

Deployment Guide for Calculator Application

Part 1: Run the Python Backend with `virtualenv` (one virtual environment per service)

The project includes separate services, for example, the graphing service: `server/graphing` .

Step 1: Create a Virtual Environment for Each Service

1. Navigate to the service directory

```
cd server/graphing
```

2. Create a virtual environment using `virtualenv`

```
virtualenv venv
```

This command creates a `venv/` directory in the current folder.

3. Activate the virtual environment

- **On Windows:**

```
venv\Scripts\activate
```

- **On macOS/Linux:**

```
source venv/bin/activate
```

After activation, you will see `(venv)` appear at the beginning of your terminal prompt.

4. Install required libraries

- If there's a `requirements.txt` file:

```
pip install -r requirements.txt
```

5. Repeat the above steps for other services under `server/`

Other services include `currency-converter` and `scientific-calculator` .

For Windows:

```
cd server/currency-converter
virtualenv venv
venv\Scripts\activate
```

```
cd server/scientific-calculator
virtualenv venv
venv\Scripts\activate
```

Step 2: Run the Backend

The main file for each service is `app.py` :

```
cd server/name-of-service
```

```
python app.py
```



By default, the server runs at `http://0.0.0.0:5001` or `http://127.0.0.1:5001` .

Step 3: Deactivate the Virtual Environment

```
deactivate
```

Part 2: Run the React Frontend (Node.js)

The frontend is located in the `client/calculator` directory.

Step 1: Install Node.js (if not already installed)

- Download from: <https://nodejs.org>

Step 2: Install frontend dependencies

```
cd client/calculator
npm install
```

Step 3: Run the React application

```
npm start
```

The app will automatically open at `http://localhost:3000`

Quick Command Summary Table

Action	Example Command
Create virtual environment	<code>virtualenv venv</code>
Activate virtual env (Windows)	<code>venv\Scripts\activate</code>
Activate virtual env (macOS/Linux)	<code>source venv/bin/activate</code>
Install Python dependencies	<code>pip install -r requirements.txt</code>
Run backend	<code>python app.py</code>
Deactivate virtual environment	<code>deactivate</code>
Install React frontend libraries	<code>npm install</code>
Run React frontend	<code>npm start</code>

Testing Guide for Calculator Application

Part 1: Standard Calculator Testing

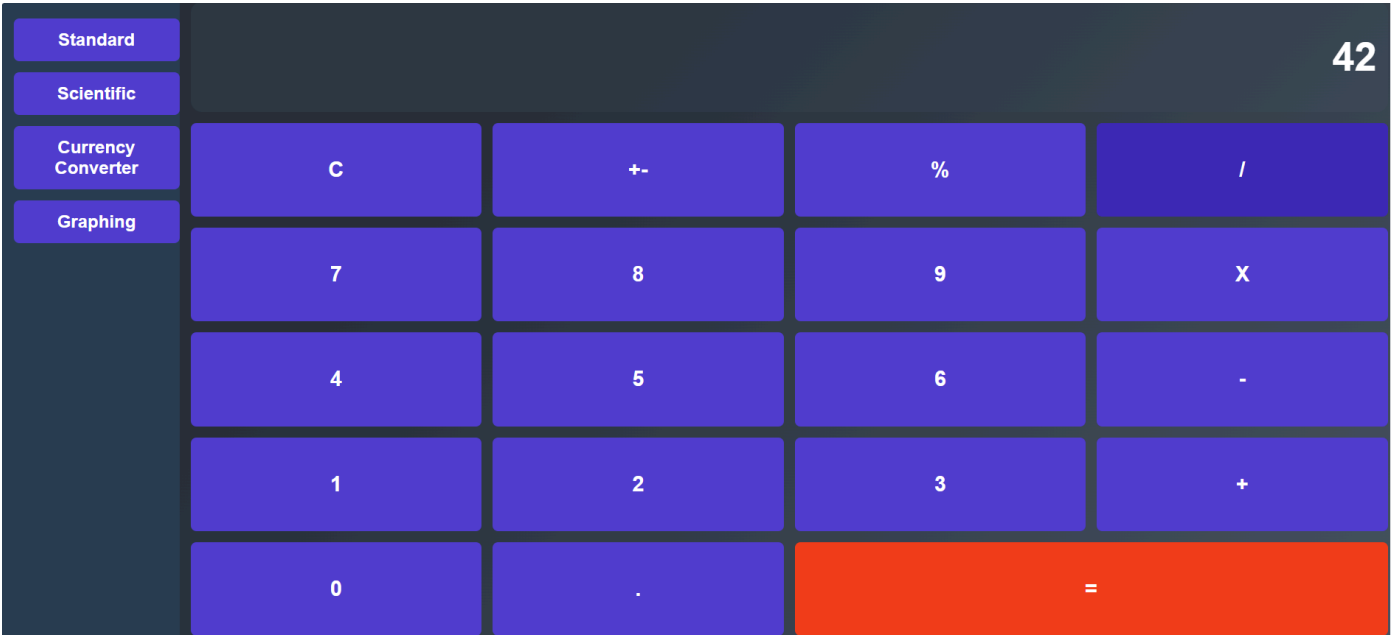
Basic Operations

Test the standard arithmetic operations:

- Addition: $5 + 3 = 8$



- Multiplication: $6 \times 7 = 42$



Part 2: Scientific Calculator Testing

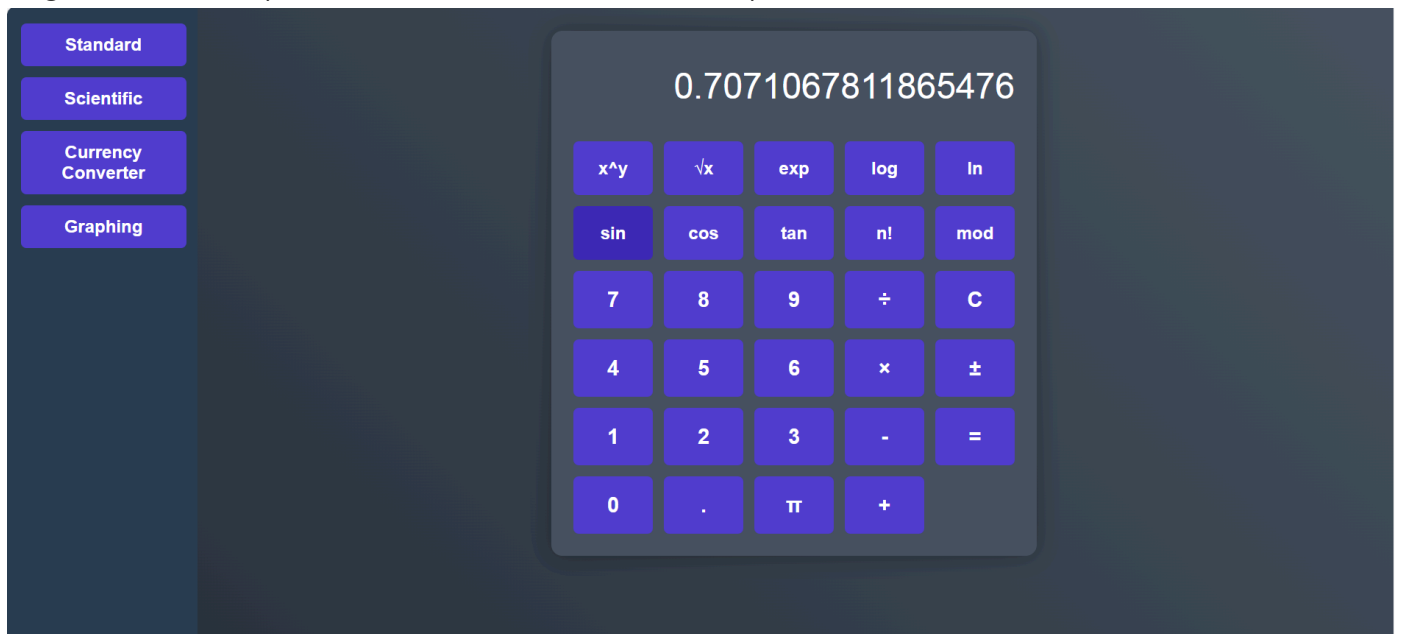
Mathematical Functions

Test advanced operations:

- Factorial: $5! = 120$



- Trigonometric: $\sin(./test-with-screenshots/assets/45^\circ) = 0.707$

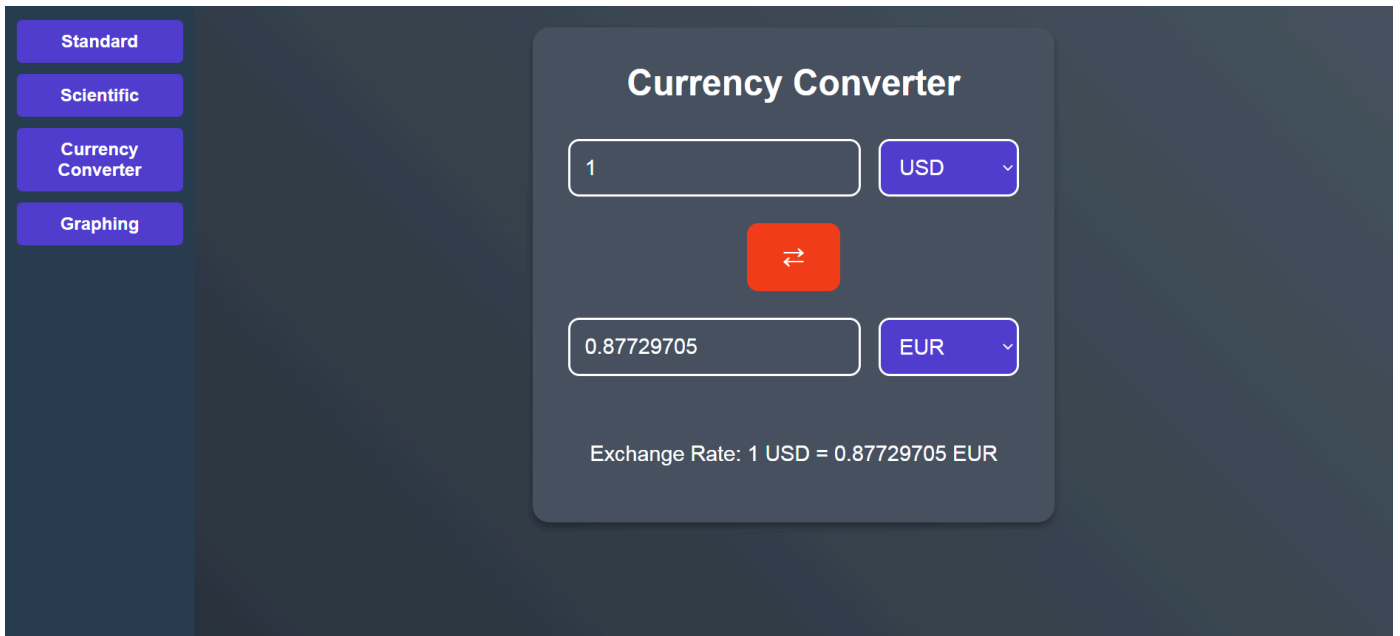


Part 3: Currency Converter Testing

Basic Conversion

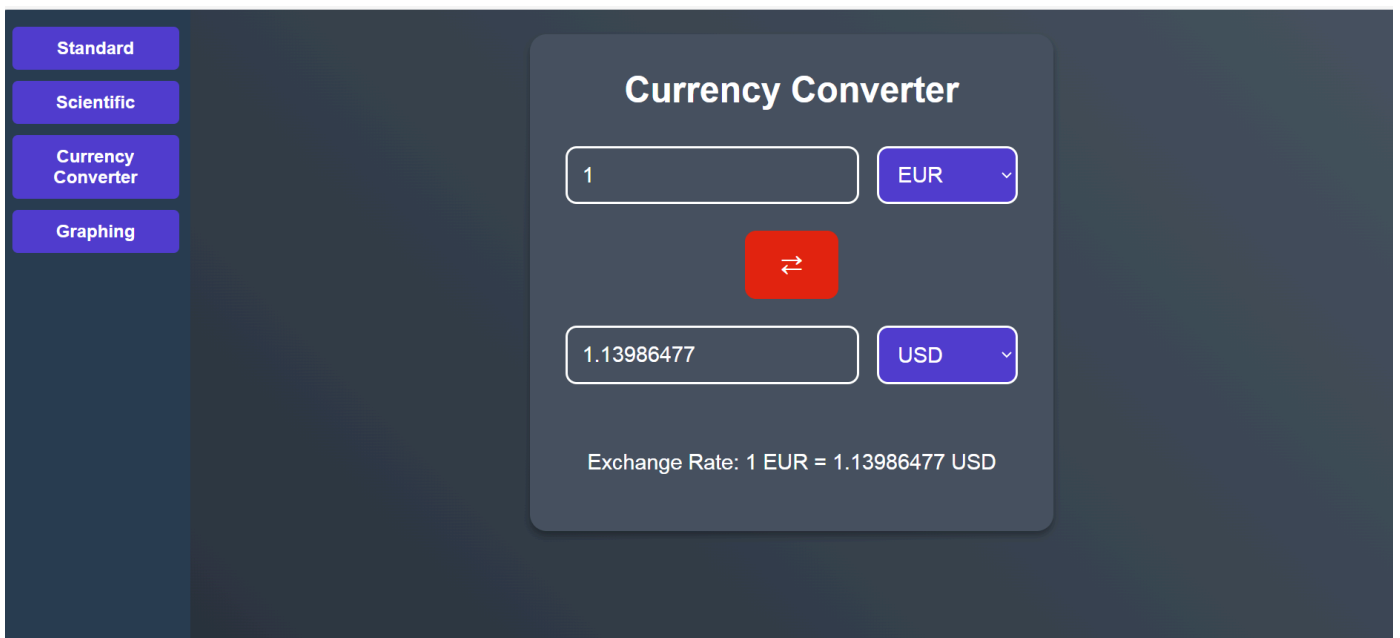
Test currency conversion:

- USD to EUR conversion



The screenshot shows a web application with a dark blue sidebar on the left containing four buttons: 'Standard', 'Scientific', 'Currency Converter', and 'Graphing'. The 'Currency Converter' button is highlighted. The main content area has a title 'Currency Converter' and two input fields. The top input field contains the value '1' and is followed by a dropdown menu showing 'USD'. Below these is a red button with a white double-headed arrow icon. The bottom input field contains the value '0.87729705' and is followed by a dropdown menu showing 'EUR'. At the bottom of the main area, the text 'Exchange Rate: 1 USD = 0.87729705 EUR' is displayed.

- Use swap currency button



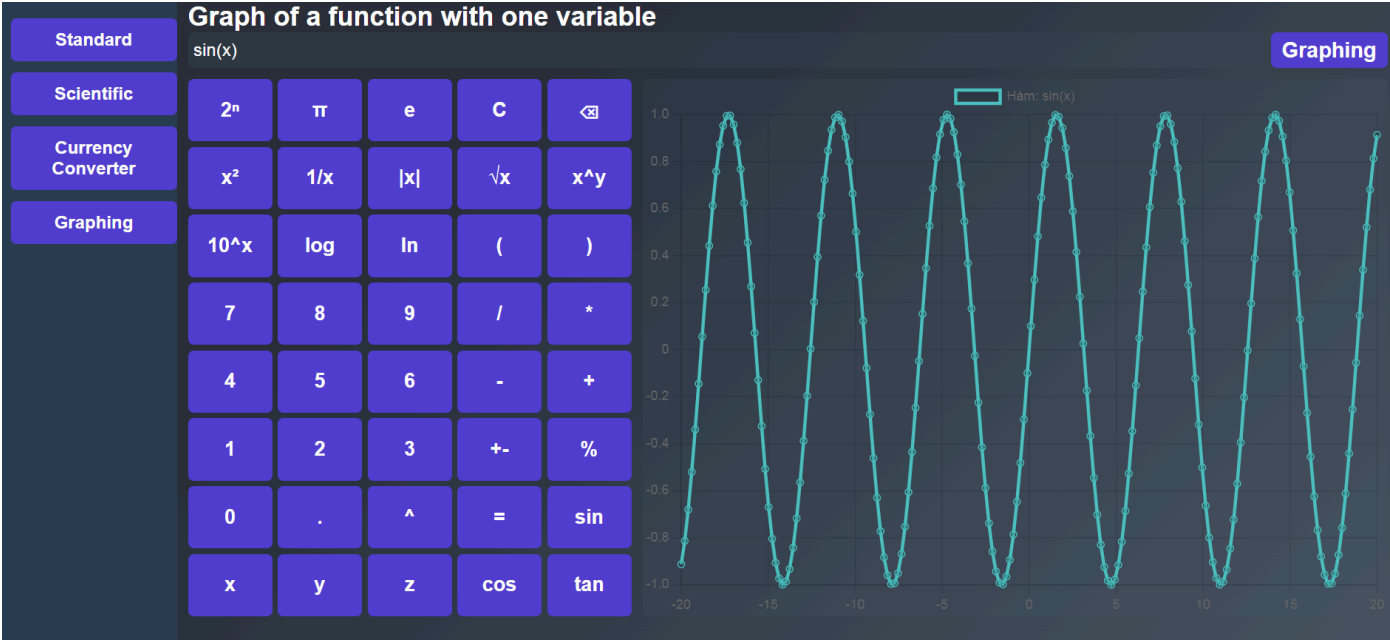
The screenshot shows the same web application as the previous one, but with the currencies swapped. The sidebar remains the same. The main content area now shows the top input field with the value '1' and a dropdown menu showing 'EUR'. The red swap button is still present. The bottom input field now contains the value '1.13986477' and a dropdown menu showing 'USD'. At the bottom, the text 'Exchange Rate: 1 EUR = 1.13986477 USD' is displayed.

Part 4: Graphing Calculator Testing

Function Plotting

Test various functions:

- Trigonometric: `sin(/test-with-screenshots/assets/x)`



- Complex: $x^2 + 2x + 1$

