**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

Reviewed by Harrison Le, 07/07/2025

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](https://github.com/samchung0117/cs628-examples).
  + Select the related module.
  + Visit the README.md file.
  + Find examples for your practices.

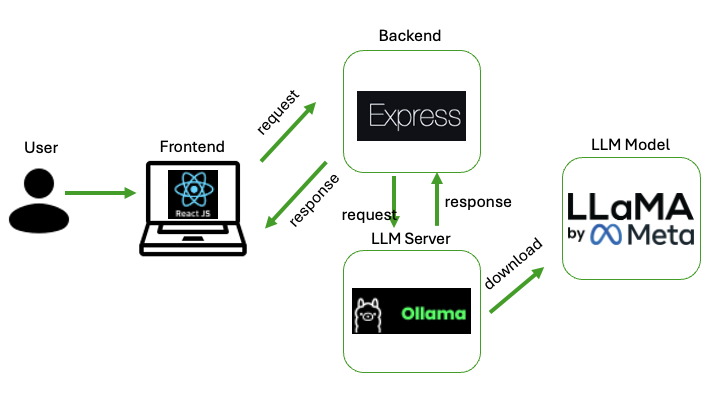
**Learning Outcomes**

* 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](https://cityuseattle.github.io/docs/git/github_codepsace/).

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](https://cityuseattle.github.io/docs/git/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



* + - 1. Type the following to download Ollama.

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



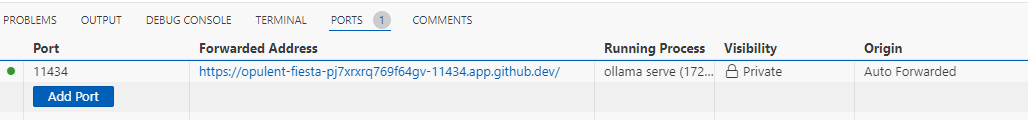
This curl (Client URL) command fetches the installation script from https://ollama.com/install.sh and runs it. Be cautious when running such commands, especially from unknown sources, as they can execute any code on your system. Always ensure the source is trustworthy.

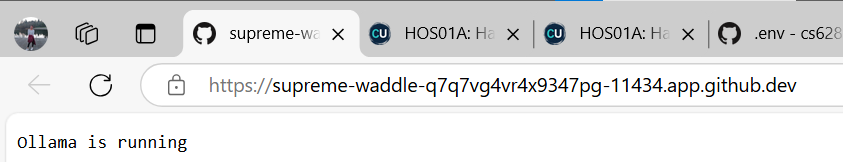
* curl: This is a command-line tool for transferring data with URLs.
* -f: This option tells curl to fail silently on server errors.
* -s: This makes curl operate in silent mode, meaning it won't show progress or error messages.
* -S: When used with -s, this option makes curl show an error message if 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.
  + - 1. 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

* + - 1. Visit the Ports tab and access the URL. You can see the Ollama is running.





1. **Pull and Chat with a Large Language Model (LLM) Conversational AI.**
2. Open New Terminal



1. Type the following in the terminal

|  |
| --- |
| ollama pull gemma2:2b |



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

A screen shot of a computer

Description automatically generated

1. **Run ExpressJS Backend Server.**
2. Open New Terminal

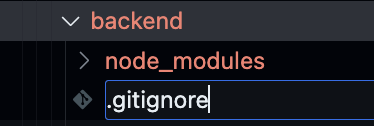


1. Type the following in terminal.

|  |
| --- |
| 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**



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

1. Create the “index.js” file. You can find the source code from the [CS628 Repository for Examples](https://github.com/samchung0117/cs628-examples).

A screenshot of a computer program

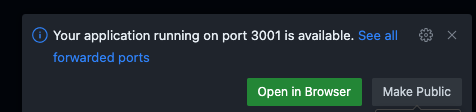
Description automatically generated

1. Execute the backend.  
   **node index.js**

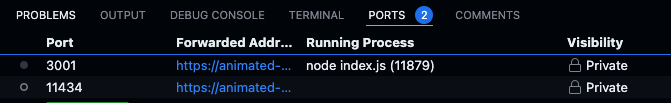
You should see the following in the terminal:

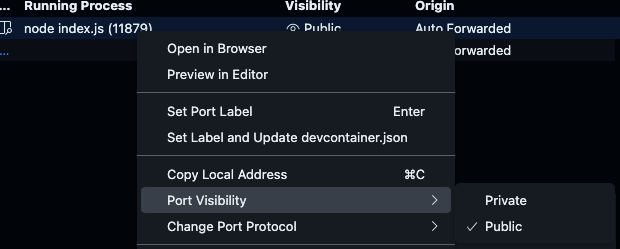


Make the Port Visibility `Public`



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

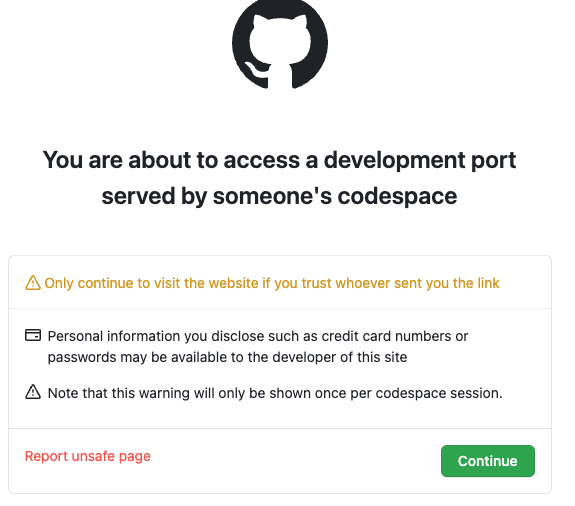




1. Access the Forwarded Address in the Ports tab by hovering in the Port 3001 row and clicking the globe icon .



If you see this prompt, click Continue.



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

A screenshot of a computer

Description automatically generated

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.

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



1. **Run React.js Frontend Server.**
2. Open another terminal.
3. Suppose you are under the current HOS directory, HOS01.  
   (Caution: The screen below shows the instructor’s GitHub Codespaces.)
4. 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**

A screenshot of a computer

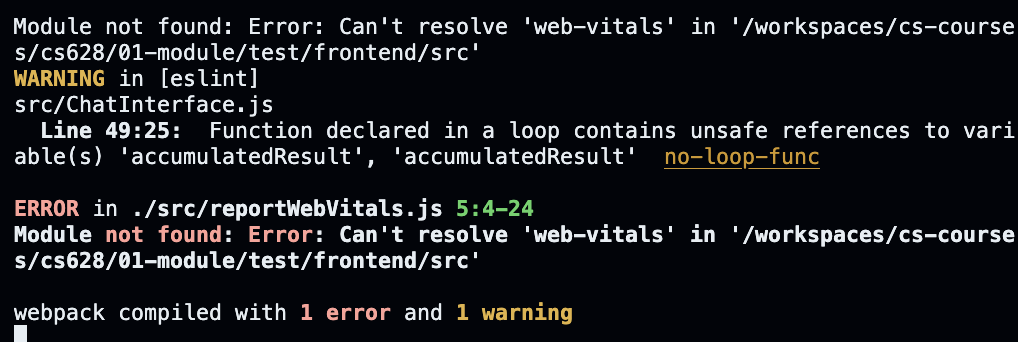
Description automatically generated

1. Change the current directory to the newly created project.  
   **cd frontend**
2. 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.

A screen shot of a computer

Description automatically generated

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



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

**npm install web-vitals**

Then try running again

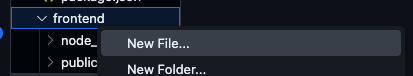
**npm start**

1. 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.

A screenshot of a computer

Description automatically generated

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



1. Create `.env` file and paste **the URL of your backend** after `REACT\_APP\_API\_URL=your\_backedn\_url/api/chat`





1. Find the “App.js” file under the “src” directory. Replace its content. You can find the source code from the [CS628 Repository for Examples](https://github.com/samchung0117/cs628-examples).

A screenshot of a computer code

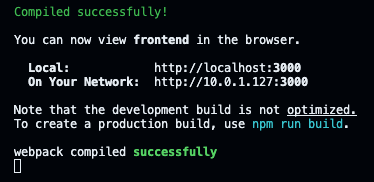
Description automatically generated

1. Create the “ChatInterface.js” component under the “src” directory. Replace its content. You can find the source code from the [CS628 Repository for Examples](https://github.com/samchung0117/cs628-examples).

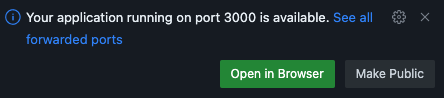
A screenshot of a computer program

Description automatically generated

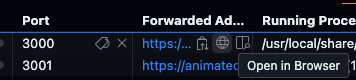
1. You should expect this output in your terminal.



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



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



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

A screenshot of a computer

Description automatically generated

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

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

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](https://cityuseattle.github.io/docs/git/github_upload_files/)