290l Assignment 3 Due Nov 10 11:59 pm

Programming Language

The following instructions and provided template code are based on **Python**, the **FastAPI** framework, and deployment on the **Render** platform.

This is just one example of how to deploy the code. You may use any programming language, an equivalent framework, and deploy to your preferred platform, as long as your API meets the assignment requirements.

Setup

Git

Git is a distributed version control system. Follow the installation instructions at: https://git-scm.com/install/

Github

GitHub is a developer platform for managing and sharing code. It uses Git to provide distributed version control.

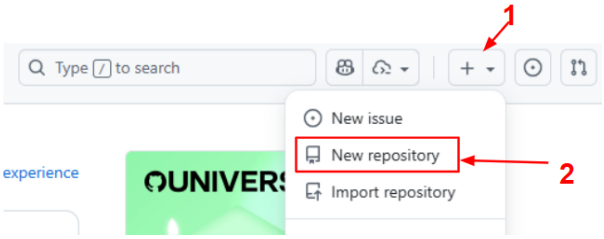
We will need this to deploy our code to the Render platform.

1. **Create an account**

   For this assignment, we will need a GitHub account to upload and deploy your code. Sign up at: https://github.com/signup

2. **Create a repository (repo) for this assignment**

   a. In the upper-right corner of any page, select **+**, then click **New repository**.



   b. Name the repository **290l-Assignment3**

c. Keep the default settings, and click **Create repository**.

3. Create a local repository

The previous step created a **remote repository** (stored in the cloud).

Now, we'll create a **local repository** on your computer and link it to the **remote repository**.

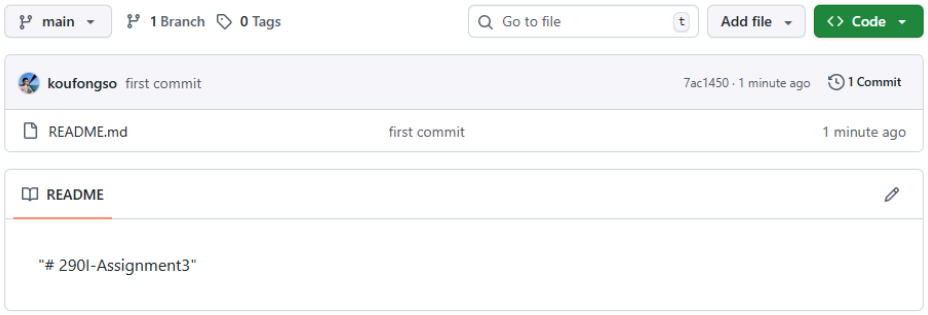
This allows you to push (upload) and pull (download) changes between them — similar to saving to or retrieving from the cloud.

a. Create a folder named 290I-Assignment3 (or any name you prefer). This will be the **root folder** of your project.
b. Open a terminal and navigate to this folder.
c. Use the commands shown on your GitHub repo page (they should look something like this):

```
None
echo "# test" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/<your_username>/<your repo name>.git
git push -u origin main
```

```
C:\Users\koufo\Desktop\290I-Assignment3>echo "# 290I-Assignment3" >> README.md

C:\Users\koufo\Desktop\290I-Assignment3>git init
Initialized empty Git repository in C:/Users/koufo/Desktop/290I-Assignment3/.git/

C:\Users\koufo\Desktop\290I-Assignment3>git add README.md

C:\Users\koufo\Desktop\290I-Assignment3>git commit -m "first commit"
[master (root-commit) 7ac1450] first commit
 1 file changed, 1 insertion(+)
 create mode 100644 README.md

C:\Users\koufo\Desktop\290I-Assignment3>git branch -M main

C:\Users\koufo\Desktop\290I-Assignment3>git remote add origin https://github.com/koufongso/290I-Assignment3.git

C:\Users\koufo\Desktop\290I-Assignment3>git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 231 bytes | 231.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/koufongso/290I-Assignment3.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

After executing the commands, your local repository should mirror your GitHub repo when you refresh the GitHub page.



d. Copy the provided template code to the root folder. In the terminal (inside the root folder), run:

```
None
git status
```

This will show any **untracked files** (new files not yet committed).

```
C:\Users\koufo\Desktop\290I-Assignment3>git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        dijkstra.py
        graph.py
        node.py
        priority_queue.py
        requirement.txt
        server.py
        server_local.py
        utils.py

nothing added to commit but untracked files present (use "git add" to track)
```

e. Then, add, commit, and push your files:

```
None
git add .
git commit -m "added template files"
```

```
git push origin main
```

```
C:\Users\koufo\Desktop\290I-Assignment3>git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        dijkstra.py
        graph.py
        node.py
        priority_queue.py
        requirement.txt
        server.py
        server_local.py
        utils.py

nothing added to commit but untracked files present (use "git add" to track)

C:\Users\koufo\Desktop\290I-Assignment3>git add .

C:\Users\koufo\Desktop\290I-Assignment3>git commit -m "added template files"
[main 3ae3dd1] added template files
 8 files changed, 326 insertions(+)
 create mode 100644 dijkstra.py
 create mode 100644 graph.py
 create mode 100644 node.py
 create mode 100644 priority_queue.py
 create mode 100644 requirement.txt
 create mode 100644 server.py
 create mode 100644 server_local.py
 create mode 100644 utils.py

C:\Users\koufo\Desktop\290I-Assignment3>git push origin main
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 8 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (10/10), 3.73 KiB | 955.00 KiB/s, done.
Total 10 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/koufongso/290I-Assignment3.git
   7ac1450..3ae3dd1  main -> main

C:\Users\koufo\Desktop\290I-Assignment3>git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
```

Render Account

Render provides an easy way to deploy code to the cloud. Create a Render account at: https://dashboard.render.com/register

Problem 1: Shortest Path Solver

We will build a server that provides a shortest path solving service using Dijkstra's algorithm, and deploy it locally (on your computer).

You are provided with the Dijkstra algorithm code.

We will use the FastAPI framework. You can refer to https://fastapi.tiangolo.com/ for documentation.

API Requirements

The application should provide two APIs:

1. **Post method: /upload_graph_json**

   Uploads a JSON file containing the node connectivity and distance information.

   If the uploaded file is not a JSON file (.json), return:

```
JSON
{"Upload Error": "Invalid file type"}
```

   If successful, return:

```
JSON
{"Upload Success": "<file_name>"}
```

2. **Get method: /solve_shortest_path/starting_node_id=<str>&end_node_id=<str>**

   Solves the shortest path problem for the given graph. A and B represent the starting and ending node IDs.

   If no valid graph has been uploaded, return:

```
JSON
{"Solver Error": "No active graph, please upload a graph first."}
```

If either the start or end node ID does not exist, return:

```
JSON
{"Solver Error": "Invalid start or end node ID."}
```

Otherwise, return:

```
JSON

{
    "shortest_path": <path>,
    "total_distance": <total_distance>
}
```

where <path> is the list of nodes forming the shortest path, or None if no path exists.

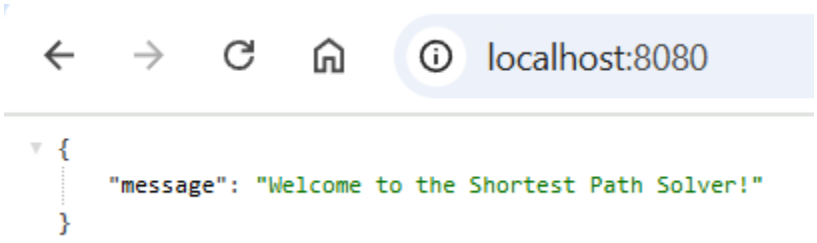
Task:

1. **Complete the `create_upload_file` and `get_shortest_path` functions in `server.py`.**
2. **Save all your code and push it to your GitHub repository. Make sure your code on GitHub is the latest version.**
3. **Include your GitHub repo link in the final submission.**

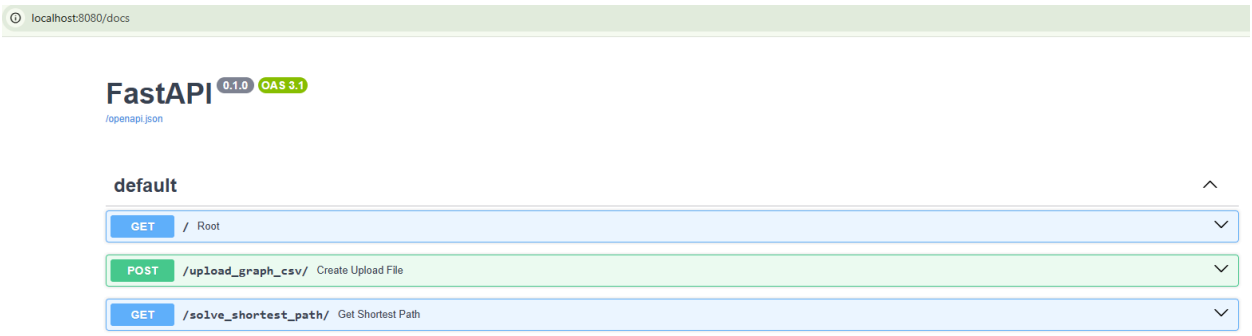
Feel free to modify any code you think is necessary.

Locally deploy your code and testing

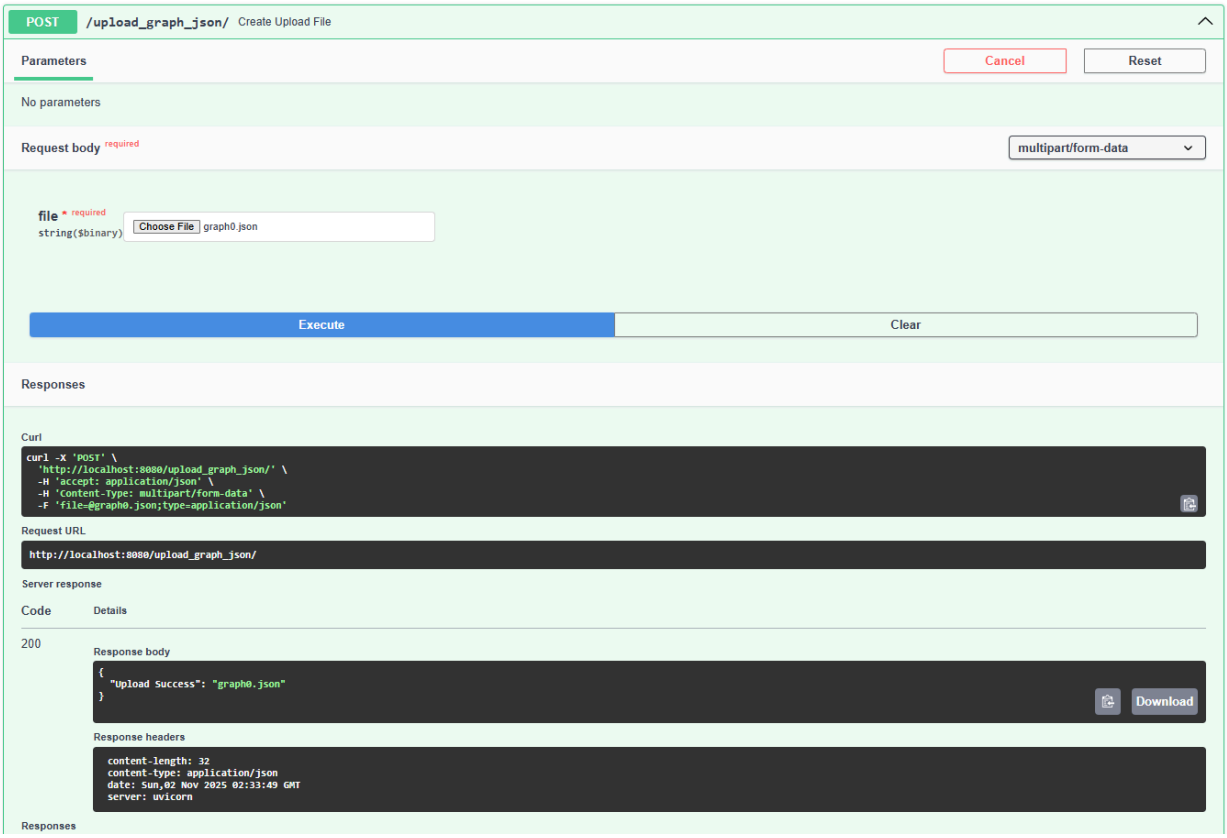
Run `server.py` to start the application locally. In your browser, go to: **http://localhost:8080/**. You should see something like this:



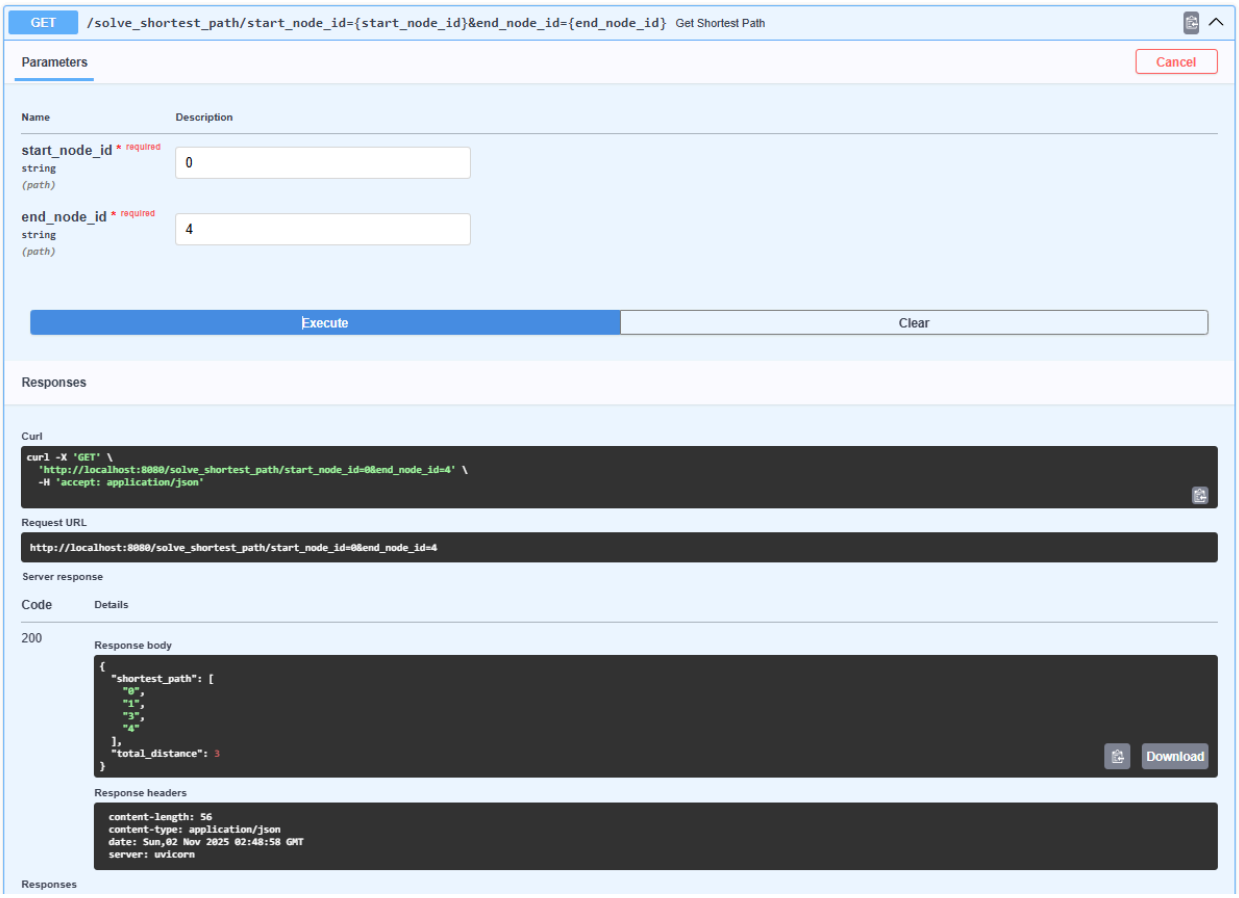
FastAPI provides an easy-to-use interface for testing and debugging your APIs: **http://localhost:8080/docs**.



Example: test /upload_graph_json



Example: test solve_shortest_path API using the provided graph0.json



Task: Attach screenshots for:

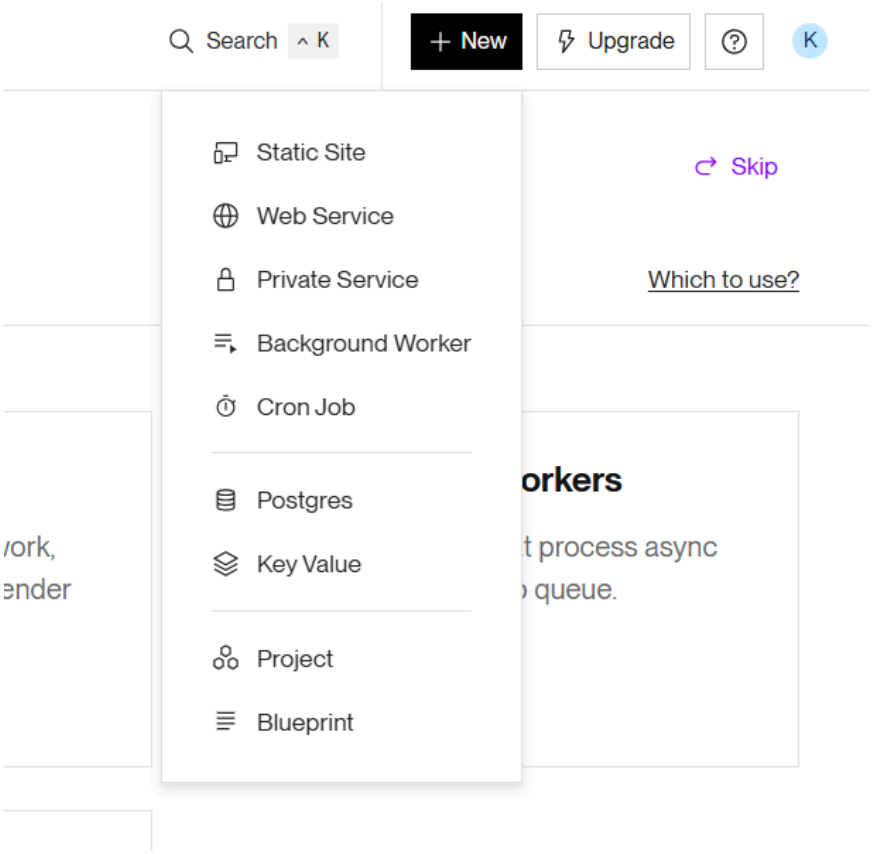
1. **Success and failure (upload a non-JSON file) case of the `upload_graph_json` API**
2. **Three possible responses of the `solve_shortest_path` API**

Problem 2: Deploying the Shortest Path Solver on Render

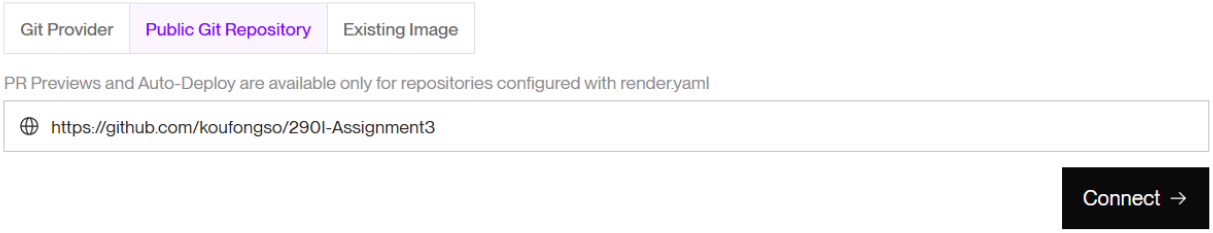
After completing Problem 1 and verifying that all APIs work correctly, you will deploy your application to the cloud using Render. (Remember to push the latest code to your GitHub repo)

1. Log in to Render and go to https://dashboard.render.com/

2. Create a Web Service

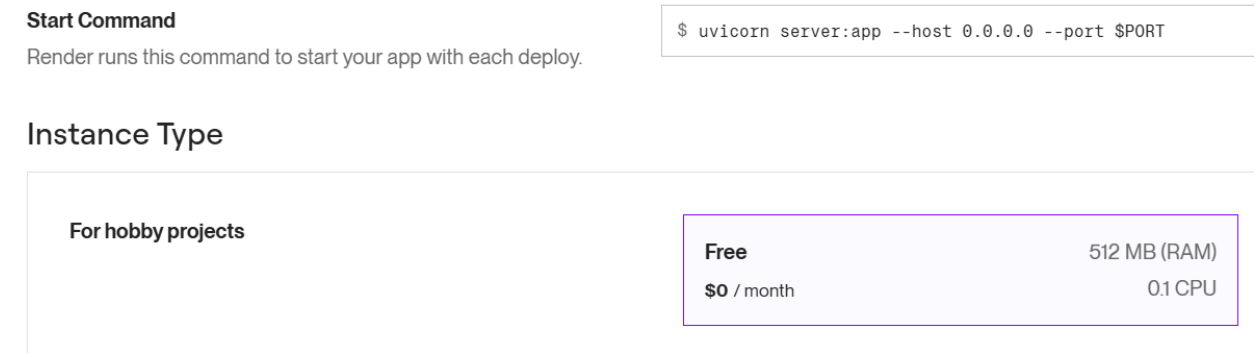


3. Connect your (public) GitHub repo to Render.

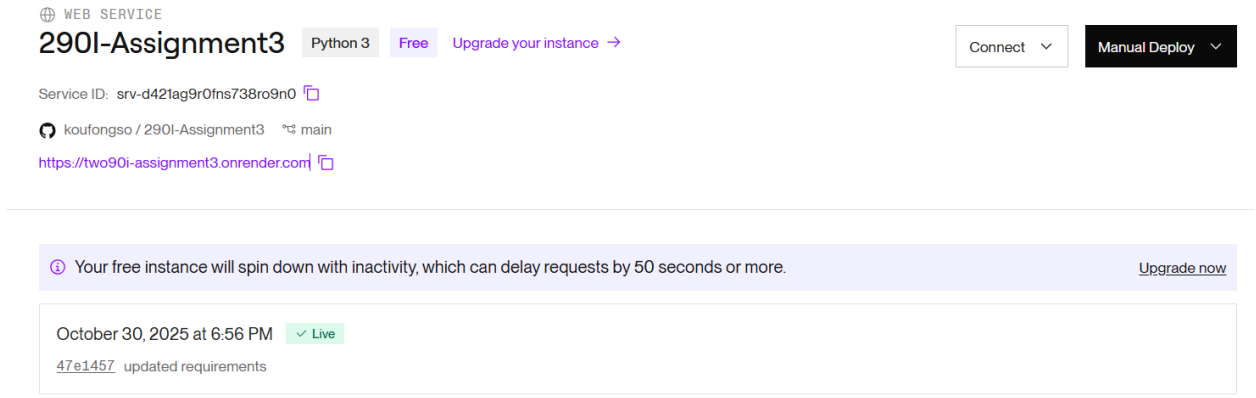


4. Name of your application and use the following **Start Command**. Use the Free Instance Type, it should be sufficient for this assignment.

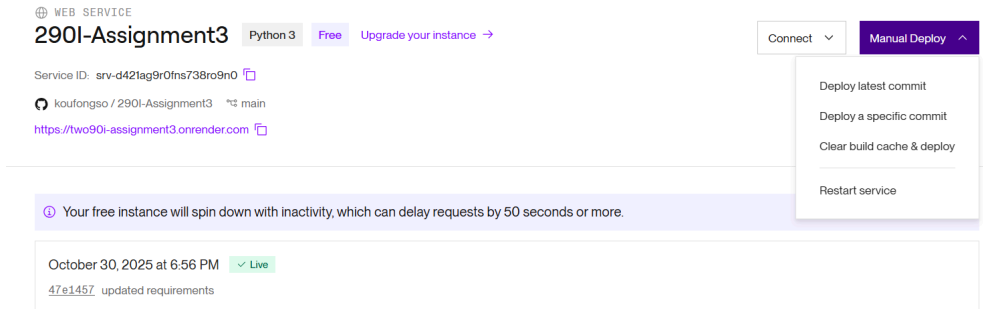
```
None
uvicorn server:app --host 0.0.0.0 --port $PORT
```



5. Deploy the web service. Render will assign a website URL to your application. You should see something like this:

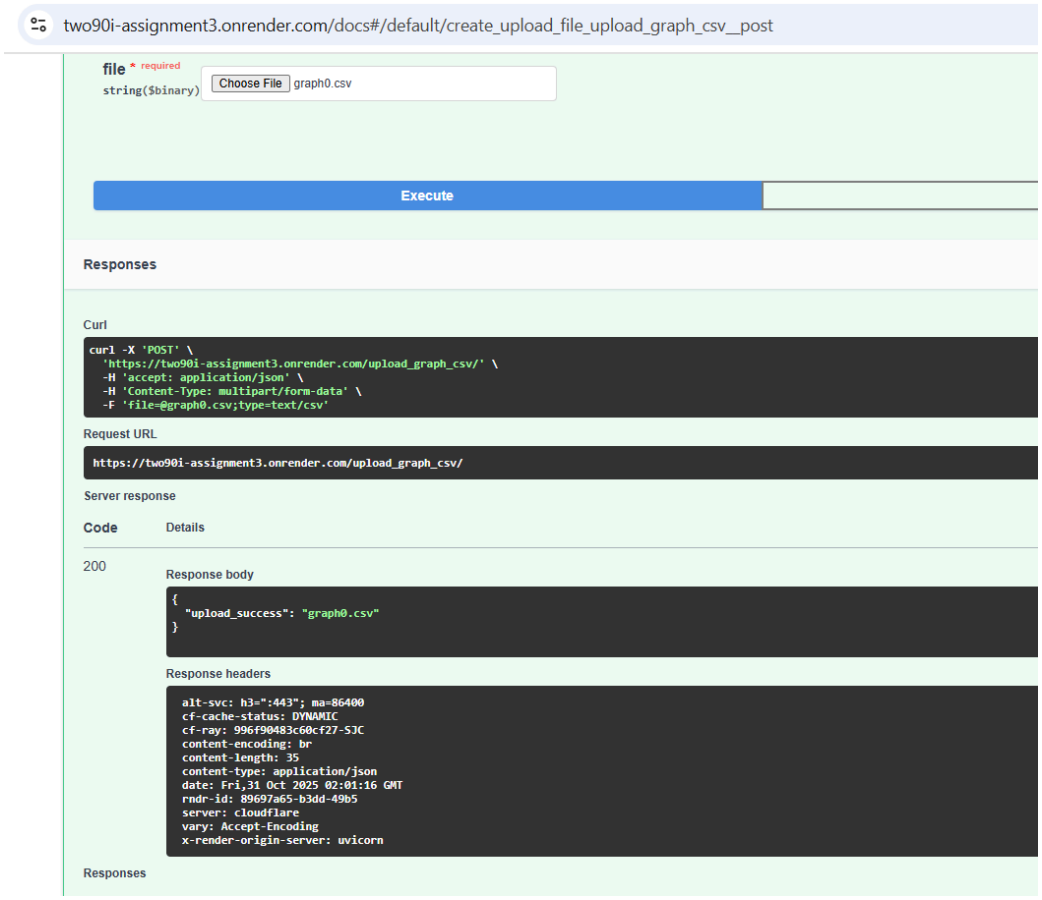


If deployment fails, fix your code locally, push the changes to GitHub, and redeploy.



You can test your cloud-deployed API using: https://your_website_url/docs

For example (noticed the website URL is not localhost now):



Deploy your application, and attach your website URL to the final submission. Keep your application running until grading is completed. You may shut it down afterward.

Submission:

Submit a PDF file that includes:

1. **Your GitHub repository link (with all source code)**
2. **Screenshots of your API testing results.**
3. **The web URL where your application is deployed.**