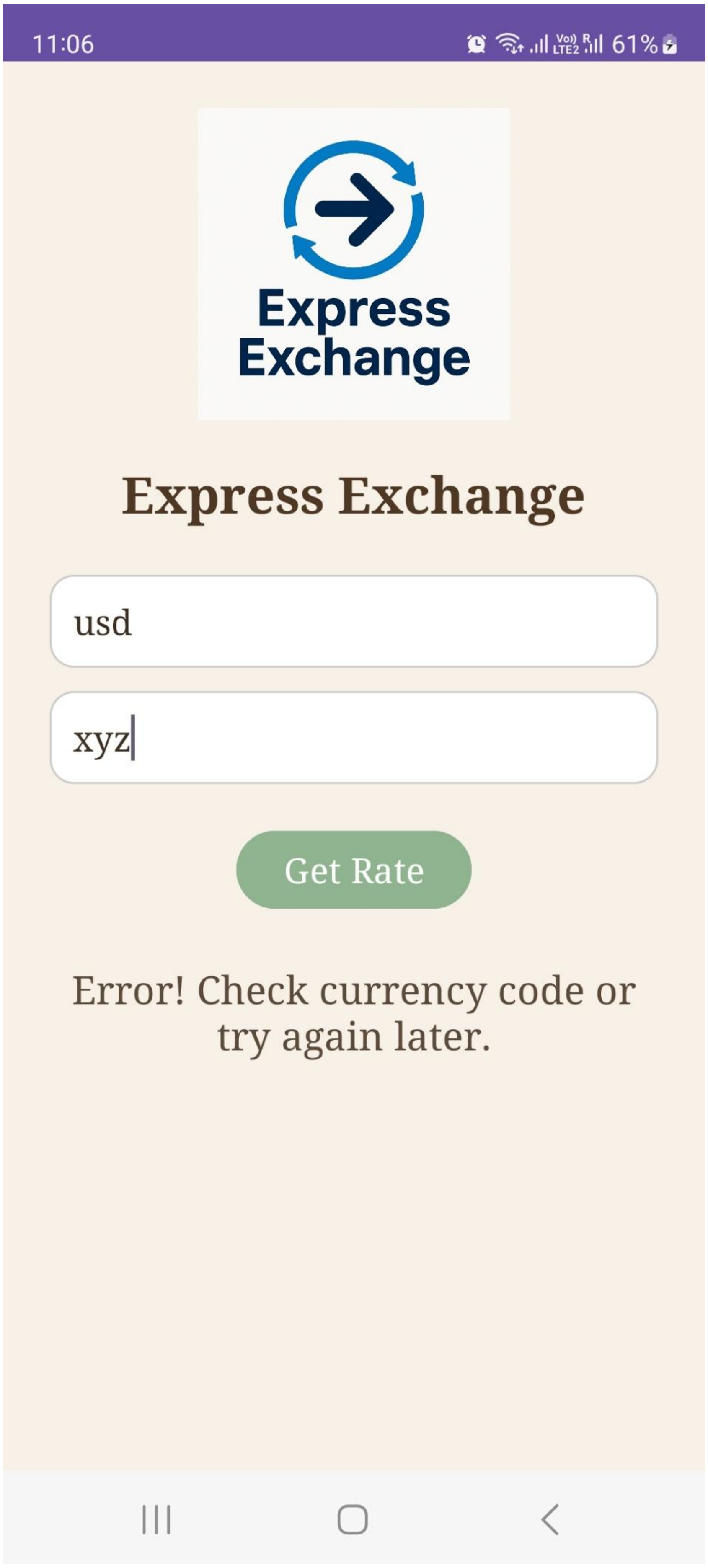
Express Exchange

usd

cad

Get Rate

Exchange Rate: 1.4014060763196994

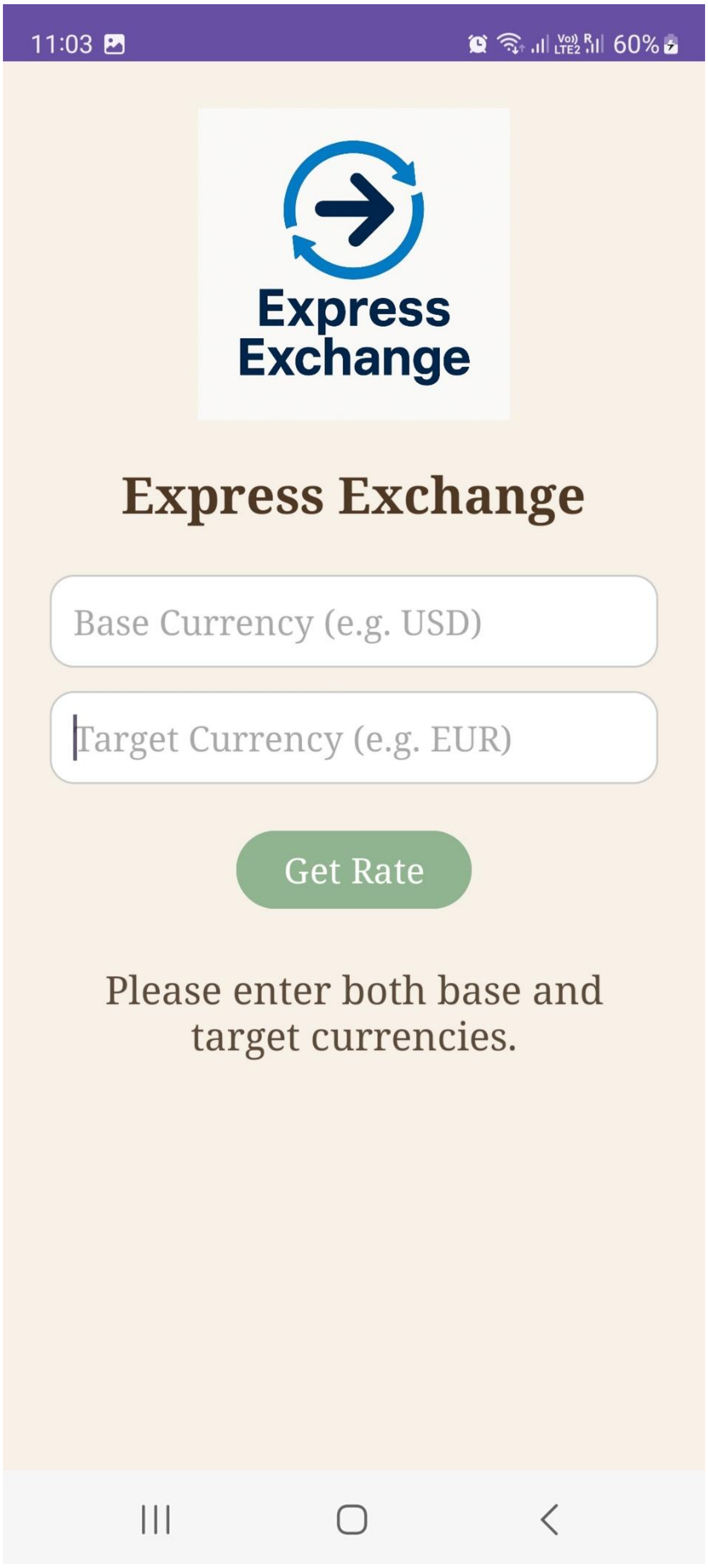


Express Exchange

usd

xyz

Error! Check currency code or try again later.


11:03 60%

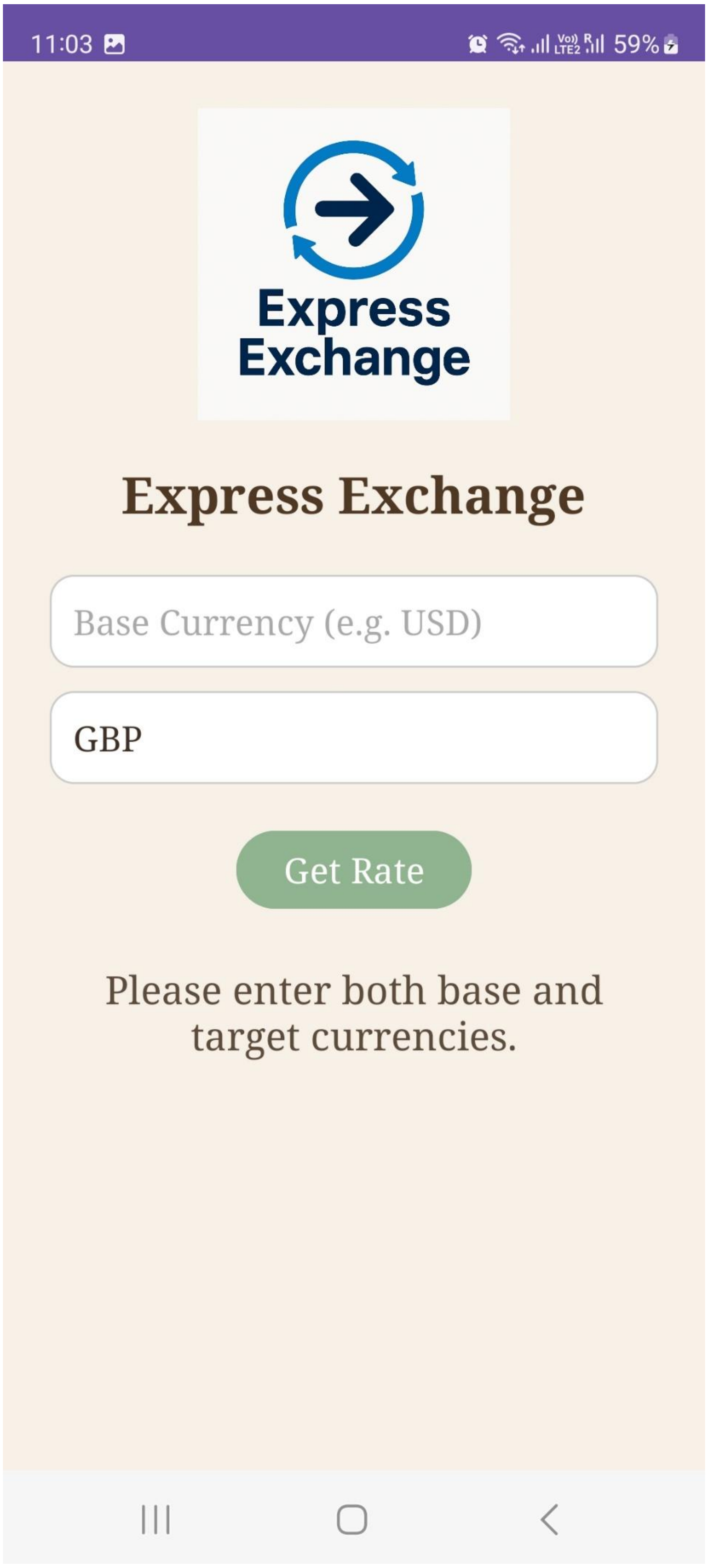
Express Exchange

Base Currency (e.g. USD)

Target Currency (e.g. EUR)

Get Rate

Please enter both base and target currencies.



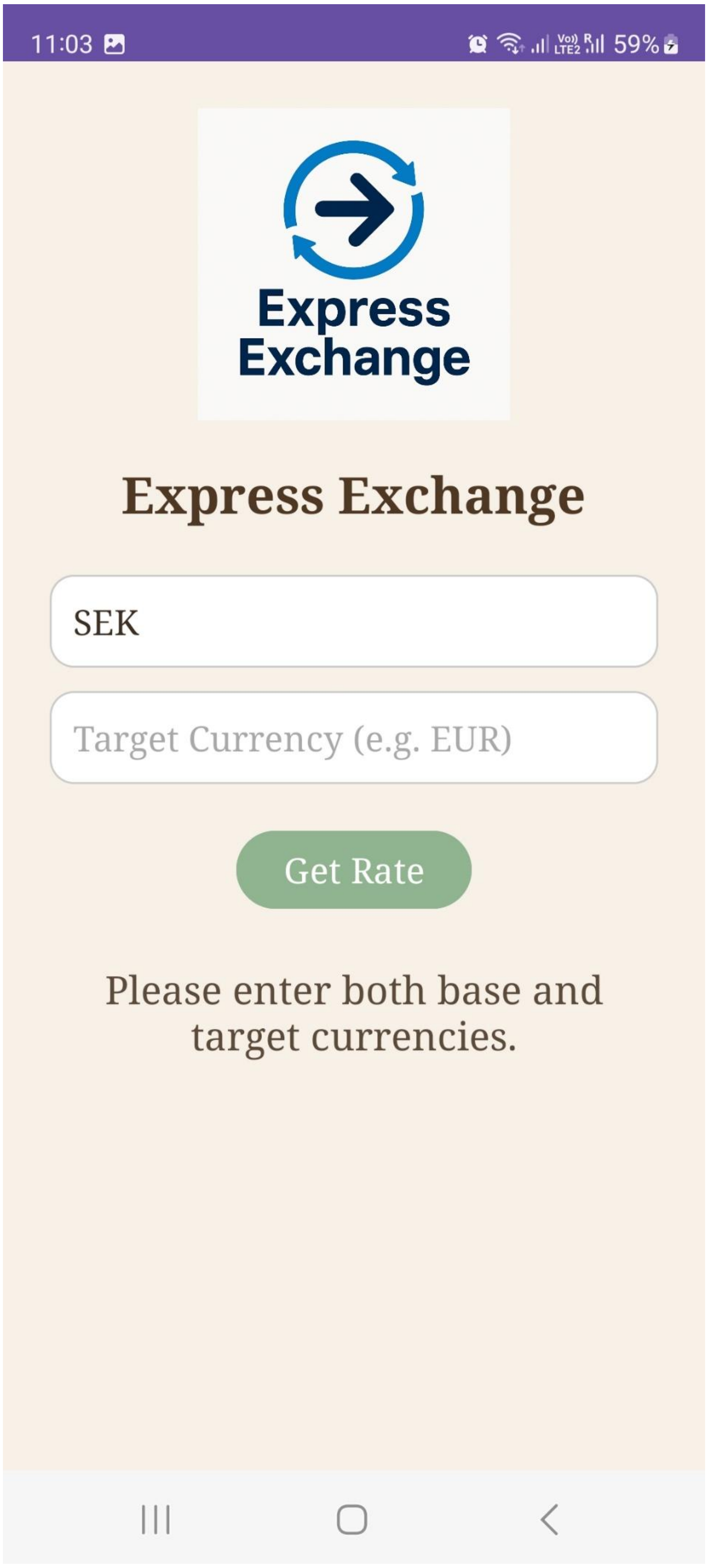
Express Exchange

Base Currency (e.g. USD)

GBP

Get Rate

Please enter both base and target currencies.



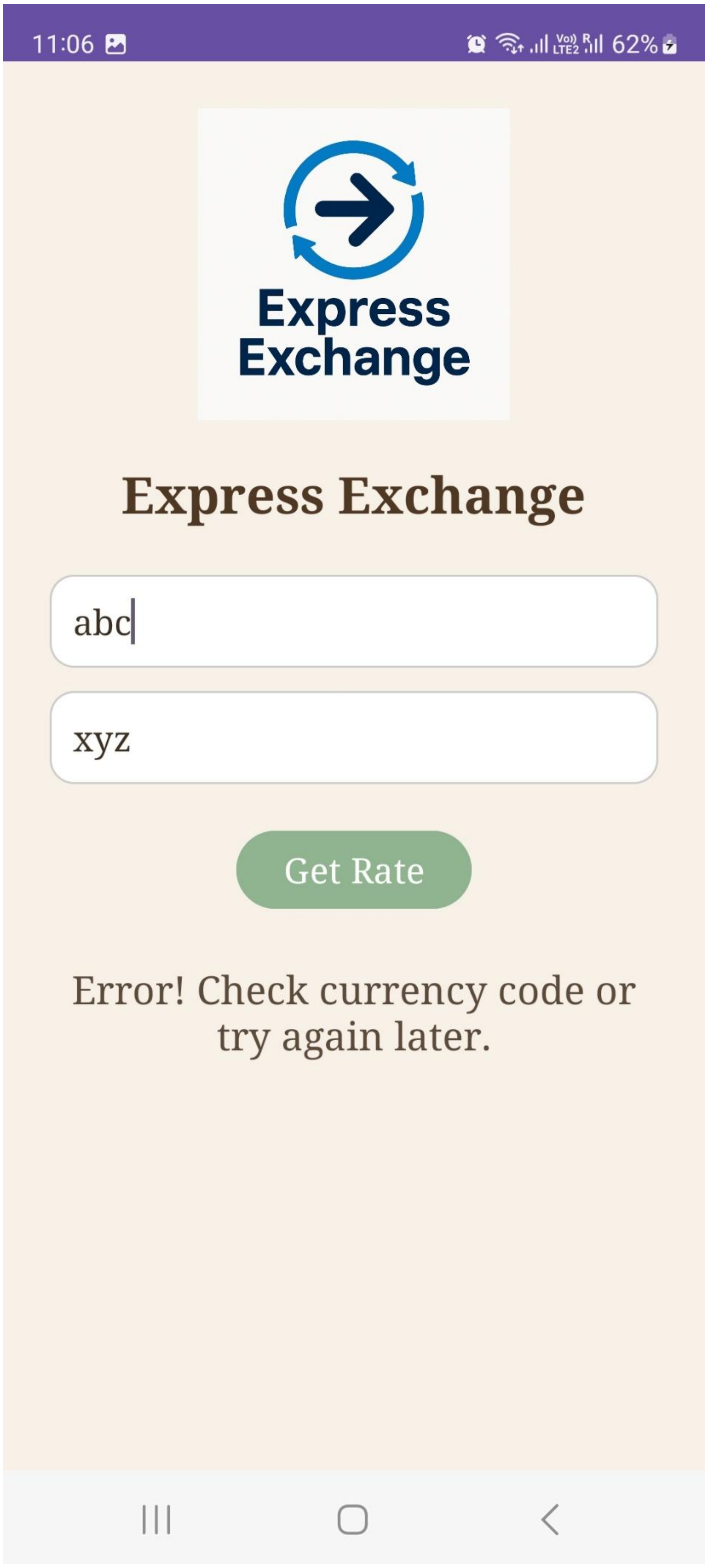
11:03 59%

Express Exchange

SEK

Target Currency (e.g. EUR)

Please enter both base and target currencies.



Express Exchange

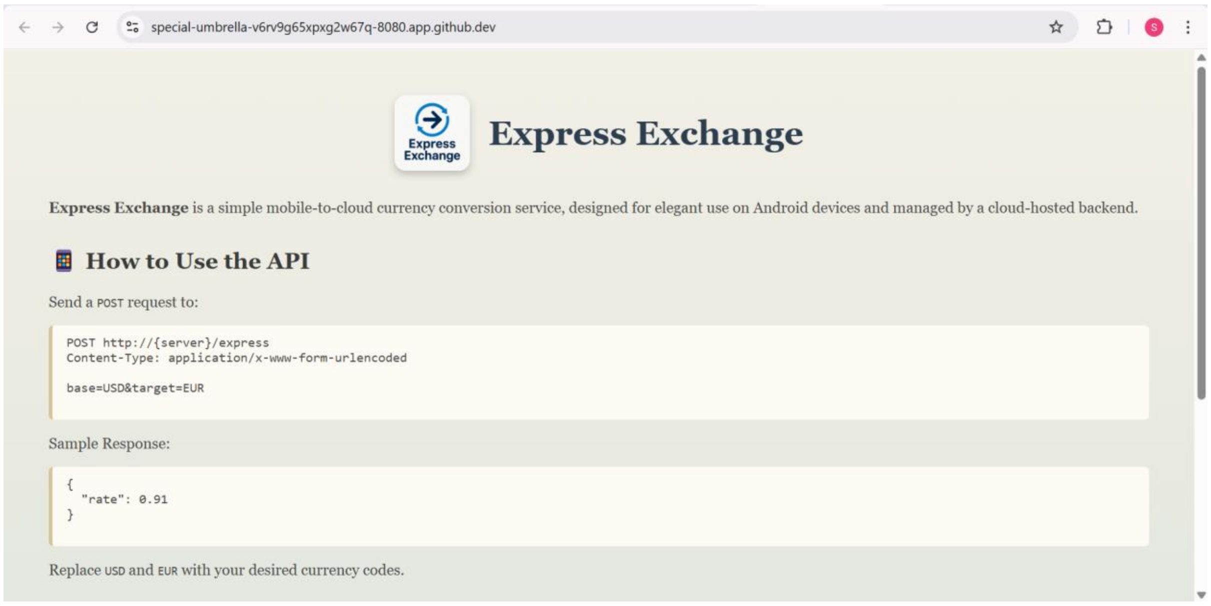
abc

xyz

Error! Check currency code or try again later.

Web Server

Landing page



special-umbrella-v6rv9g65xpxg2w67q-8080.app.github.dev

Express Exchange

Express Exchange is a simple mobile-to-cloud currency conversion service, designed for elegant use on Android devices and managed by a cloud-hosted backend.

📱 How to Use the API

Send a `POST` request to:

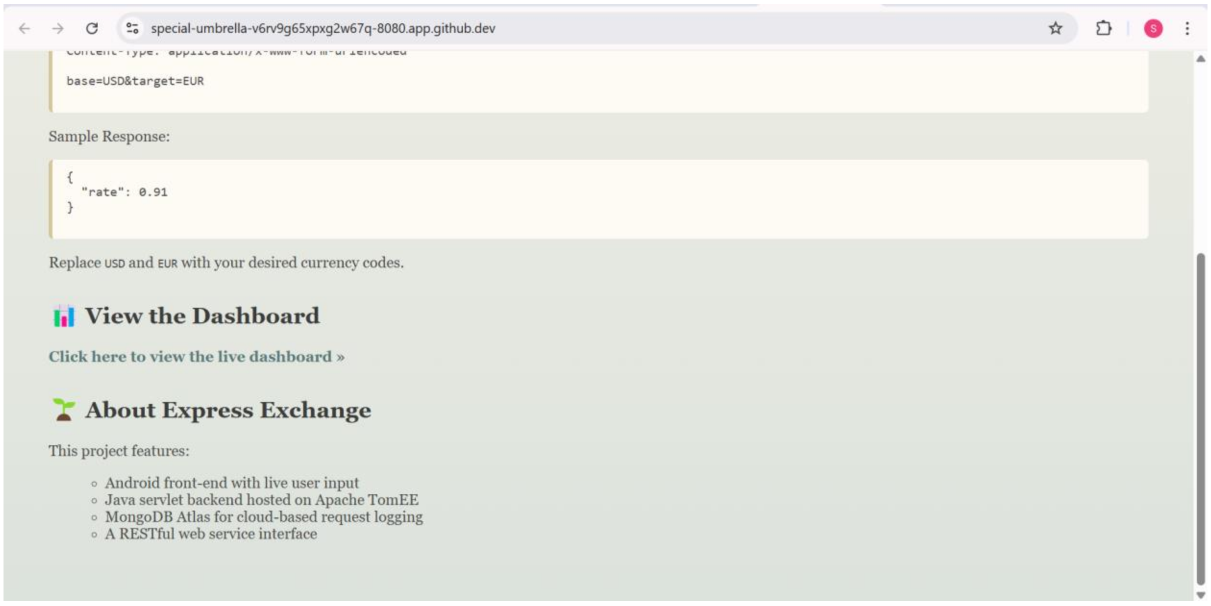
```
POST http://{server}/express
Content-Type: application/x-www-form-urlencoded

base=USD&target=EUR
```

Sample Response:

```
{
  "rate": 0.91
}
```

Replace `USD` and `EUR` with your desired currency codes.



special-umbrella-v6rv9g65xpxg2w67q-8080.app.github.dev

```
base=USD&target=EUR
```

Sample Response:

```
{
  "rate": 0.91
}
```

Replace `USD` and `EUR` with your desired currency codes.

📊 View the Dashboard

Click here to view the live dashboard »

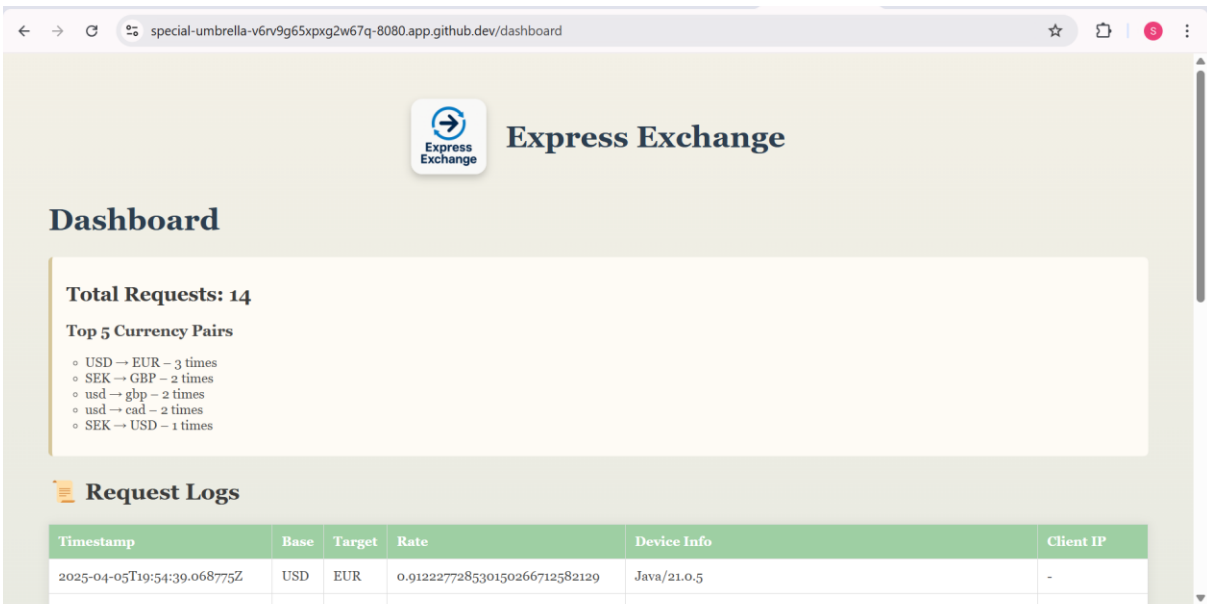
🌱 About Express Exchange

This project features:

- Android front-end with live user input
- Java servlet backend hosted on Apache TomEE
- MongoDB Atlas for cloud-based request logging
- A RESTful web service interface

Web Server

Dashboard



special-umbrella-v6rv9g65xpxg2w67q-8080.app.github.dev/dashboard

Express Exchange

Dashboard

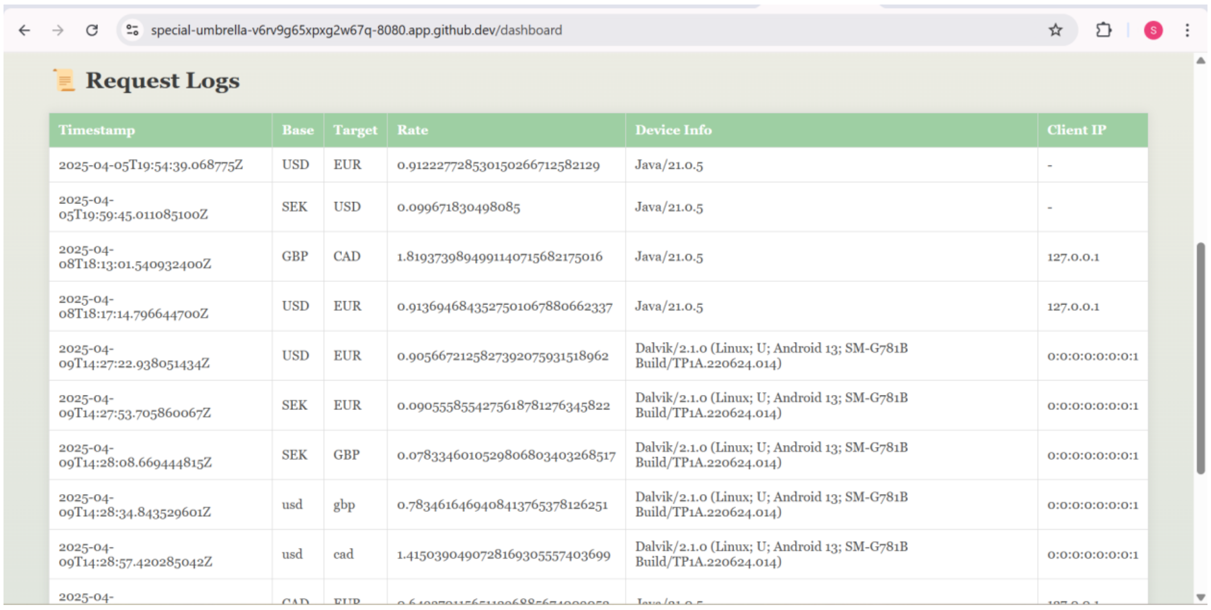
Total Requests: 14

Top 5 Currency Pairs

- USD → EUR – 3 times
- SEK → GBP – 2 times
- usd → gbp – 2 times
- usd → cad – 2 times
- SEK → USD – 1 times

Request Logs

Timestamp	Base	Target	Rate	Device Info	Client IP
2025-04-05T19:54:39.068775Z	USD	EUR	0.912227728530150266712582129	Java/21.0.5	-

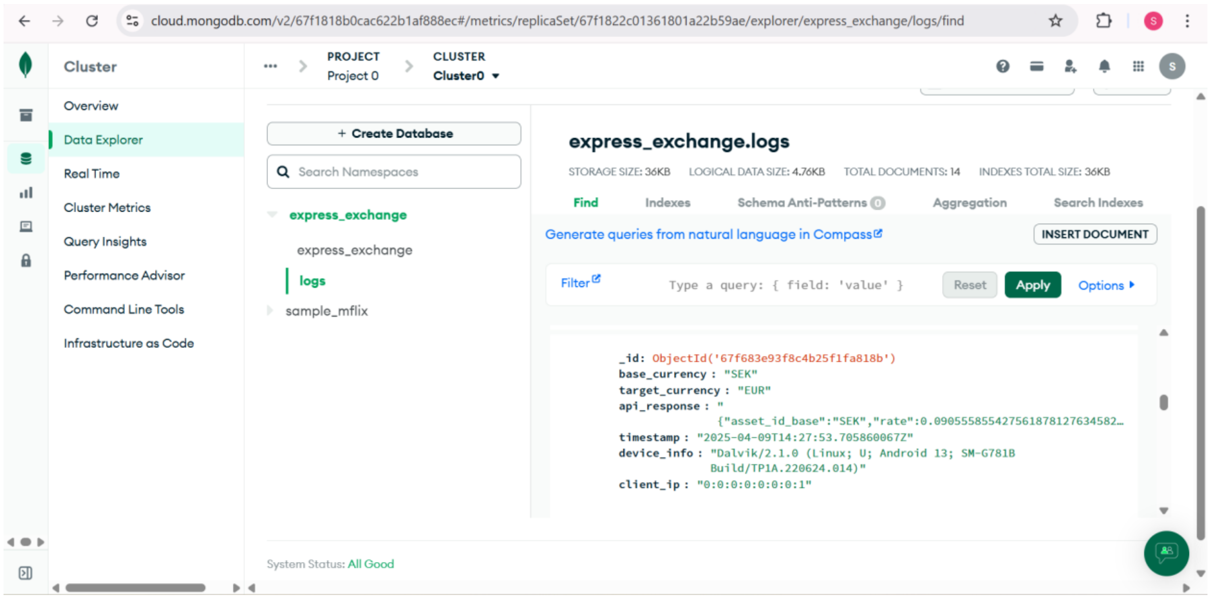


special-umbrella-v6rv9g65xpxg2w67q-8080.app.github.dev/dashboard

Request Logs

Timestamp	Base	Target	Rate	Device Info	Client IP
2025-04-05T19:54:39.068775Z	USD	EUR	0.912227728530150266712582129	Java/21.0.5	-
2025-04-05T19:59:45.011085100Z	SEK	USD	0.099671830498085	Java/21.0.5	-
2025-04-08T18:13:01.540932400Z	GBP	CAD	1.8193739894991140715682175016	Java/21.0.5	127.0.0.1
2025-04-08T18:17:14.796644700Z	USD	EUR	0.9136946843527501067880662337	Java/21.0.5	127.0.0.1
2025-04-09T14:27:22.938051434Z	USD	EUR	0.9056672125827392075931518962	Dalvik/2.1.0 (Linux; U; Android 13; SM-G781B Build/TP1A.220624.014)	0:0:0:0:0:0:0:1
2025-04-09T14:27:53.705860067Z	SEK	EUR	0.0905558554275618781276345822	Dalvik/2.1.0 (Linux; U; Android 13; SM-G781B Build/TP1A.220624.014)	0:0:0:0:0:0:0:1
2025-04-09T14:28:08.669444815Z	SEK	GBP	0.0783346010529806803403268517	Dalvik/2.1.0 (Linux; U; Android 13; SM-G781B Build/TP1A.220624.014)	0:0:0:0:0:0:0:1
2025-04-09T14:28:34.843529601Z	usd	gbp	0.7834616469408413765378126251	Dalvik/2.1.0 (Linux; U; Android 13; SM-G781B Build/TP1A.220624.014)	0:0:0:0:0:0:0:1
2025-04-09T14:28:57.420285042Z	usd	cad	1.4150390490728169305557403699	Dalvik/2.1.0 (Linux; U; Android 13; SM-G781B Build/TP1A.220624.014)	0:0:0:0:0:0:0:1
2025-04-	CAD	EUR	[illegible]	Java/21.0.5	127.0.0.1

MongoDB



cloud.mongodb.com/v2/67f1818b0cac622b1af888ec#/metrics/replicaSet/67f1822c01361801a22b59ae/explorer/express_exchange/logs/find

express_exchange.logs

STORAGE SIZE: 36KB LOGICAL DATA SIZE: 4.76KB TOTAL DOCUMENTS: 14 INDEXES TOTAL SIZE: 36KB

```
_id: ObjectId('67f683e93f8c4b25f1fa818b')
base_currency : "SEK"
target_currency : "EUR"
api_response : "
              {"asset_id_base":"SEK","rate":0.090555855427561878127634582…
timestamp : "2025-04-09T14:27:53.705860067Z"
device_info : "Dalvik/2.1.0 (Linux; U; Android 13; SM-G781B
              Build/TP1A.220624.014)"
client_ip : "0:0:0:0:0:0:0:1"
```

LLM self-reporting:

I have used the help of AI tools for some code snippets and documentation.

Some prompts used:

Java code along with - “document this class without changing any code/formatting”

JSP file along with - “Can you format this jsp in Studio Ghibli design”

Java code along with - “Can you implement proxy design on this class?”