

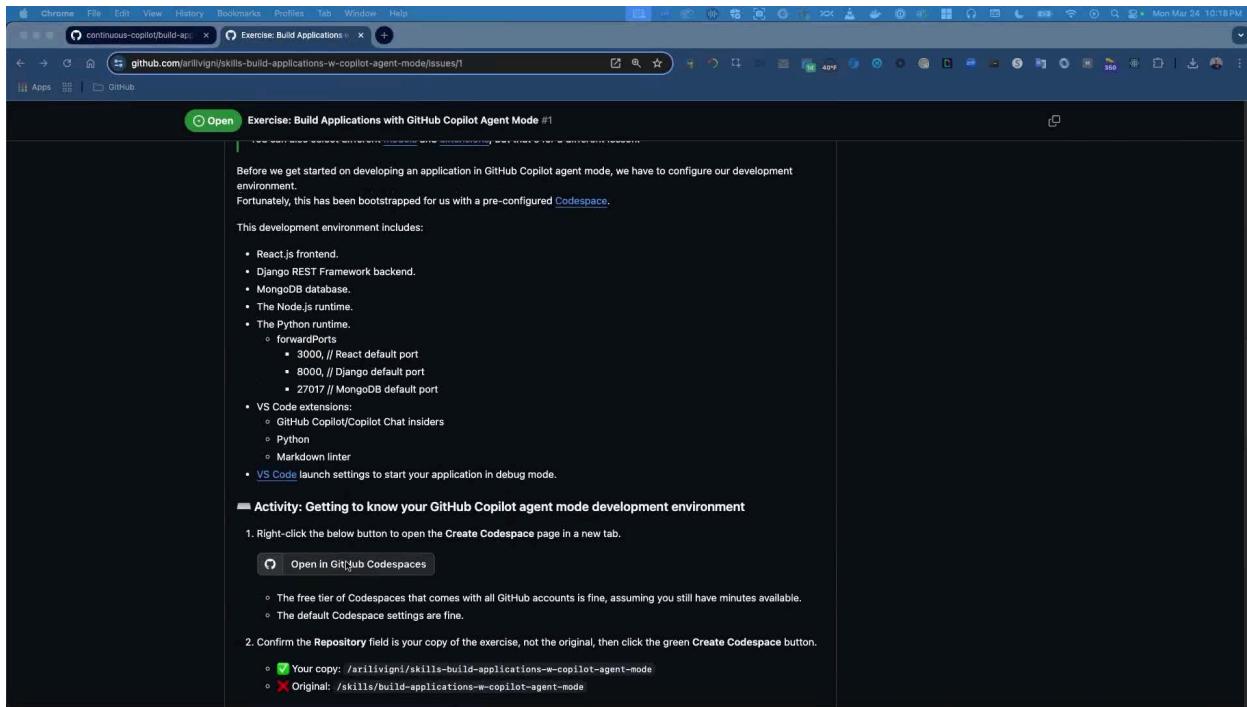
# AI Skills Fest - Train-the-Trainer - Build Applications with GitHub Copilot Agent Mode

## Step 1: Create a Repository 0:05



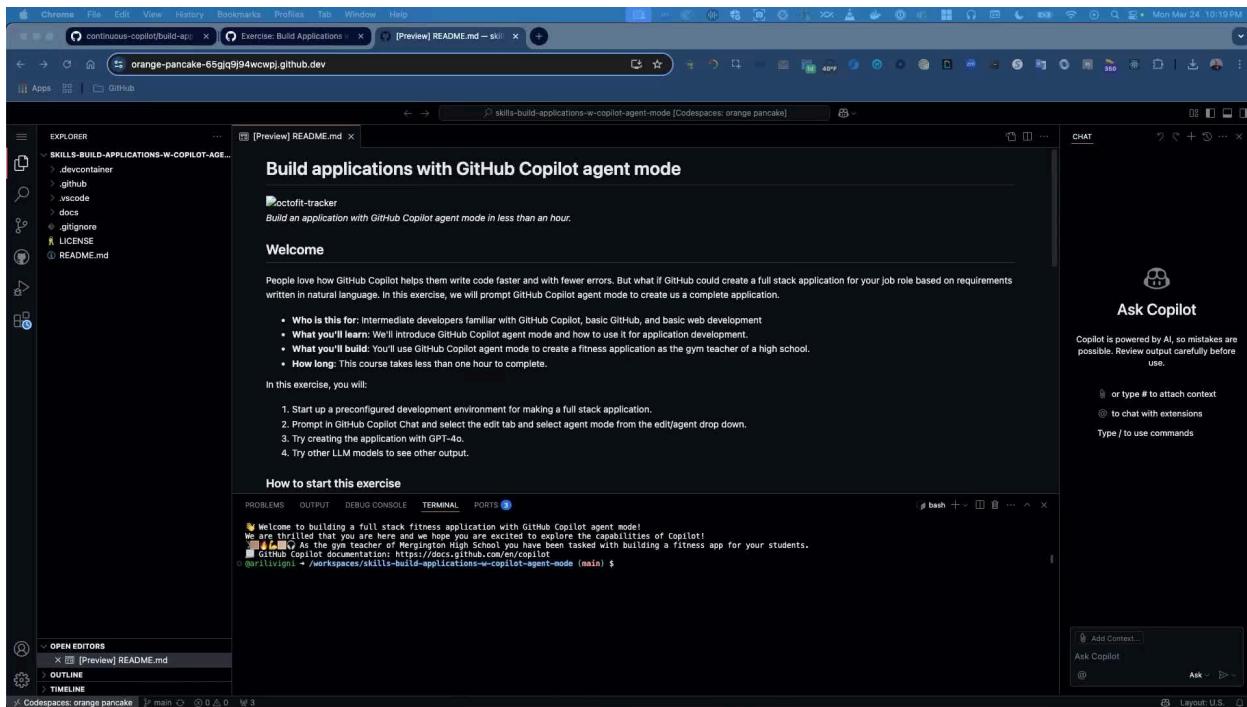
- Welcome to the tutorial on building applications with Co-Pilot Agent Mode.
- Copy the exercise to your GitHub handle or organization.
- Add a note to the description and create the repository.

## Step 2: Open in GitHub Code Spaces 0:56



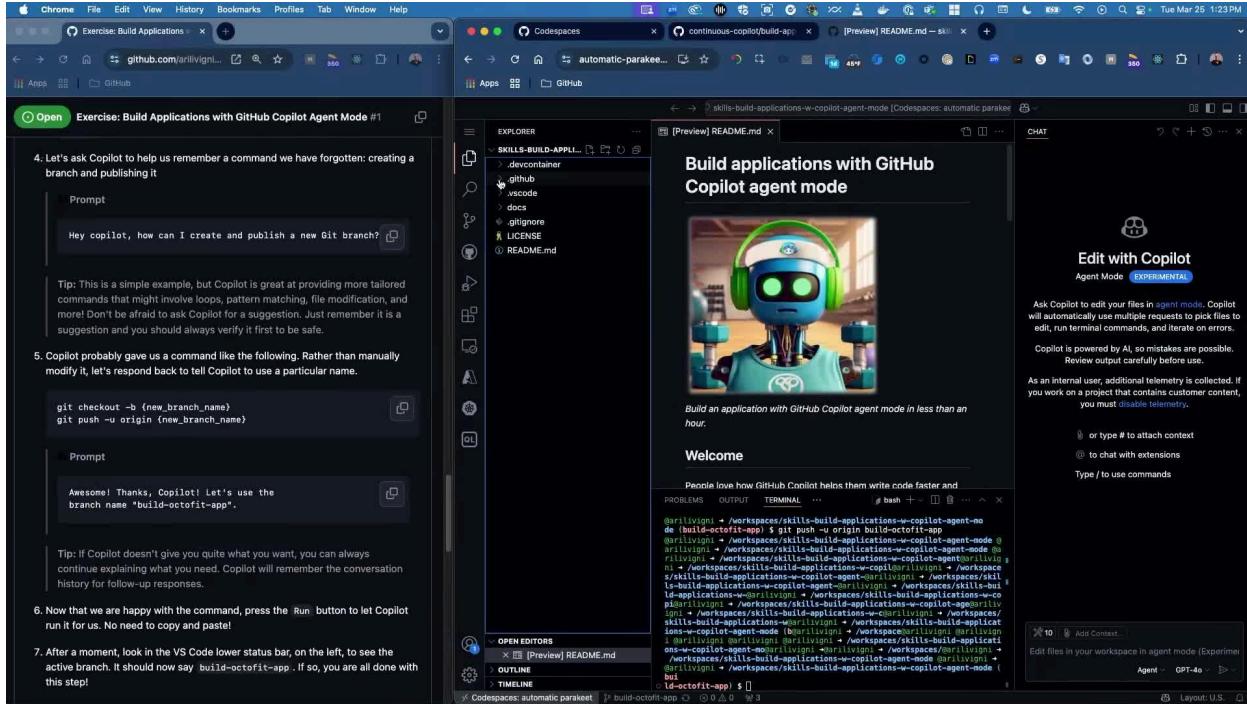
- Right-click to open in GitHub Code Spaces and create a new code space.
  - This environment includes necessary extensions like GitHub Copilot and Copilot Chat.

Step 3: Create a Branch 1:27



- Copy the first prompt and use Command-I (or Control-I on Windows) to ask Copilot for a command to create a branch.
- Create a specific branch called 'BuildOctoFitApp'.

## Step 4: Prepare Files 2:14



- Open the README file with information about the reference app.
- Ensure Copilot Chat Agent Mode can use this as a reference.

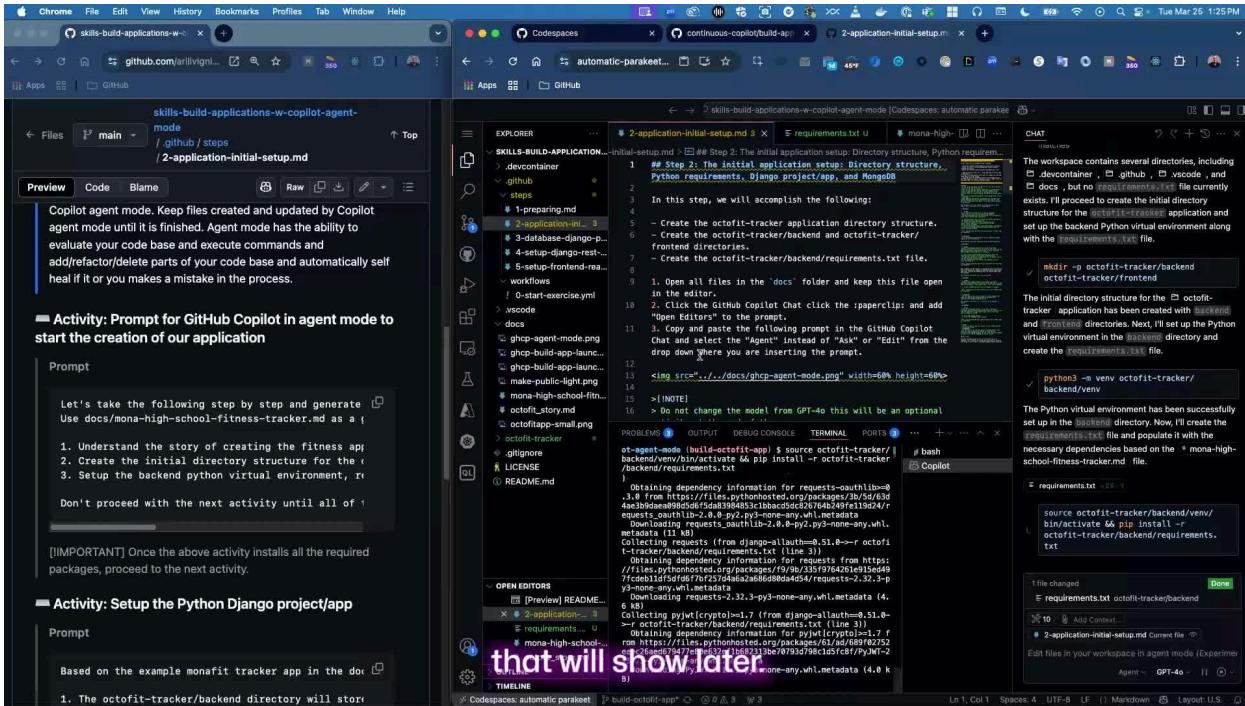
## Step 5: Set Up Application Structure 3:01

- Copilot identifies the need to create the structure for the OctoFit tracker app.
- Create a virtual environment for Python development.

## Step 6: Create Requirements File 3:26

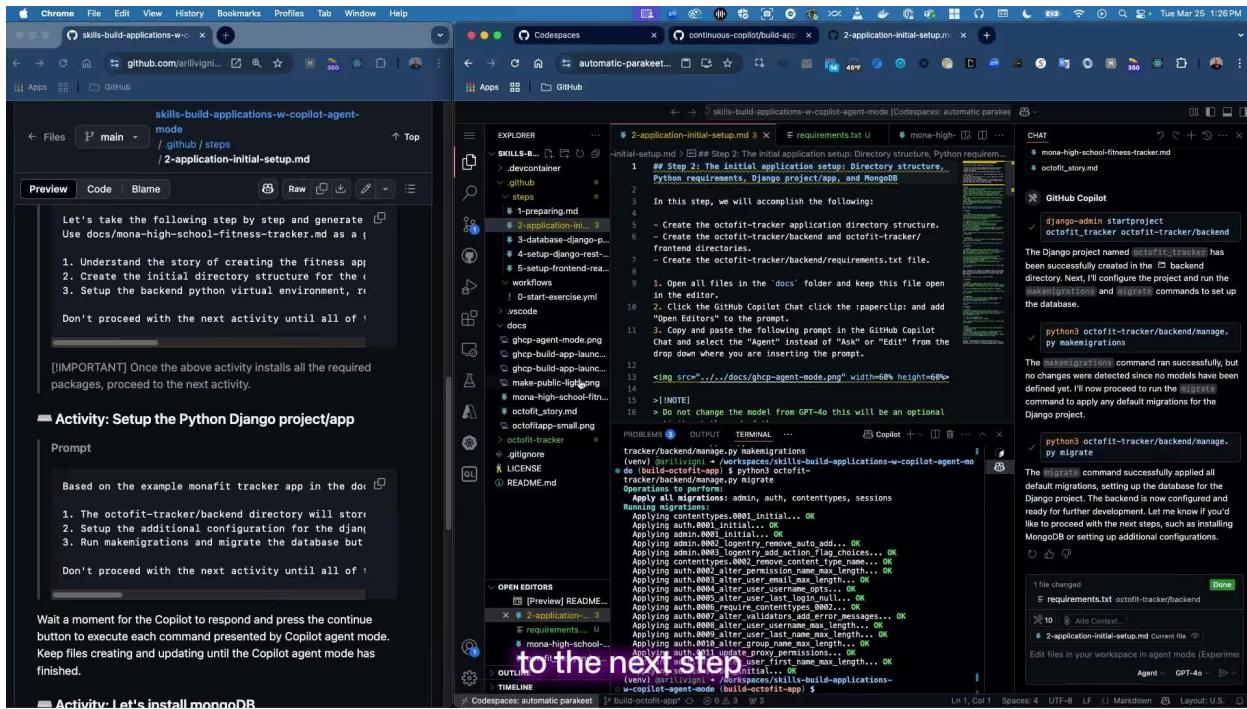
- Generate a requirements.txt file with necessary modules and libraries for the app.
  - Install the required packages.

Step 7: Set Up Django Backend 4:33



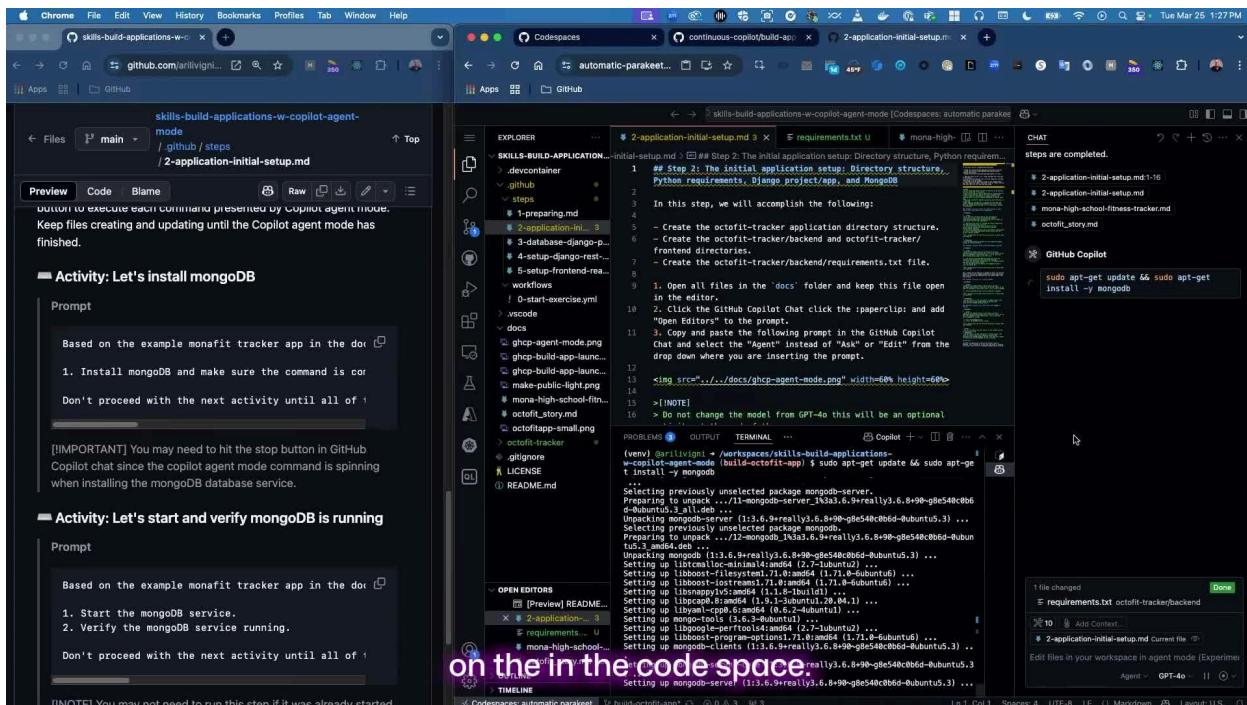
- Ask Copilot to create the Django backend for the project.
  - Run system commands related to Django to migrate the database.

## Step 8: Install MongoDB 5:08



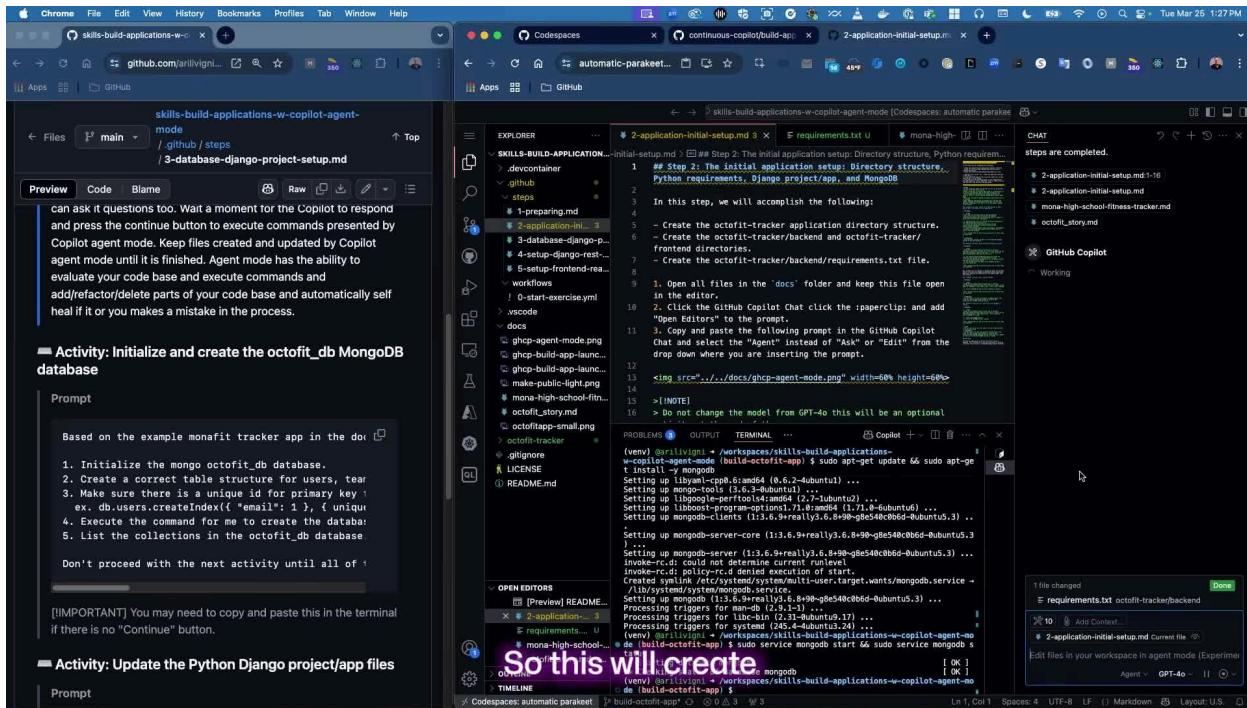
- Install MongoDB using Copilot's reference command

## Step 9: Start MongoDB Service 5:46



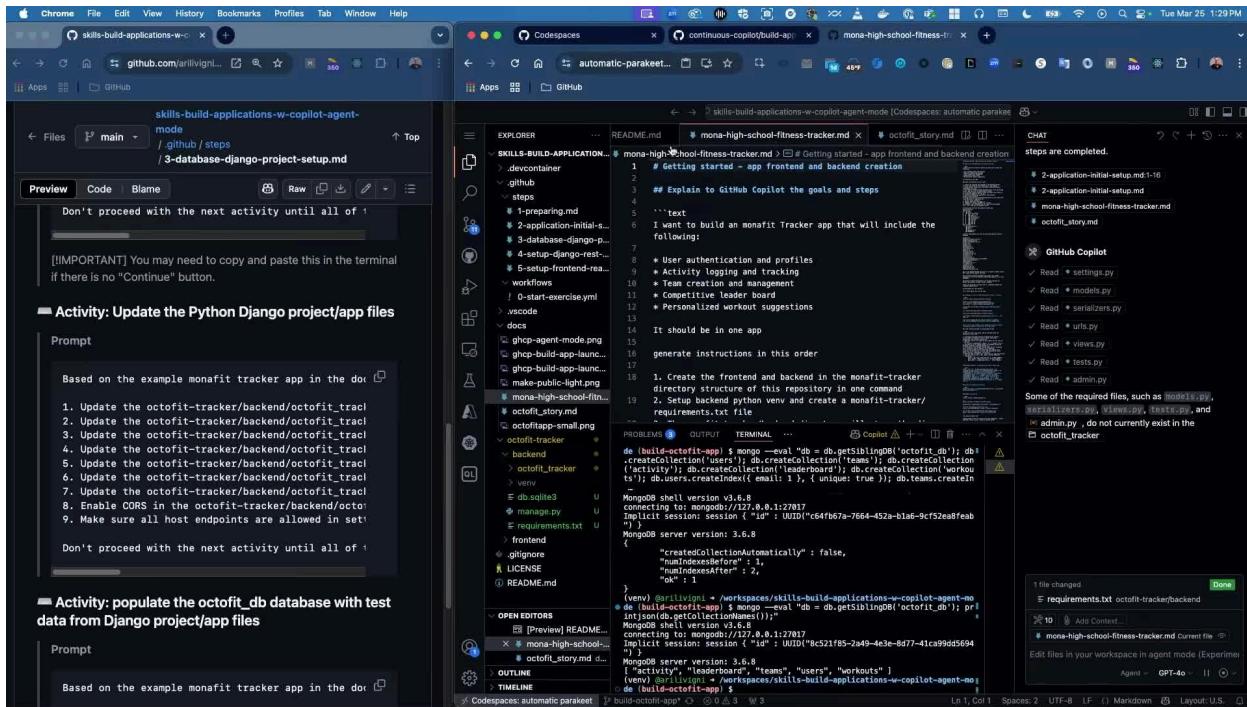
- Ensure the MongoDB service is running

## Step 10: Create Database Structure 6:01



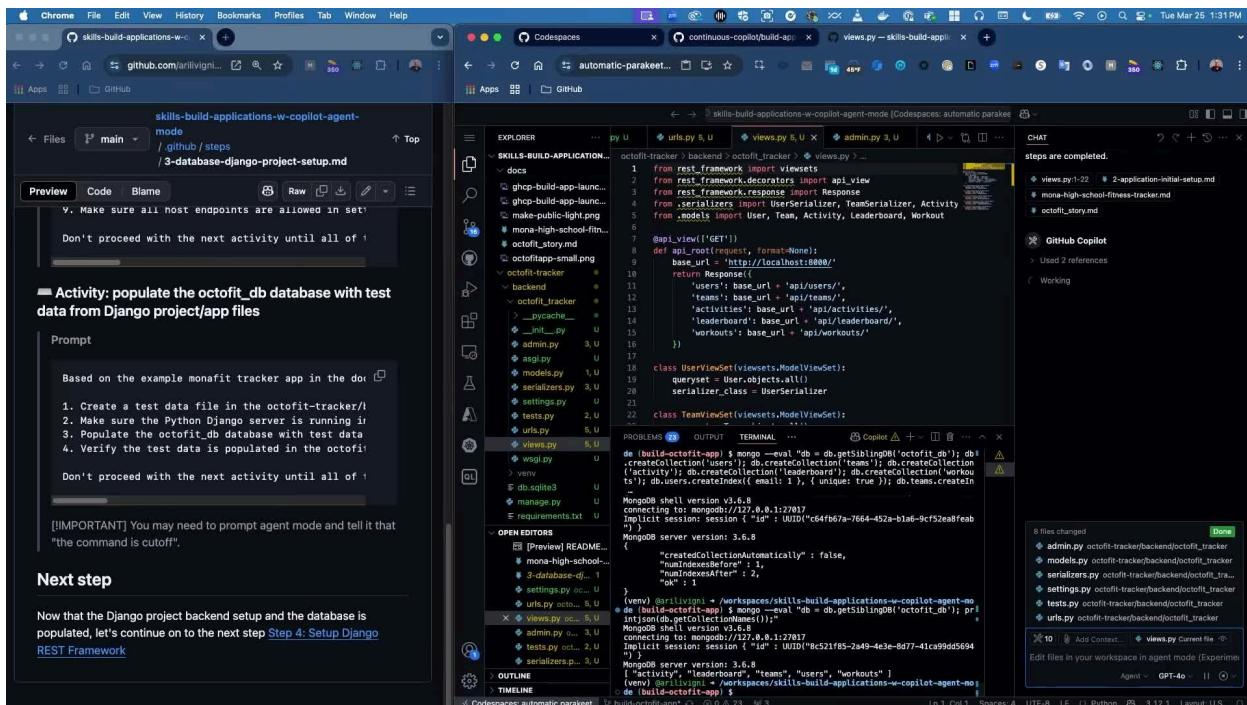
- Set up the database structure with collections for leaderboard, activities, users, teams, and workouts.

## Step 11: Update Django Files 6:45



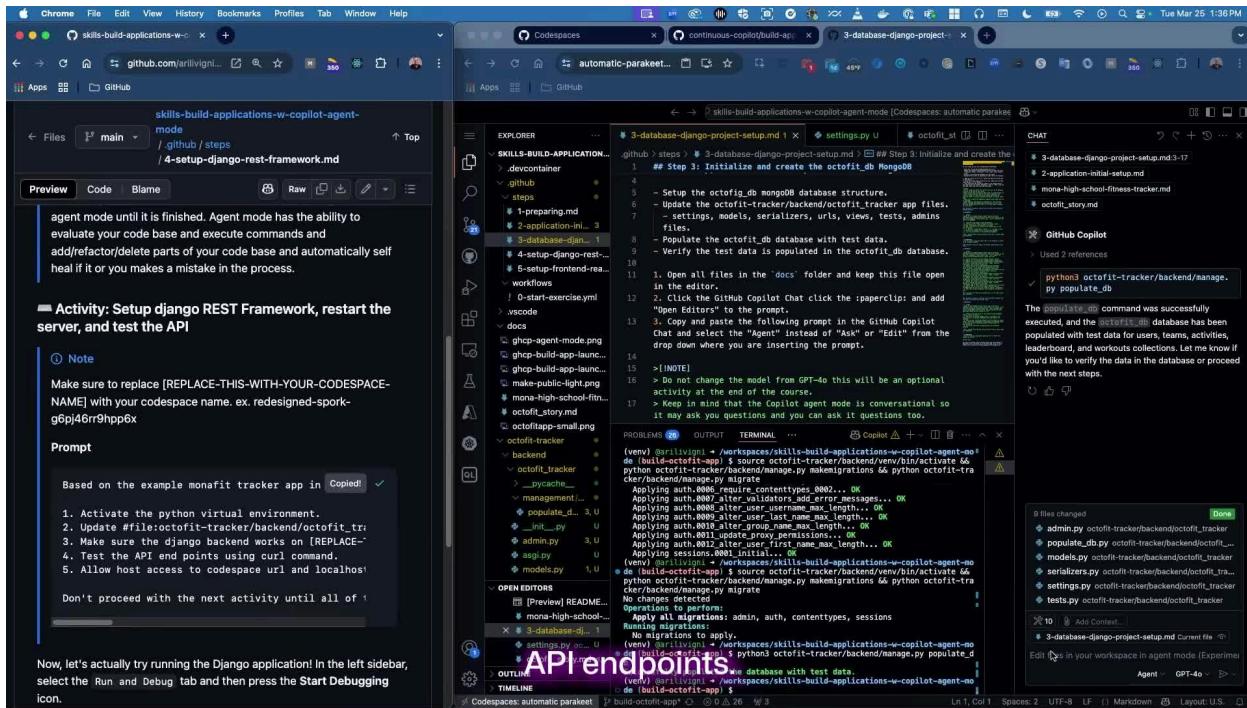
- Update Django app files including views, serializers, URLs, and models.

Step 12: Create Admin and Test Files 7:30



- Copilot creates admin.py and test.py files for the Django project.

Step 13: Write Database Population Script 8:03



- Write a script called `populate_db.py` to populate the database.

Step 14: Create REST API Endpoints 9:53

- Create REST API endpoints using the code space name and update `views.py`.

## Step 15: Make Ports Public 11:30

- Make both the React front end (port 3000) and Django back end (port 8000) public.

## Step 16: Update Component Files 12:20

The screenshot shows a developer's workspace with the following elements:

- Top Bar:** Chrome browser with tabs for "skills-build-applications-w-copilot-agent-mode", "github.com/arill...", "React App", and "User List - Django REST fram...".
- Left Sidebar:** "EXPLORER" view showing the project structure of "skills-build-application-w-copilot-agent-mode".
- Center Editor:** "JS Users.js" file open in the "REACT" editor. The code is as follows:

```
3  function Users() {
4    useEffect(() => {
5      .catch(error => console.error('Error fetching users
6      )));
7    });
8  }
9
10  return (
11    <div className="card">
12      <div className="card-body">
13        <h1 className="card-title">Users</h1>
14        <ul className="list-group">
15          {users.map(user => (
16            <li className="list-group-item" key={user._id}>
17              {user.name}
18            </li>
19          ))}
20        </ul>
21      </div>
22    </div>
23  );
24}
25
```

- Bottom Editor:** "JS Users.js" file open in the "OPEN EDITORS" list.
- Bottom Bar:** "PROBLEMS", "OUTPUT", "DEBUG CONSOLE", "PORTS", and "...".
- Right Side:** "PORTS" table showing forwarded ports:

Port	Forwarded Address	Running Process	Visibility
3000	https://jubilant...	/user/local/share/...	Public
8000	https://jubilant...	/workspaces/ski...	Public
27017	https://jubilant...	Process informa...	Private
43709	https://jubilant...	/home/codespa...	Private
47007	https://jubilant...	/home/codespa...	Private

- Bottom Right:** GitHub Copilot interface showing "14 files changed" and a list of files.

- Update JavaScript files for each component to improve formatting and appearance.

## Step 17: Beautify the App 14:38

Activity: Let's make the octofit tracker nice, pretty, and add some color

Prompt

Based on the example monafit tracker app

- Edit the App.css file to do the following
- Add some color to the background.
- Add some color to the text.
- Add some color to the tables.
- Add some color to the buttons.
- Add some color to the headings.
- Add some color to the links.
- Add some color to the navigation menu.

Don't proceed with the next activity unless you've completed this one.

this kind of shows where we're headed.

- Add color and background to enhance the app's visual appeal.

## Step 18: Encourage User Interaction 15:02

Activity: Let's make the octofit tracker nice, pretty, and add some color

Prompt

Based on the example monafit tracker app

- Edit the App.css file to do the following
- Add some color to the background.
- Add some color to the text.
- Add some color to the tables.
- Add some color to the buttons.
- Add some color to the headings.
- Add some color to the links.
- Add some color to the navigation menu.

Don't proceed with the next activity unless you've completed this one.

Thank you.

- Allow users to suggest changes or features for the app in future exercises.

## **Link to Loom**

<https://www.loom.com/share/e5faf03cbc164cf6b0020dae469e2847>