# Overview

The goal of this project is to create a Reddit clone using microservices.

# Teams

|  |  |
| --- | --- |
| **Name** | **Role** |
| Javier Melendrez | Developer 1 |
| Zexin Zhuang | Developer 2 |
| Nathaniel Richards | SDET |
| Antonio Lopez | Operations |

# Production Stack

This project is written in Python 3 and the deployment environment is in Tuffix (Xubuntu). You can check the version of Python by running the command python3 --version If Python 3 is not installed you can get it using the command sudo apt-get install python3 Once you have installed Python check the version of Pip you have by running the command pip3 --version Python includes Pip upon installation but if Pip cannot be found you can install it using the sudo apt-get install pip3 command. The database is handled by the microservices using SQLite3 and is viewed using API requests or DB Browser for SQLite. CURL (Client URL Request Library) requests are created using Postman. Requests update/retrieve information to/from the database. The retrieved information is outputted in JSON format.

These are the libraries/dependencies that are required for the project to run and test it. You may install them individually after setting up your working environment or run the shell script provided at the end of the SETUP section.

|  |  |
| --- | --- |
| Flask | pip3 install flask |
| Gunicorn 3 | sudo apt install --yes gunicorn3 |
| Caddy | curl https://getcaddy.com | bash -s personal |
| Foreman | gem install foreman |

# Setup

These instructions will only work for Linux based distributions, specifically Tuffix. These steps will help create a working environment while installing all libraries and dependences.

Open a terminal and create a directory:

$ mkdir reddit\_clone

Change the current directory:

$ cd reddit\_clone/

Clone repository:

$ git clone https://github.com/antonio-lopez/cspc-449-project-1.git

Create a virtual environment:

$ python3 -m venv ENV

Activate the virtual environment:

$ source ENV/bin/activate

Create the database by running the python file:

$ python3 create\_db.py

Run shell script to download and install requirements:

$ bash requirements.sh

# Run

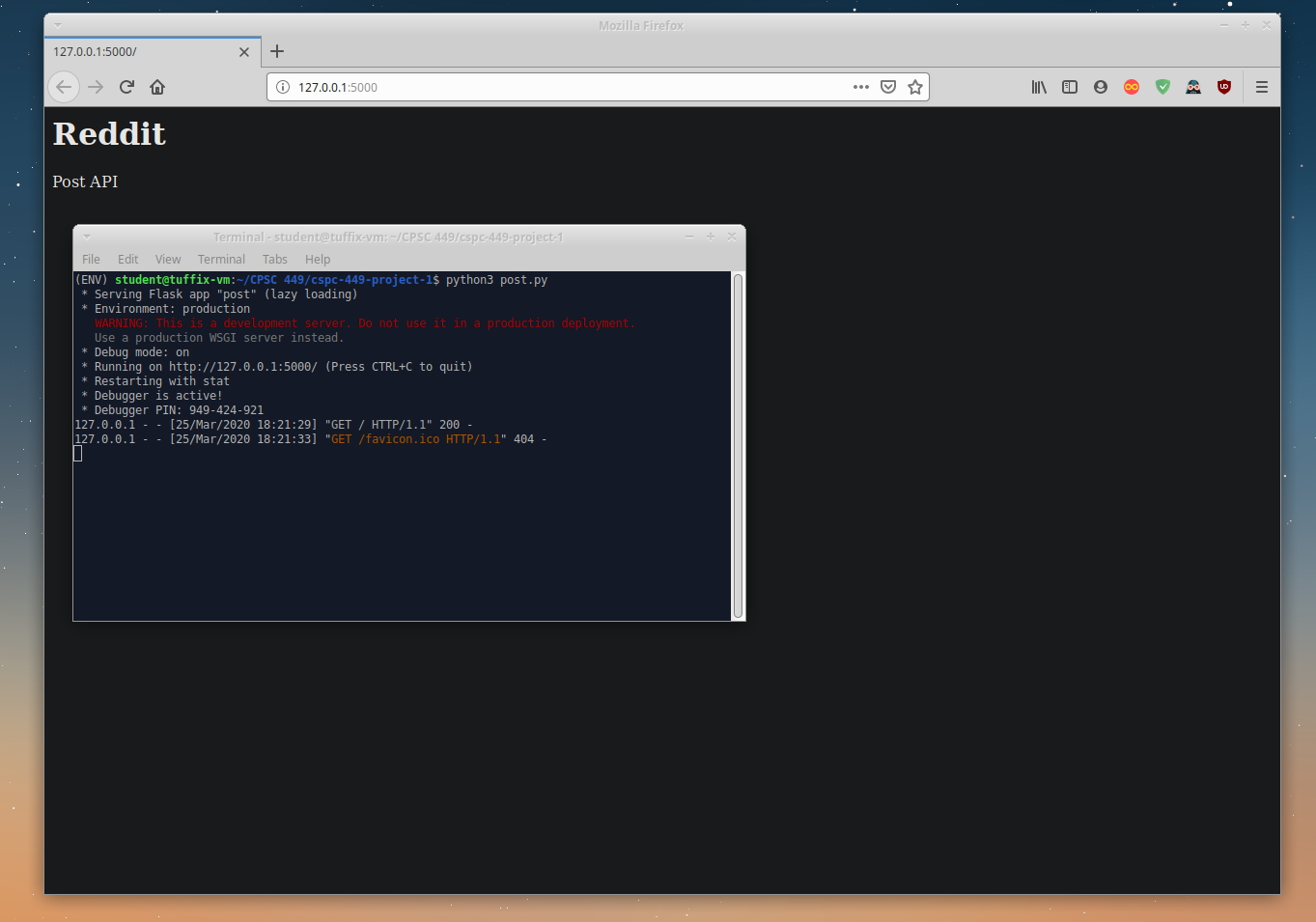
To verify the setup is complete run either the Post (post.py) or User (user.py) microservice.

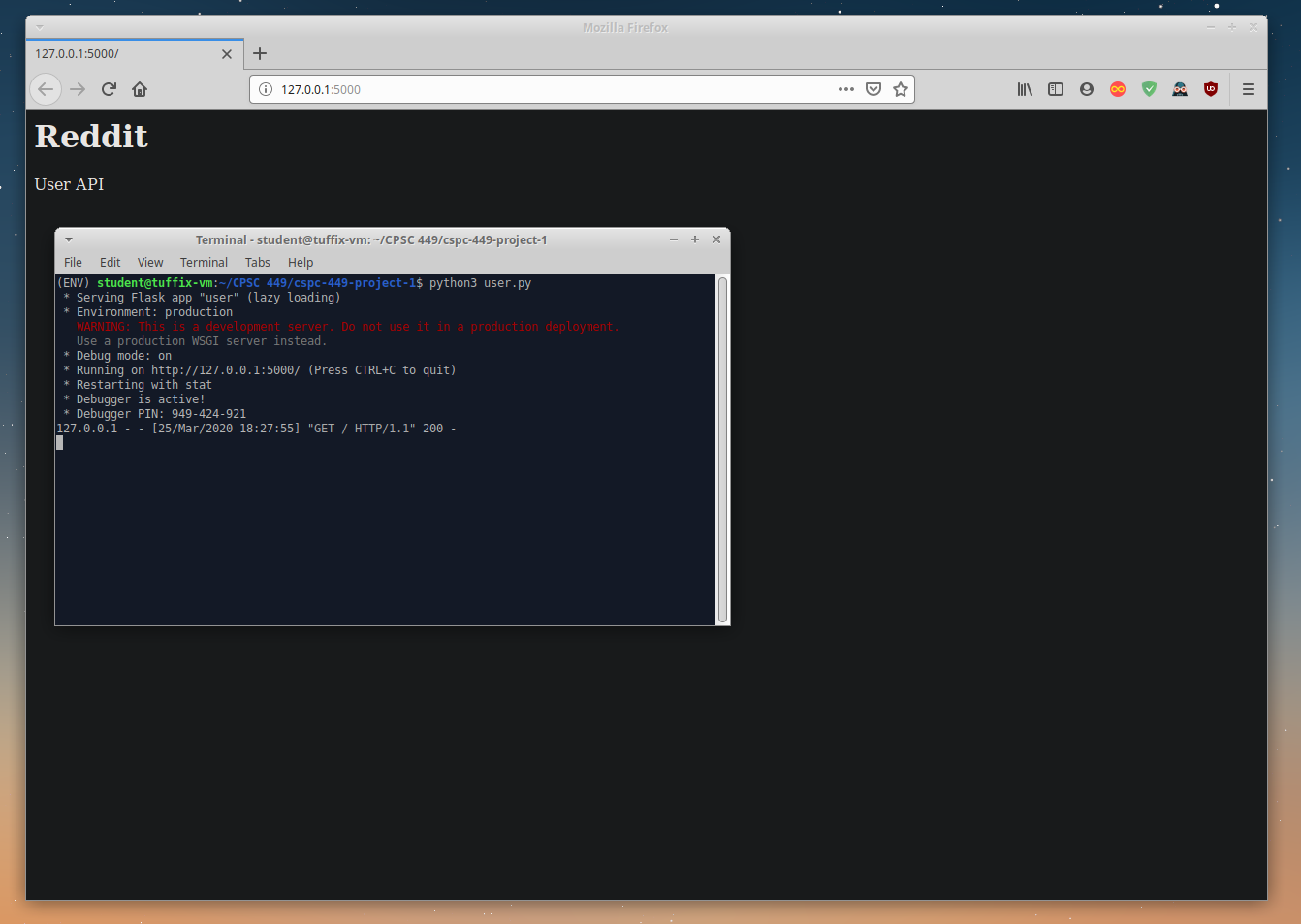
$ python3 post.py

Or

$ python3 user.py

Follow the link provided (ctrl + left click). Your browser will open with the main page greeting for the chosen microservice.





# Testing

Testing can be done in either non-production or production. In the initial testing for either, make sure you are using a newly created database. The testing scripts create new users and posts. If you are following the instructions and have completed the SETUP and RUN sections continue with testing. If the database is populated, delete it and create a new one using the $ python3 create\_db.py command.

### Testing (Non-production):

Non-production testing is done using the Flask Development Server.

Post Microservice Testing:

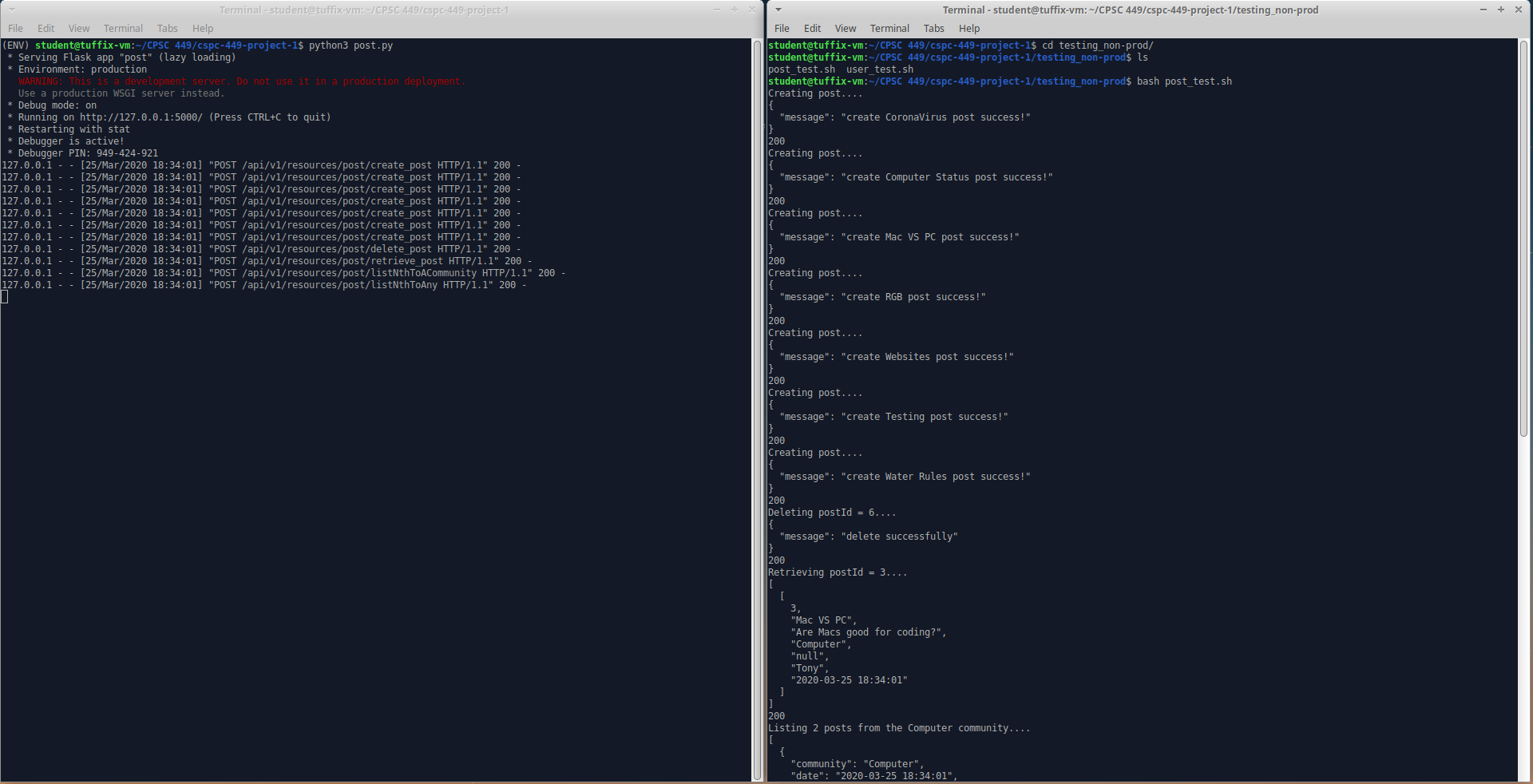
Run the Post microservice on a terminal:

$ python3 post.py

Open a new terminal, change to the testing directory, and run the Post testing shell script:

$ cd testing\_non-prod/

$ bash post\_test.sh



Running the script tests the following API requests:

* Create a new post
* Delete an existing post given a postId
* Retrieve an existing post given a postId
* List the n most recent posts to a particular community
* List the n most recent posts to any community

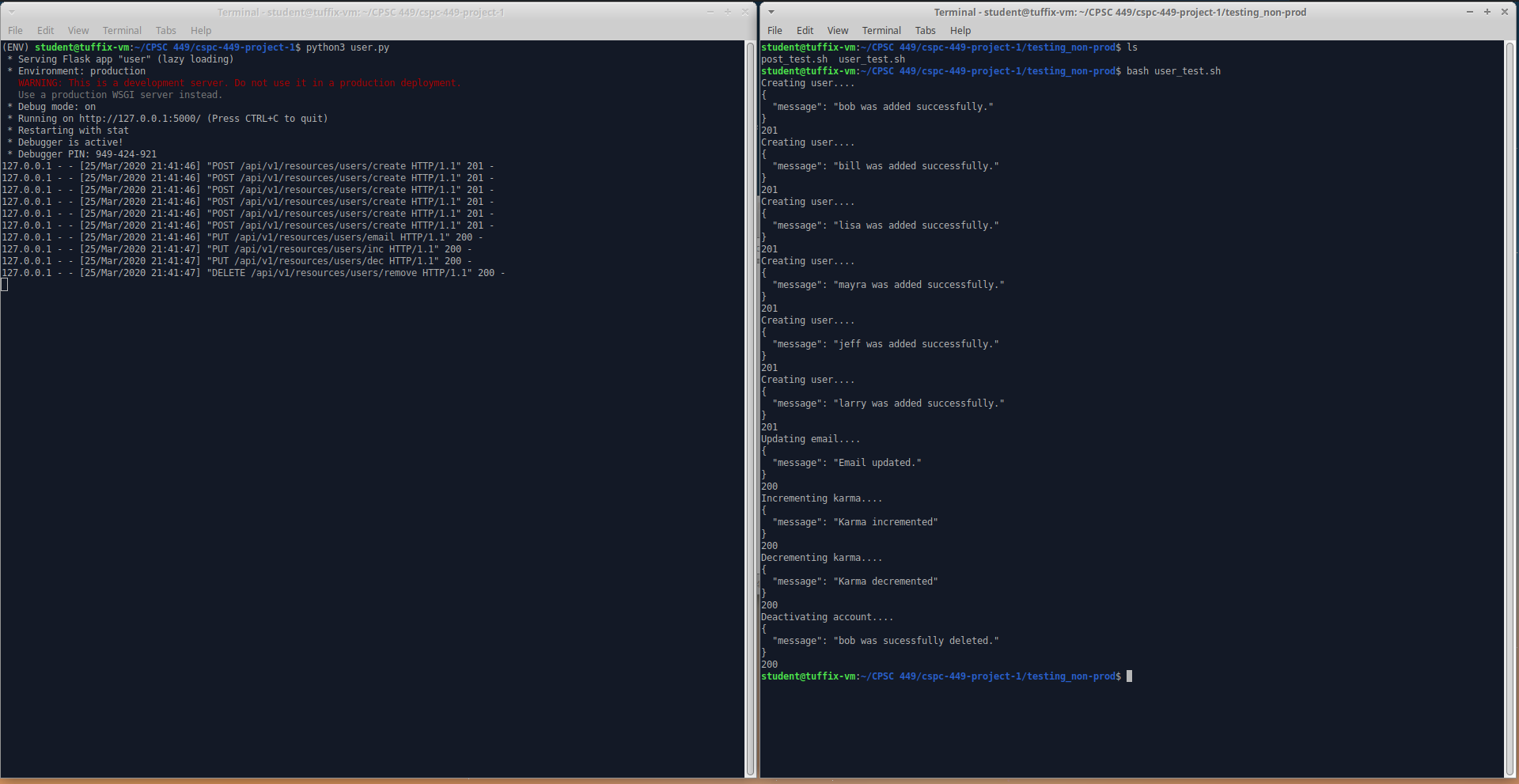
User Microservice Testing:

Cancel the Post microservice and run the User microservice:

$ python3 user.py

On the other terminal run the User testing shell script:

$ bash user\_test.sh



Running the script tests the following API requests:

* Create user
* Update email
* Increment Karma
* Decrement Karma
* Deactivate account

### Testing (Production)

Production testing is done using a WSGI server called Gunicorn. Foreman is used to run a Procfile that manages the Gunicorn and Caddy processes for each microservice. 3 instances are created for each microservice. The Caddyfile proxies’ direct requests for “http://localhost:2015/posts” to the Posts microservice and requests for “http://localhost:2015/users” to the Users microservice. It also handles the load balancing method which in this case is round robin.

Post Microservice Testing:

Open 4 terminals, be sure to activate the virtual environment for each terminal using the $ source ENV/bin/activate command.

On one terminal run Foreman for the Post microservice:

$ foreman start -m posts=3

On the second terminal run Foreman for the User microservice:

$ foreman start -m users=3

On the third terminal run the Caddyfile:

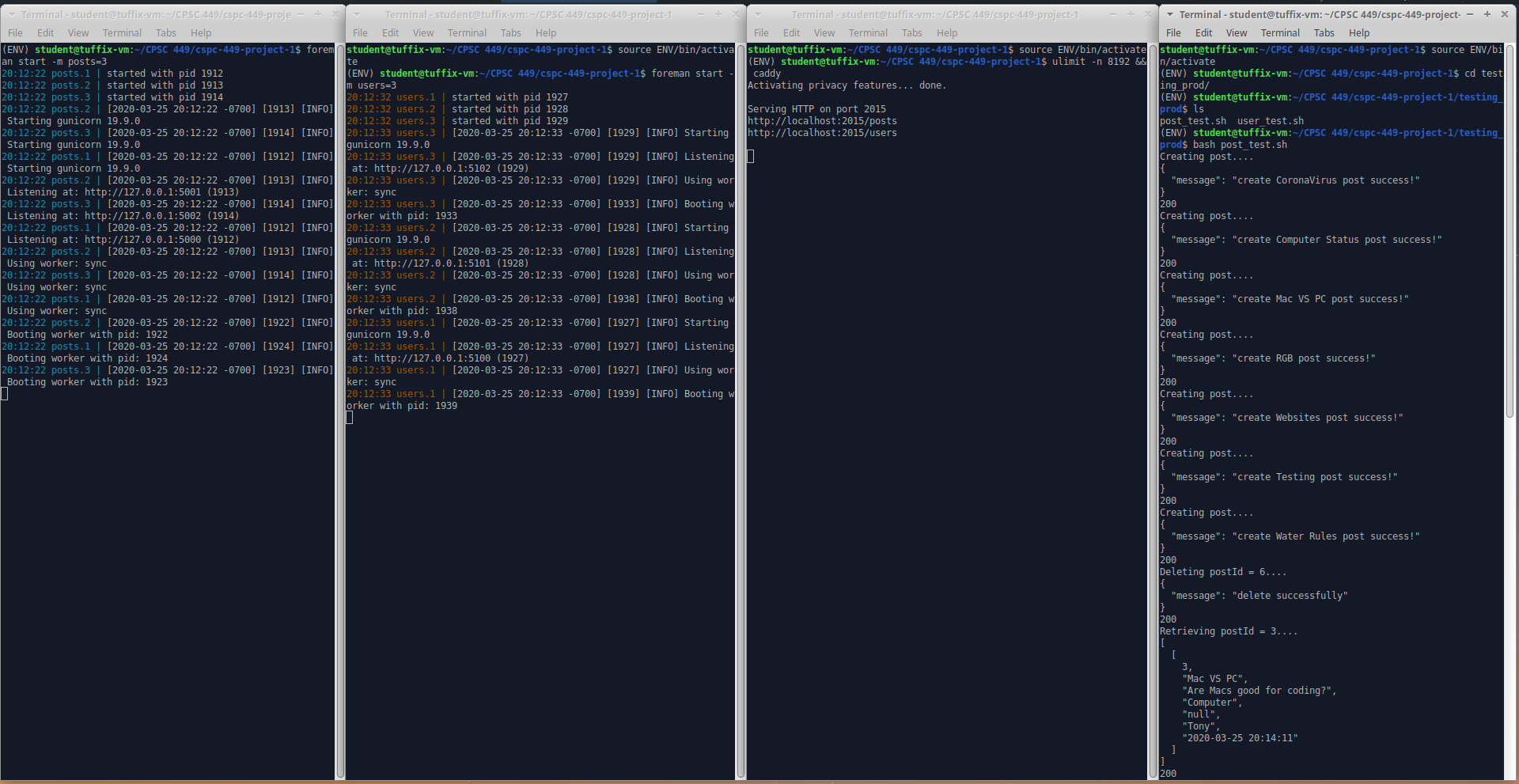
$ ulimit -n 8192 && caddy

On the fourth terminal change to the testing\_prod directory

and run the Post testing shell script:

$ cd testing\_prod/

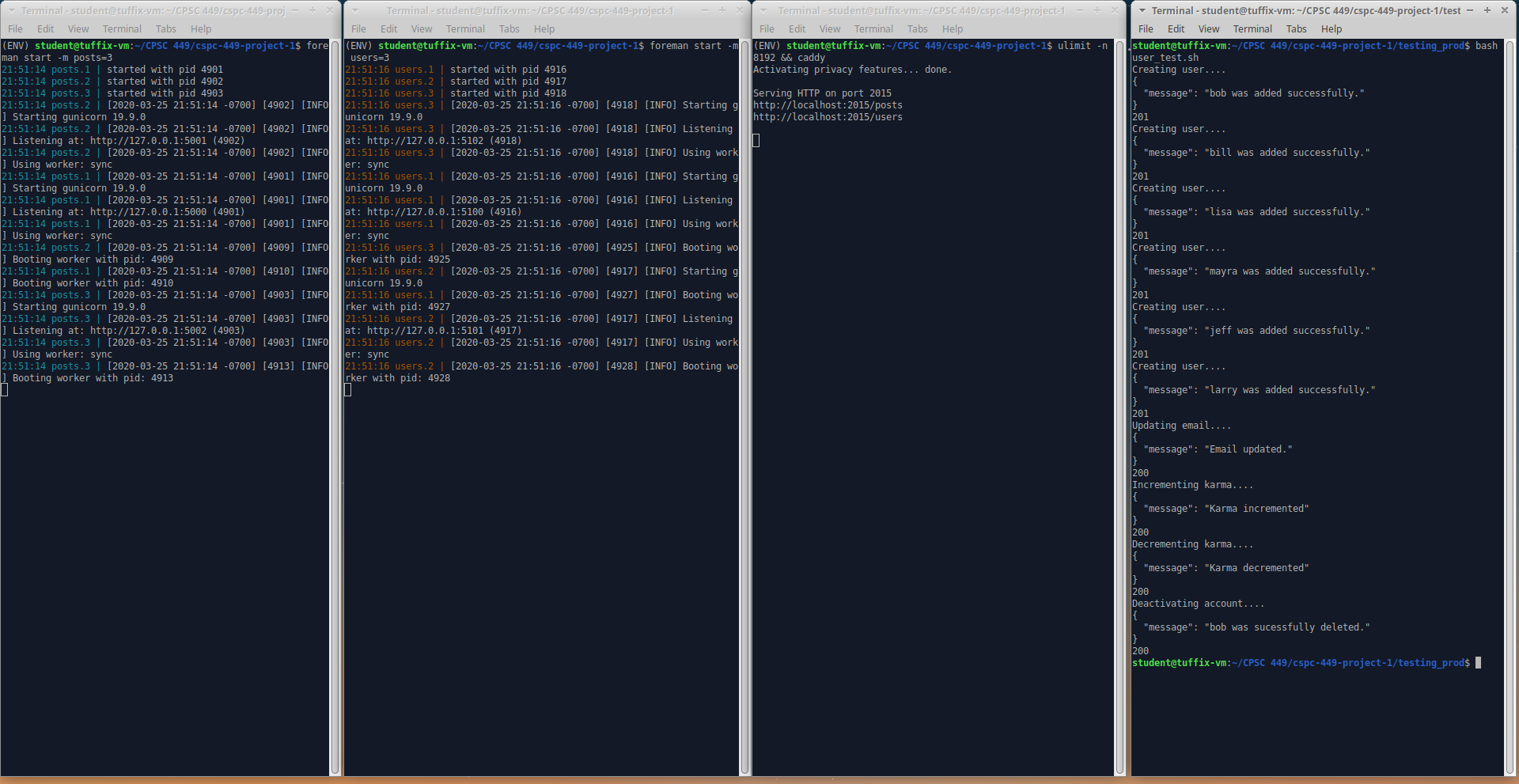
$ bash post\_test.sh



User Microservice Testing:

Cancel the Post testing shell script and run the User testing shell script:

$ bash user\_test.sh



### Testing for Duplicates

Needs work

# API Requests

Enter requests and codes