How much time did you end up spending on it?
It took me about 3-4 hours to lookup what technologies are being used for authentication across multiple applications and to learn basics of OAuth2 and the corresponding django plugin. I'd like to add that I'd be used to handle single-token authentication but this is the first time I had to implement authentication across multiple apps. Coding took me 5 hours and I also spent roughly half an hour writing documentation.

What are some of the design decisions you made?
- I wanted to stick to as many industry-standard technologies and frameworks as possible given this is a fairly standard task, done countless times by many other professionals and any attempts on reinvention of wheel would be both a waste of time and a security risk.
- Structure of the project should be clean and easy to extend for additional functionality.
- Should provide an easy interface for other apps to register.
- Result should be easy to configure and deploy.

The combination django + rest and oauth2 frameworks + docker meets all of the above.

What do you like about your implementation?
- It's easy to connect with third party apps without any session id stores or any other requests apart from registration.
- It's self-contained, all configurations can be done via env vars, so it would be easy to deploy and scale using e.g. kubernetes.
- It can be easily extended for different user roles and corresponding permissions (already present in django so would only need to define the groups and permissions).

What would you improve next time?
I'd spend some time replacing internal requests between the UI and OAuth2 with a proper ORM-based token generation. (would need to dive deeper into the oauth2 plugin, didn't have enough time for that during the 5 hours) I'd also add proper tests to check connecting different applications, user authentication across the apps and also corresponding permissions.

What things are missing to make it production-ready?
We'd need a reverse proxy for SSL (to make sure all requests/data/tokens are encrypted). Also would need to move from sqlite to e.g. postgres running in a separate container to allow for horizontal scaling of both the database (e.g. using kubernetes stateful pods) and the app itself (this is easy as it is stateless). Also would need to map volumes from the db container to the host (or an external store) to make sure user profiles are persistent even if the db containers were deleted.