Introduction

Part 1, Chapter 1

Changelog »

Uber App Using Django Channels

Many apps rely on real-time, bi-directional communication to give users a great experience. One example is a ride-sharing app like Uber or Lyft, which is built on the messages that are sent between a rider and a driver. A rider selects a starting location and destination, and then the app broadcasts a trip request to all nearby drivers. An available driver accepts the trip and meets the rider at the pick-up address. In the meantime, every move the driver makes is sent to the rider almost instantaneously and the rider can track the trip status as long as it is active.

In this course, we'll demonstrate how to program a ride-sharing app using the bi-directional communication that WebSockets and Django Channels provide. We'll then tie it all together by creating a nice UI with React.

The instruction will be given in three parts:

1. **Part 1**: Using Test-Driven Development, we'll write and test the server-side code powered by Django and Django Channels.
2. **Part 2**: We'll set up the client-side React app along with authentication and authorization. Also, we'll streamline the development workflow by adding Docker.
3. **Part 3**: Finally, we'll walk through the process of creating the app UI with React.

In the end, you'll have an app with two user experiences: one from the perspective of the driver and the other from the rider. You'll be able to access both experiences simultaneously in order to see how a trip is planned and executed in real-time.

> At any time, consult the taxi-react-app repository to compare your code to the source of truth.

Our server-side application uses:

- Python (v3.10)
- Django (v4.1)
- Django Channels (v4.0)
- Django REST Framework (v3.14)
- pytest (v7.2)
- Redis (v7.0)
- PostgreSQL (v15.0)

Client-side:

- React (v18.2)

We'll also use Docker Engine (v20.10) and Docker Compose (v2.12).

Feedback

Objectives

By the end of Part 1, you'll be able to:

1. Create a RESTful API with Django REST Framework.
2. Implement token-based authentication with JSON Web Tokens (JWT).
3. Use Django Channels to create and update data on the server.
4. Send messages to the UI from the server via WebSockets.
5. Test asyncio coroutines with pytest.

Changelog »

✓ Mark as Completed