CS628 Full-Stack Development – Web App
HOS01A: Experiencing AI-Powered Web App

Developed by Clark Ngo, 10/17/2024
Reviewed by Sam Chung, 12/24/2024
Reviewed by Yogesh Goel, 01/03/2024
Reviewed by Naveena Moddu, 04/11/2025

School of Technology & Computing (STC)
City University of Seattle (CityU)

Before You Start

- Screenshots may be different from your environment.
- The directory path shown in screenshots may be different from yours.
- There might be subtle discrepancies along with the steps. Please use your best judgment while going through this cookbook-style tutorial to complete each step.
- Some steps may not be explained in detail. If you are not sure what to do:
  1. Consult the resources from the course.
  2. If you cannot solve the problem after a few tries (usually 15 -30 minutes), ask a TA for help.

Readings and Examples:

- Visit the CS 628 Repository for Examples.
  - Select the related module.
  - Visit the README.md file.
  - Find examples for your practices.

Learning Outcomes

- Section 1: Accessing GitHub Codespaces
- Section 2: Installing and running AI-Powered Web App
- Section 3: Pushing your work to GitHub

Upload the following to your GitHub Repository generated from GitHub Classroom.

1. Upload the screenshot of your '01 Hello World' as '*01*_hello_world_firstname_lastname.png' by using your first and last name.
2. Upload the screenshot of your '02 Chat Response' as '*02*_chat_response_firstname_lastname.png' by using your first and last name.

Key Concepts

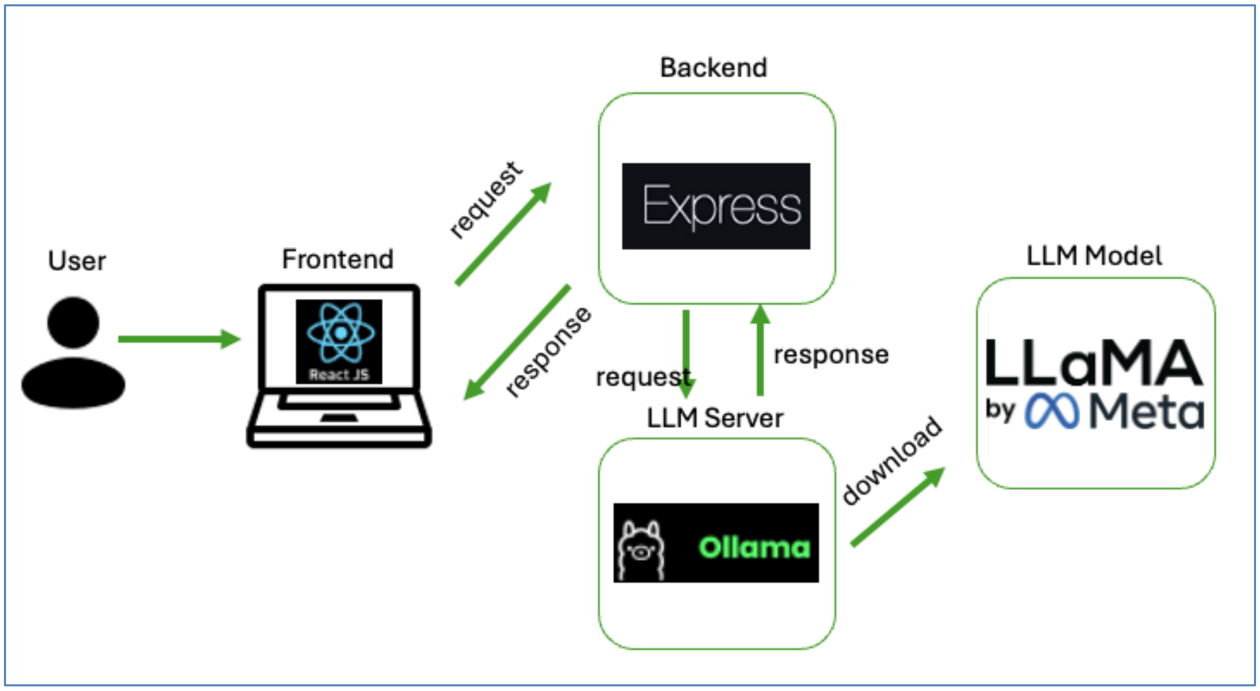


Figure 1. Deployment Diagram

The architecture above allows your app to offer a seamless chat experience with a front-end interface, a backend proxy, and an LLM backend for generating responses.

React

- Frontend Framework: ReactJS is a JavaScript library for building user interfaces, specifically for single-page applications (SPAs). It manages the view layer of the app.
- Component-Based Architecture: React uses components, which are reusable UI elements, to build the interface. This makes it easy to manage complex UIs.
- State Management: React manages the states internally within components or with libraries like Redux, Context API, or other state management tools.
- JSX Syntax: React uses JSX (JavaScript XML), a syntax that lets you write HTML within JavaScript code.

Express

- Backend Framework: Express is a minimal, fast, and flexible Node.js web application framework. It handles the server-side logic of your app.
- Middleware Integration: Express uses middleware functions to handle requests, responses, routing, authentication, etc., making it highly customizable.
- Routing: It manages HTTP request routing, enabling RESTful API design, which allows the frontend to communicate with the backend efficiently.
- Proxy for Services: In your case, Express likely acts as a proxy to communicate with Ollama and handle API interactions, such as sending and receiving data to/from Gemma 2.

Ollama

- LLM Hosting and Management: Ollama is a platform for hosting and managing large language models (LLMs), enabling developers to use models like Gemma 2 in their applications.
- API Endpoint: Ollama typically provides an API endpoint that you can interact with via HTTP requests, making it easier to integrate LLMs into existing apps.
- Use in Chatbots/Assistants: Ollama can be used to provide conversational capabilities or other AI-driven responses, which are utilized in your chat interface.

Gemma 2 (Large Language Model)

- Large Language Model: Gemma 2 is an advanced model for understanding and generating text, ideal for natural language processing tasks.
- Integration with Ollama: Hosted on Ollama, it receives prompts via Express and returns responses, creating a seamless backend-to-frontend flow.
- Conversational AI: Generates human-like responses, facilitating natural language conversations within the app.

Section 1: Accessing GitHub Codespaces

Follow the instructions here.

GitHub Codespaces are online cloud-based development environments that allow you to easily write, run, and debug your code. It is fully integrated with your GitHub repository and provides a seamless experience for developers.

Prerequisites:

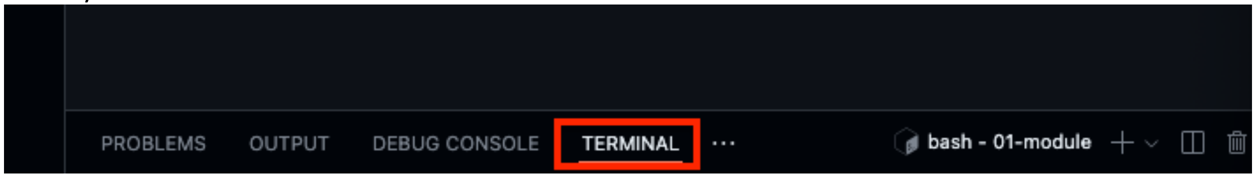
- Create your GitHub account

Section 2: Install and Run AI-Powered Web App

Note: The directory may differ. Make sure you are at the default directory (aka root directory of the project).

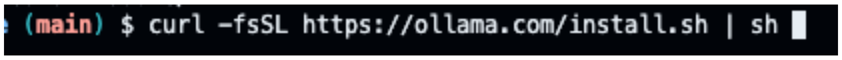
1) Run Ollama Server.

1. In your Terminal



2. Type the following to download Ollama.

```
curl -fsSL https://ollama.com/install.sh | sh
```



```
(main) $ curl -fsSL https://ollama.com/install.sh | sh
```

This curl (Client URL) command fetches the installation script from https://ollama.com/install.sh and runs it. Be cautious when running such commands, especially from unknown sources, as they can execute any code on your system. Always ensure the source is trustworthy.

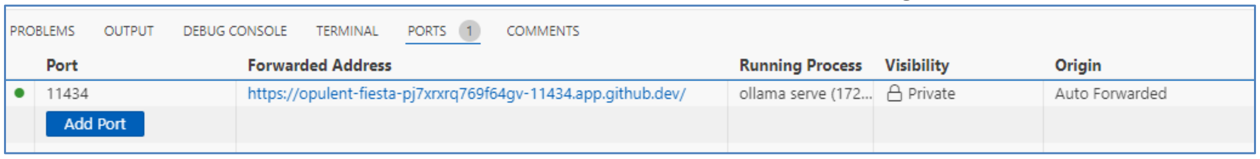
- curl: This is a command-line tool for transferring data with URLs.
- -f: This option tells curl to fail silently on server errors.
- -s: This makes curl operate in silent mode, meaning it won't show progress or error messages.
- -S: When used with -s, this option makes curl show an error message if it fails.
- -L: This option tells curl to follow any redirects.
- https://ollama.com/install.sh: This is the URL of the shell script you want to download.
- | sh: This pipes the downloaded script to the sh command, which executes it.

3. After the download has finished, type the following in the terminal.

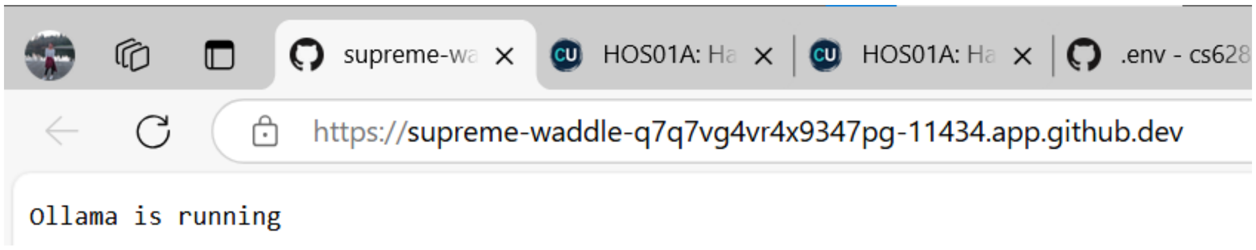
```
ollama serve
```

The command is used to start the Ollama service on your local system. This command makes Ollama accessible, allowing you to interact with it and run various language models

4. Visit the Ports tab and access the URL. You can see the Ollama is running.



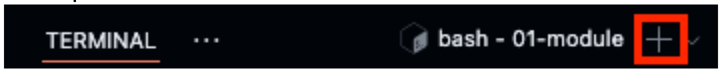
Port	Forwarded Address	Running Process	Visibility	Origin
11434	https://opulent-fiesta-pj7xrxrq769f64gv-11434.app.github.dev/	ollama serve (172...	Private	Auto Forwarded


supreme-wa
HOS01A: Ha
HOS01A: Ha
.env - cs628
https://supreme-waddle-q7q7vg4vr4x9347pg-11434.app.github.dev

```
Ollama is running
```

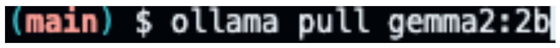
2) Pull and Chat with a Large Language Model (LLM) Conversational AI.

1. Open New Terminal



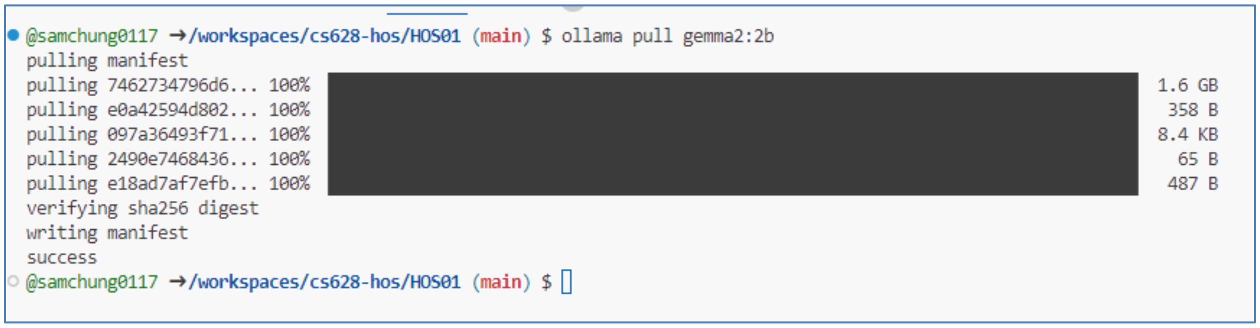
2. Type the following in the terminal

```
ollama pull gemma2:2b
```



```
(main) $ ollama pull gemma2:2b
```

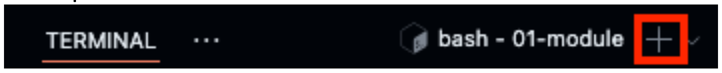
The command is used to download the Gemma 2 model with 2 billion parameters from the Ollama library. For example,



```
@samchung0117 →/workspaces/cs628-hos/HOS01 (main) $ ollama pull gemma2:2b
pulling manifest
pulling 7462734796d6... 100%                                              1.6 GB
pulling e0a42594d802... 100%                                               358 B
pulling 097a36493f71... 100%                                              8.4 KB
pulling 2490e7468436... 100%                                                65 B
pulling e18ad7af7efb... 100%                                               487 B
verifying sha256 digest
writing manifest
success
@samchung0117 →/workspaces/cs628-hos/HOS01 (main) $
```

3) Run ExpressJS Backend Server.

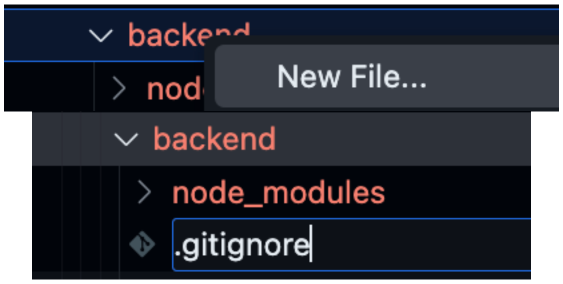
1. Open New Terminal



2. Type the following in terminal.

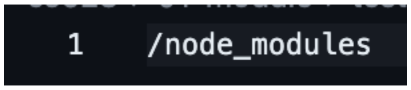
```
mkdir backend
cd backend
```

Before moving add, let's add a **.gitignore** file as we usually don't add **node_modules** folder as part of version control.



Why? The file size can get very large and it's not optimal to host it over the internet. We let other users or systems install their own **node_modules** folder as defined in **package.json**, which will be created later with npm init and npm install.

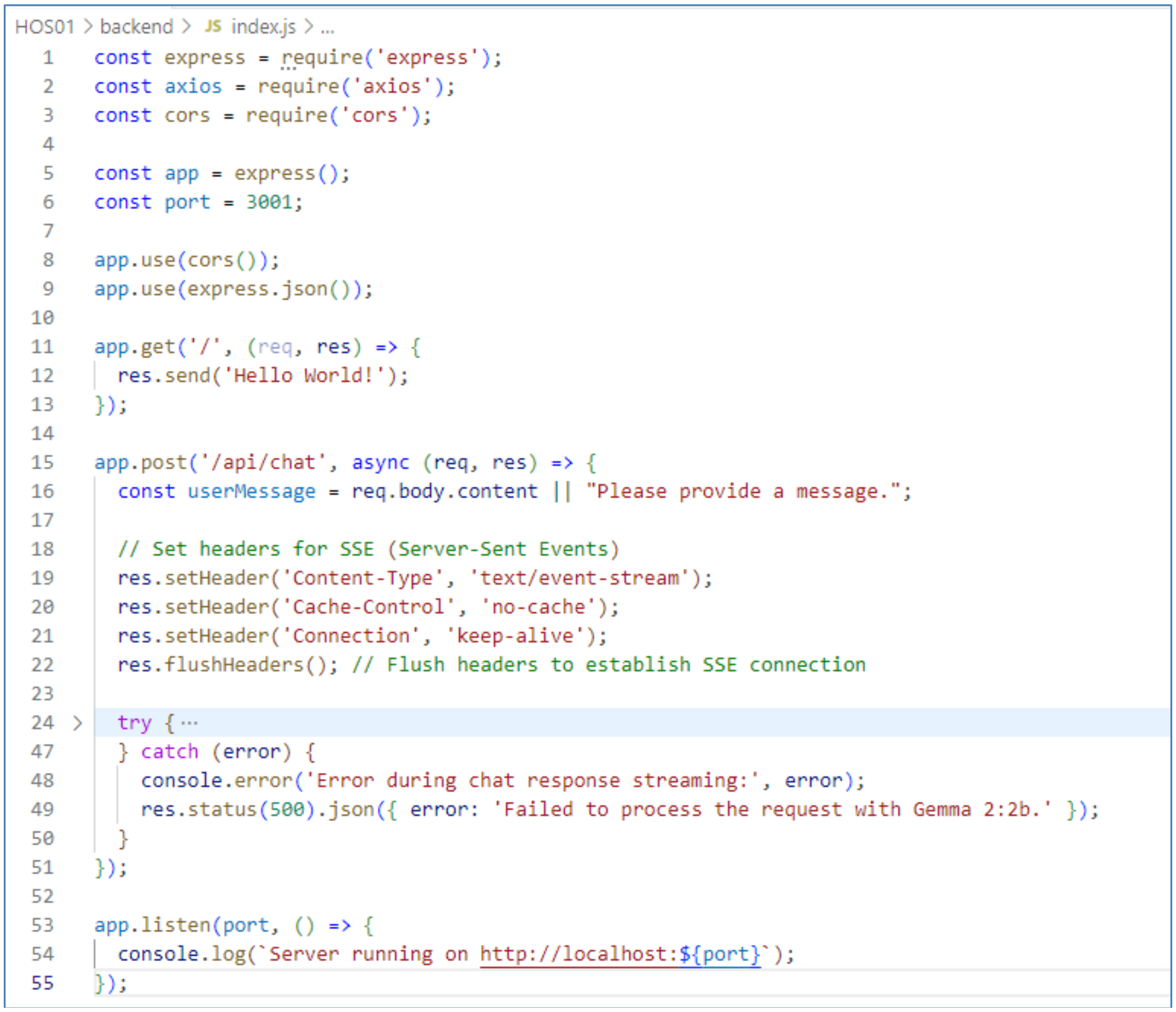
Add in the first line:
/node_modules



```
/node_modules
```

```
npm init -y
npm install axios cors express
```

3. Create the “index.js” file. You can find the source code from the CS628 Repository for Examples.

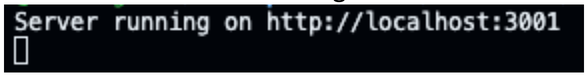


HOS01 > backend > JS index.js > ...

```javascript
const express = require('express');
const axios = require('axios');
const cors = require('cors');

const app = express();
const port = 3001;

app.use(cors());
app.use(express.json());

app.get('/', (req, res) => {
  res.send('Hello World!');
});

app.post('/api/chat', async (req, res) => {
  const userMessage = req.body.content || "Please provide a message.";

  // Set headers for SSE (Server-Sent Events)
  res.setHeader('Content-Type', 'text/event-stream');
  res.setHeader('Cache-Control', 'no-cache');
  res.setHeader('Connection', 'keep-alive');
  res.flushHeaders(); // Flush headers to establish SSE connection

  try {…
  } catch (error) {
    console.error('Error during chat response streaming:', error);
    res.status(500).json({ error: 'Failed to process the request with Gemma 2:2b.' });
  }
});

app.listen(port, () => {
  console.log(`Server running on http://localhost:${port}`);
});
```

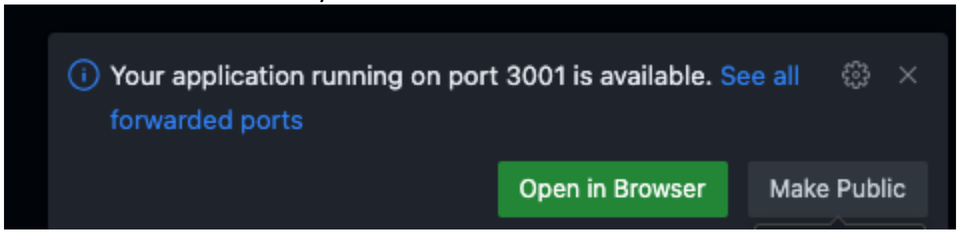
4. Execute the backend.
   node index.js

You should see the following in the terminal:



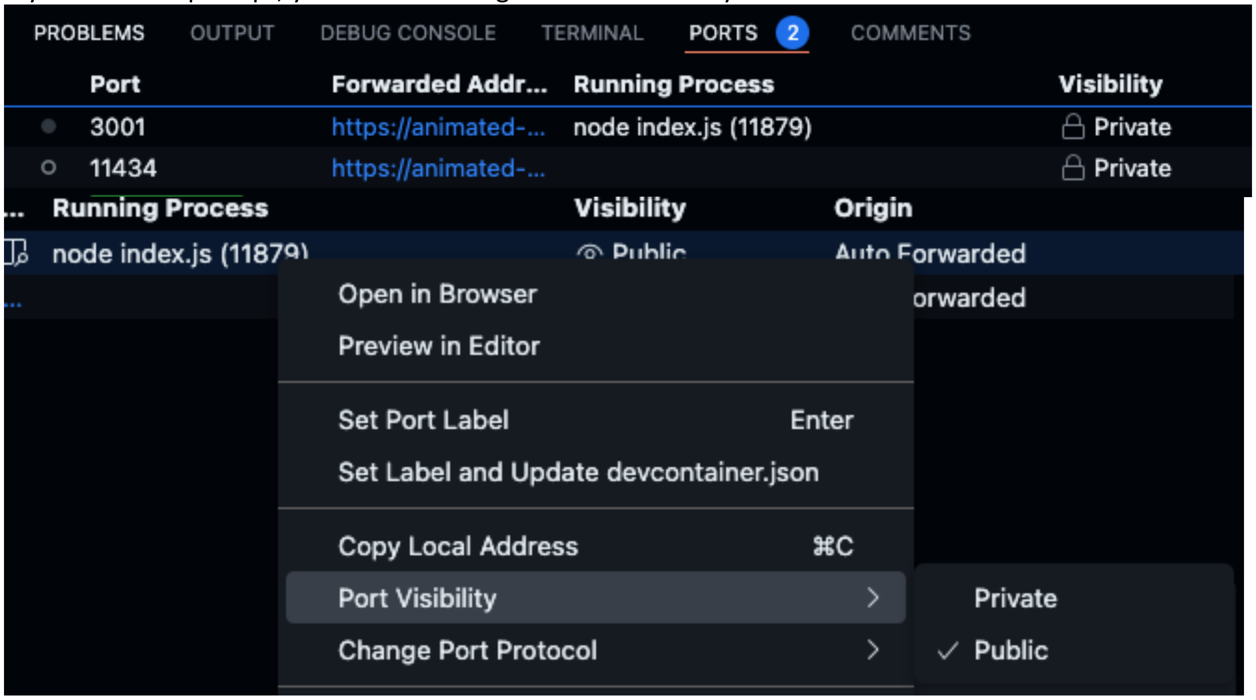
```
Server running on http://localhost:3001
```

Make the Port Visibility `Public`



Your application running on port 3001 is available. See all forwarded ports

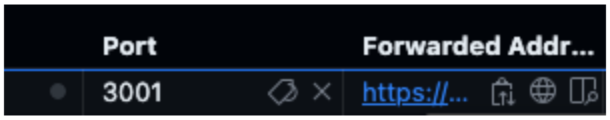
If you miss this prompt, you can still change the Port Visibility in the Ports tab.



Port	Forwarded Addr...	Running Process	Visibility
3001	https://animated-...	node index.js (11879)	Private
11434	https://animated-...		Private

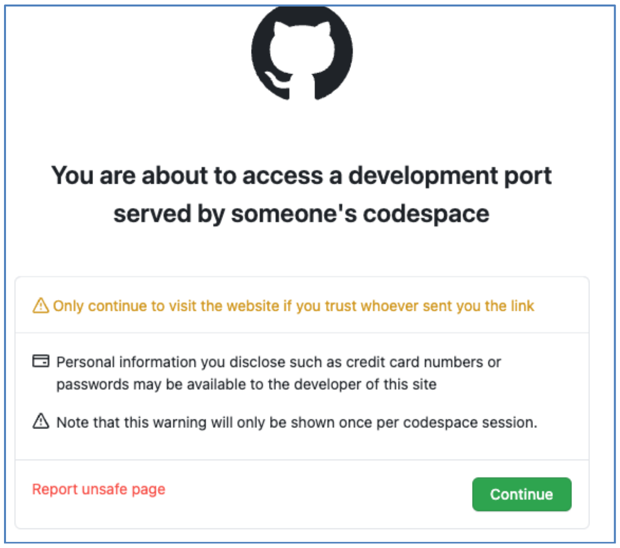
Running Process	Visibility	Origin
node index.js (11879)	Public	Auto Forwarded
		orwarded

5. Access the Forwarded Address in the Ports tab by hovering in the Port 3001 row and clicking the globe icon .



Port	Forwarded Addr...
3001	https://...

If you see this prompt, click Continue.



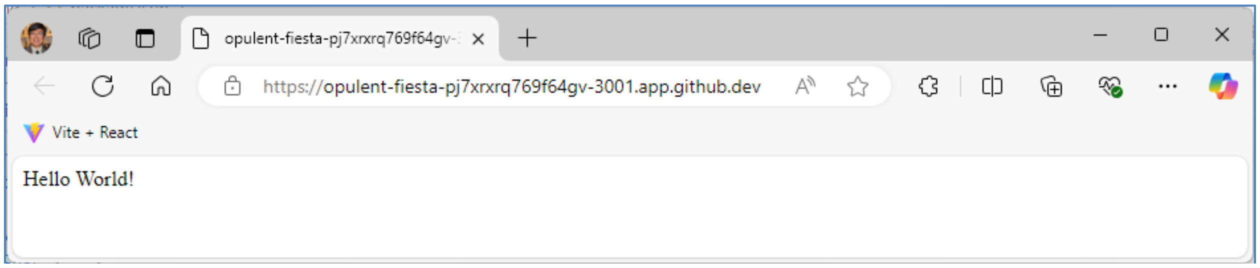
You are about to access a development port served by someone's codespace

Only continue to visit the website if you trust whoever sent you the link

Personal information you disclose such as credit card numbers or passwords may be available to the developer of this site

Note that this warning will only be shown once per codespace session.

You should expect to see a Hello World text. For example,

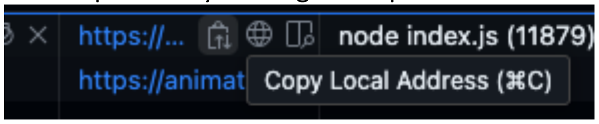


opulent-fiesta-pj7xrxrq769f64gv-
https://opulent-fiesta-pj7xrxrq769f64gv-3001.app.github.dev
Vite + React

Hello World!

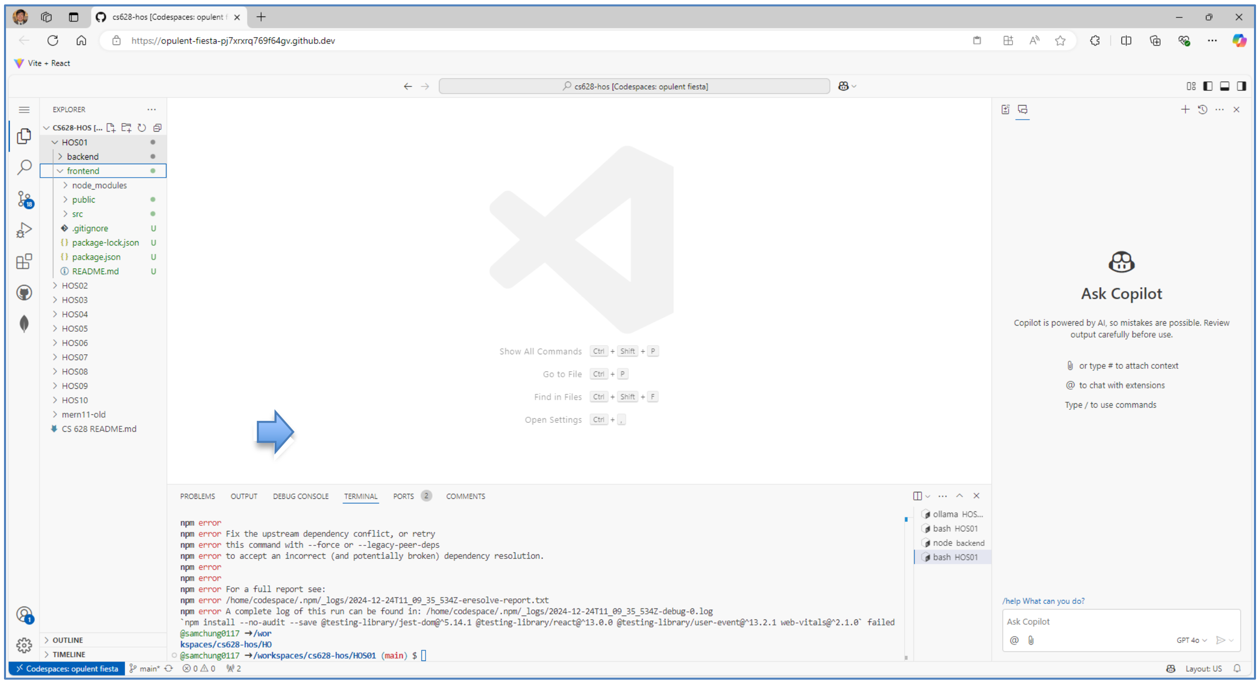
Take a screenshot of your '01 Hello World' as '*01*_hello_world_firstname_lastname.png' by using your first and last name. The URL is different from mine.

6. Copy the forwarded address ExpressJS by clicking the clipboard icon .



4) Run React.js Frontend Server.

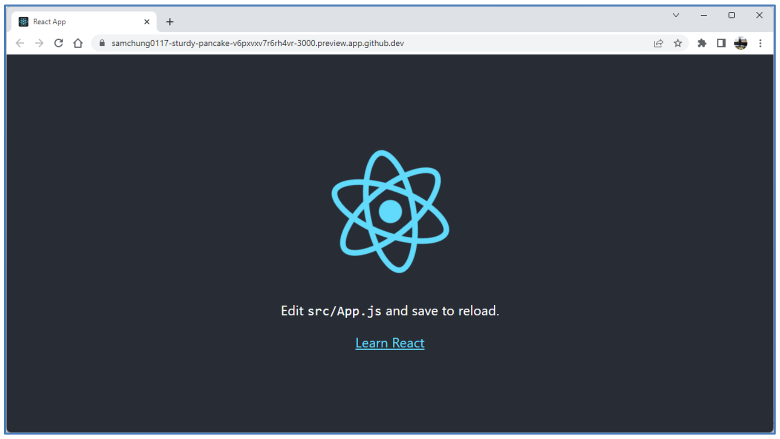
1. Open another terminal.
2. Suppose you are under the current HOS directory, HOS01.
   (Caution: The screen below shows the instructor's GitHub Codespaces.)
3. Run the command "**npx create-react-app frontend**" to create a React app, **frontend**, under the HOS01.
   You can see a directory called "frontend" was created by using the CRA (Create React App).
   npx create-react-app frontend



```
npm error
npm error Fix the upstream dependency conflict, or retry
npm error this command with --force or --legacy-peer-deps
npm error to accept an incorrect (and potentially broken) dependency resolution.
npm error
npm error
npm error For a full report see:
npm error /home/codespace/.npm/_logs/2024-12-24T11_09_35_534Z-eresolve-report.txt
npm error A complete log of this run can be found in: /home/codespace/.npm/_logs/2024-12-24T11_09_35_534Z-debug-0.log
`npm install --no-audit --save @testing-library/jest-dom@^5.14.1 @testing-library/react@^13.0.0 @testing-library/user-event@^13.2.1 web-vitals@^2.1.0` failed
@samchung0117 ➜/wor
kspaces/cs628-hos/HO
@samchung0117 ➜/workspaces/cs628-hos/HOS01 (main) $
```

4. Change the current directory to the newly created project.
   cd frontend

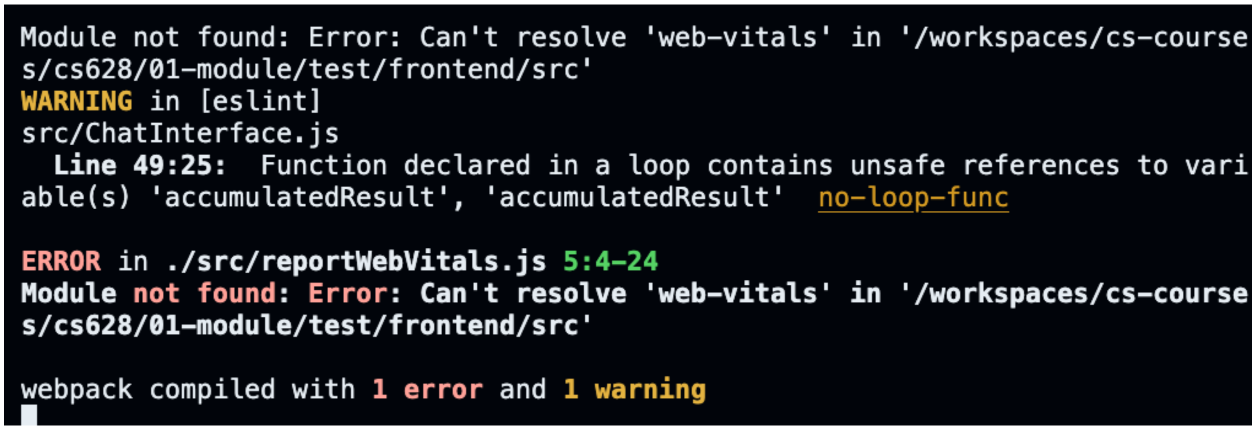
5. Run the command **npm start** to start the development server.
   npm start
   If the web app works, you can see the web app in a web browser.


React App
samchung0117-sturdy-pancake-v6pxvxv7r6rh4vr-3000.preview.app.github.dev

Edit `src/App.js` and save to reload.

Learn React

If you see an error about **web-vitals:**



```
Module not found: Error: Can't resolve 'web-vitals' in '/workspaces/cs-course
s/cs628/01-module/test/frontend/src'
WARNING in [eslint]
src/ChatInterface.js
  Line 49:25:  Function declared in a loop contains unsafe references to vari
able(s) 'accumulatedResult', 'accumulatedResult'  no-loop-func

ERROR in ./src/reportWebVitals.js 5:4-24
Module not found: Error: Can't resolve 'web-vitals' in '/workspaces/cs-course
s/cs628/01-module/test/frontend/src'

webpack compiled with 1 error and 1 warning
```

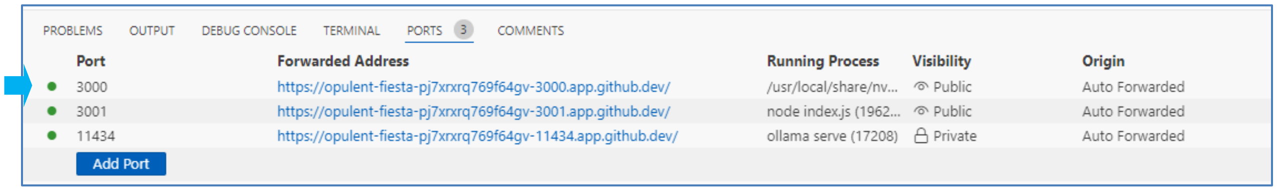
Please cancel the running terminal with **Ctrl + C** then type and execute:

npm install web-vitals

Then try running again

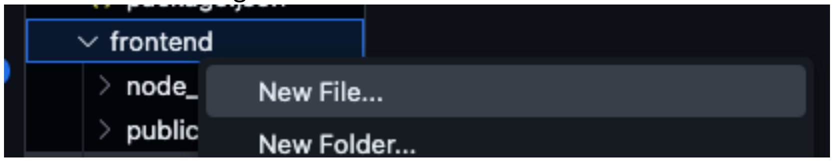
npm start

6. You can also visit the PORTS panel below the editor region for output or debug information, errors and warnings, or an integrated terminal.
   You can select the "Open in Browser" icon.
   Click it to open the client web application at the port number 3000.

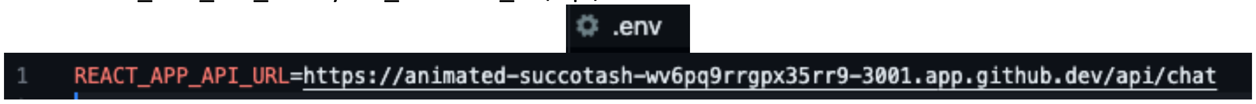


	Port	Forwarded Address	Running Process	Visibility	Origin
●	3000	https://opulent-fiesta-pj7xrxrq769f64gv-3000.app.github.dev/	/usr/local/share/nv...	Public	Auto Forwarded
●	3001	https://opulent-fiesta-pj7xrxrq769f64gv-3001.app.github.dev/	node index.js (1962...	Public	Auto Forwarded
●	11434	https://opulent-fiesta-pj7xrxrq769f64gv-11434.app.github.dev/	ollama serve (17208)	Private	Auto Forwarded

7. Go to your frontend folder and right-click to create a new file.

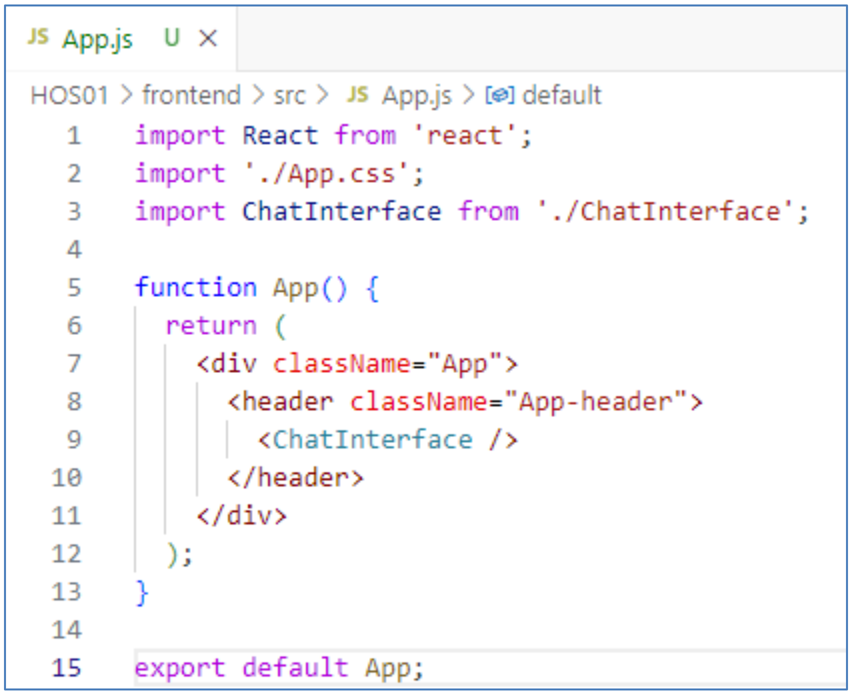


8. Create `.env` file and paste **the URL of your backend** after `REACT_APP_API_URL=your_backedn_url/api/chat`



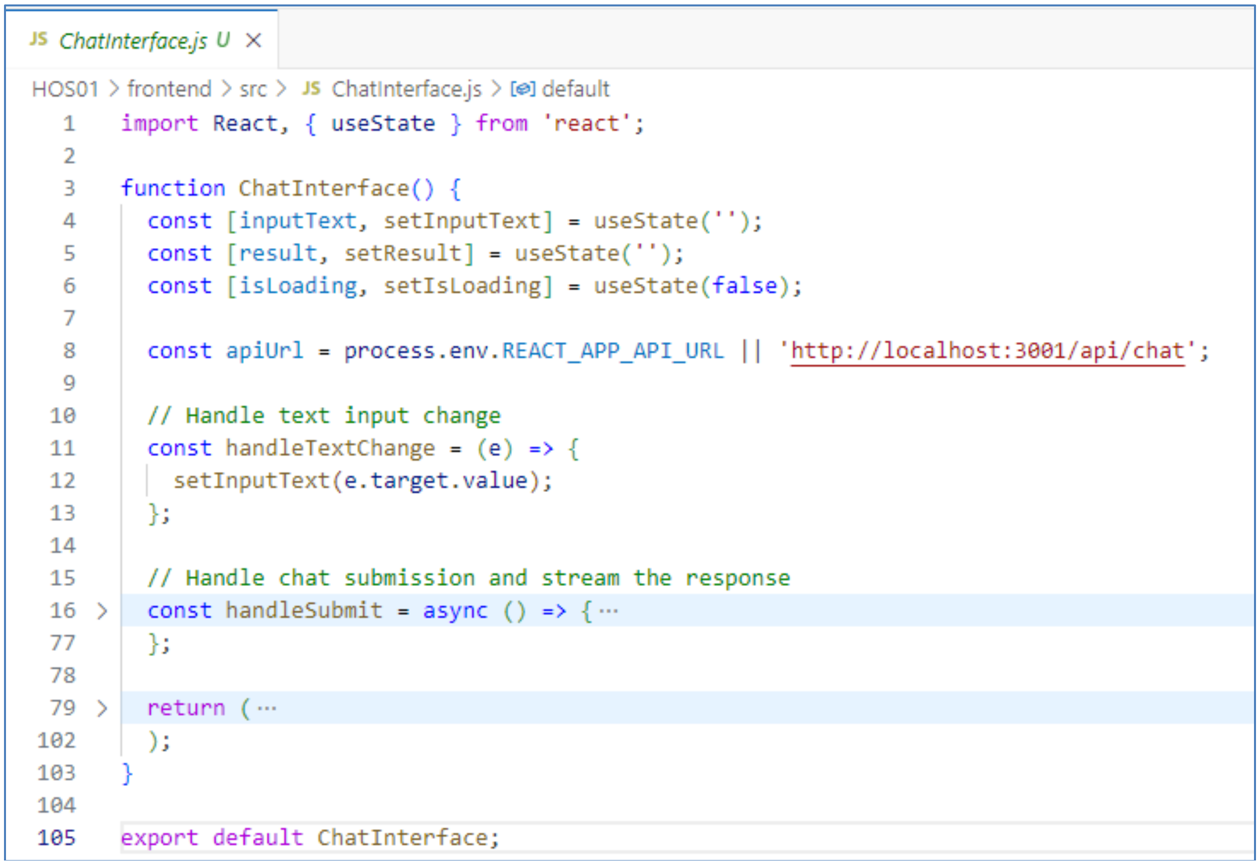
```
.env
REACT_APP_API_URL=https://animated-succotash-wv6pq9rrgpx35rr9-3001.app.github.dev/api/chat
```

9. Find the "App.js" file under the "src" directory. Replace its content. You can find the source code from the [CS628 Repository for Examples.]()



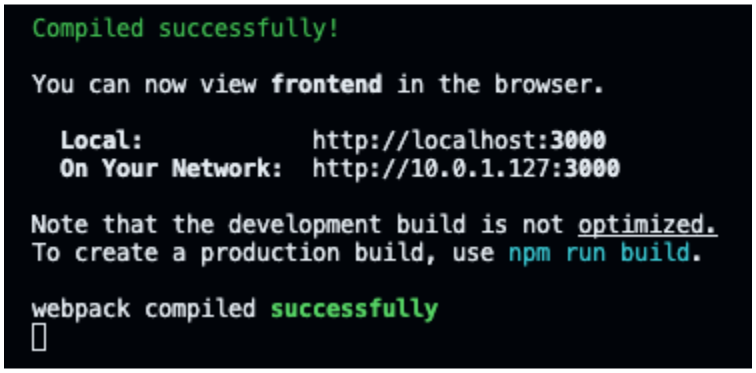
```
App.js  U ×
HOS01 > frontend > src > App.js > default
import React from 'react';
import './App.css';
import ChatInterface from './ChatInterface';

function App() {
  return (
    <div className="App">
      <header className="App-header">
        <ChatInterface />
      </header>
    </div>
  );
}

export default App;
```

10. Create the "ChatInterface.js" component under the "src" directory. Replace its content. You can find the source code from the CS628 Repository for Examples.



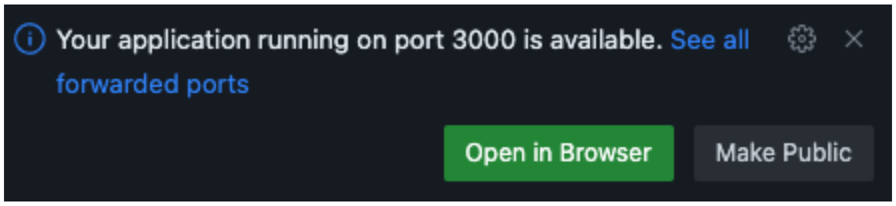
```
import React, { useState } from 'react';

function ChatInterface() {
  const [inputText, setInputText] = useState('');
  const [result, setResult] = useState('');
  const [isLoading, setIsLoading] = useState(false);

  const apiUrl = process.env.REACT_APP_API_URL || 'http://localhost:3001/api/chat';

  // Handle text input change
  const handleTextChange = (e) => {
    setInputText(e.target.value);
  };

  // Handle chat submission and stream the response
  const handleSubmit = async () => {…
  };

  return (…
  );
}

export default ChatInterface;
```

11. You should expect this output in your terminal.



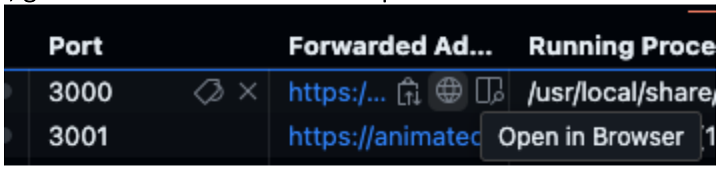
```
Compiled successfully!

You can now view frontend in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://10.0.1.127:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```

Click Open in Browser to access the front-end interface.



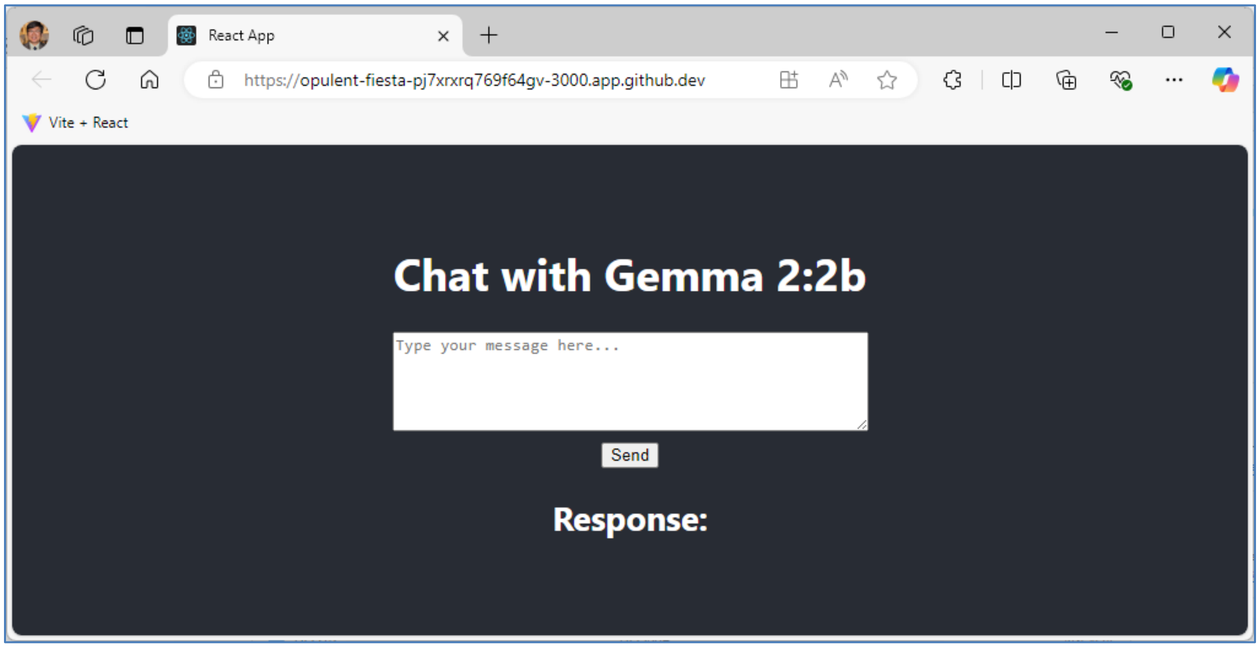
Your application running on port 3000 is available. See all forwarded ports

If you miss this prompt, go to the Ports tab and click Open in Browser for Port 3000.



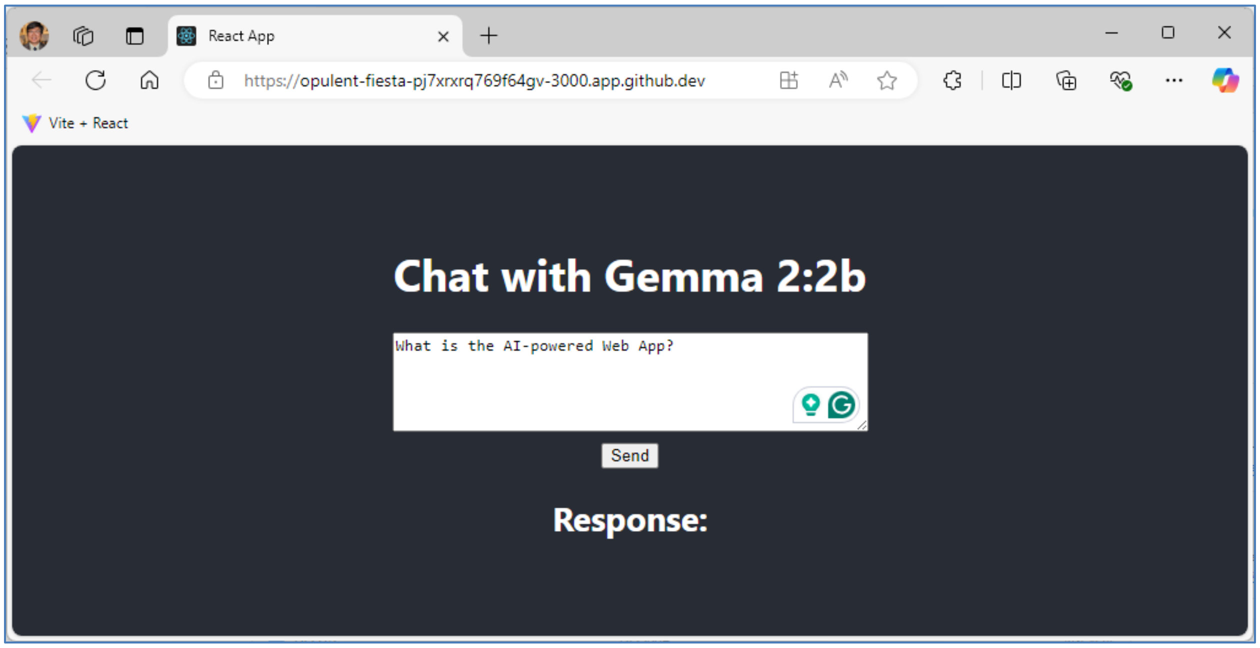
Port	Forwarded Ad...	Running Proce
3000	https:/...	/usr/local/share/
3001	https://animatec	1

12. You should see this screen. If not, refresh the web browser for the frontend. You can see the updated screen.



React App
https://opulent-fiesta-pj7xrxrq769f64gv-3000.app.github.dev
Vite + React
Chat with Gemma 2:2b
Type your message here...
Send
Response:

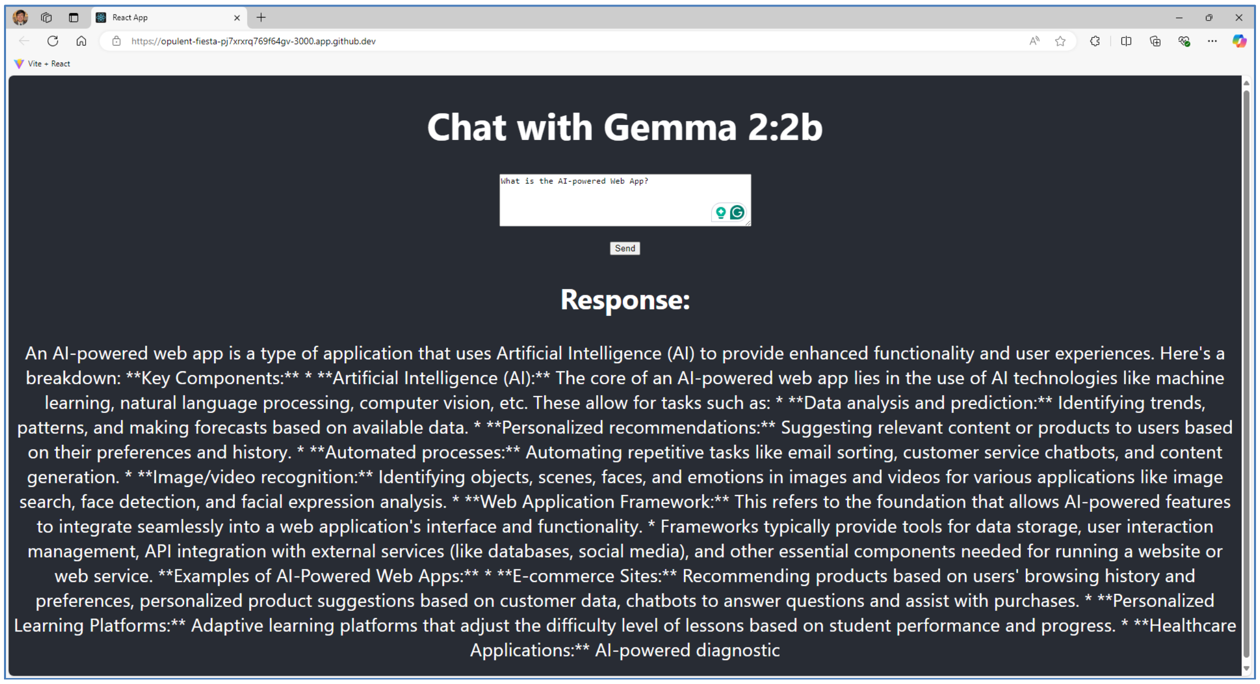
13. In the textbox, type the following and click Send:



Chat with Gemma 2:2b

```
What is the AI-powered Web App?
```

Response:



Chat with Gemma 2:2b

```
What is the AI-powered Web App?
```

Response:

An AI-powered web app is a type of application that uses Artificial Intelligence (AI) to provide enhanced functionality and user experiences. Here's a breakdown: **Key Components:** * **Artificial Intelligence (AI):** The core of an AI-powered web app lies in the use of AI technologies like machine learning, natural language processing, computer vision, etc. These allow for tasks such as: * **Data analysis and prediction:** Identifying trends, patterns, and making forecasts based on available data. * **Personalized recommendations:** Suggesting relevant content or products to users based on their preferences and history. * **Automated processes:** Automating repetitive tasks like email sorting, customer service chatbots, and content generation. * **Image/video recognition:** Identifying objects, scenes, faces, and emotions in images and videos for various applications like image search, face detection, and facial expression analysis. * **Web Application Framework:** This refers to the foundation that allows AI-powered features to integrate seamlessly into a web application's interface and functionality. * Frameworks typically provide tools for data storage, user interaction management, API integration with external services (like databases, social media), and other essential components needed for running a website or web service. **Examples of AI-Powered Web Apps:** * **E-commerce Sites:** Recommending products based on users' browsing history and preferences, personalized product suggestions based on customer data, chatbots to answer questions and assist with purchases. * **Personalized Learning Platforms:** Adaptive learning platforms that adjust the difficulty level of lessons based on student performance and progress. * **Healthcare Applications:** AI-powered diagnostic

Take a screenshot of your '02 Chat Response' as '02_chat_response_firstname_lastname.png' by using your first and last name.

Section 3: Pushing your work to GitHub

How to submit your work in GitHub