Software Engineering Immersive Course: Project 4 - Django + React App

Ringers

Brief

Build a full stack app by making your own front and back end using a Python Django API. The app will most likely have multiple relationships and CRUD functionality for at least a couple of models.

Timeframe: 1 week, independent coding

Summary

Ringers is a full stack app for sourcing team players at the last minute. I used Python and Django REST Framework to serve my data from a Postgres database (visualised in TablePlus,) and React on the front end. The app has multiple relationships and CRUD functionality.

Deployment

Installation Instructions

To run the app from the source code, use the clone button to download the source code from GitHub. From the root directory type "yarn", "yarn seed" and "yarn start" in the terminal. The project will run on localhost:4000, and will be viewable on any web browser.

Technologies & Methodologies Used

- HTML 5
- CSS 3
- Bulma
- SASS
- JavaScript (ES6)
- React.js
- React Router
- Python
- Django
- PostgreSQL
- Webpack
- Yarn
- Insomnia
- Adobe Illustrator
- Adobe Photoshop
- Wireframes
- MVC Architecture
- Authentication
- RESTful APIs
- CRUD Functionality

Features

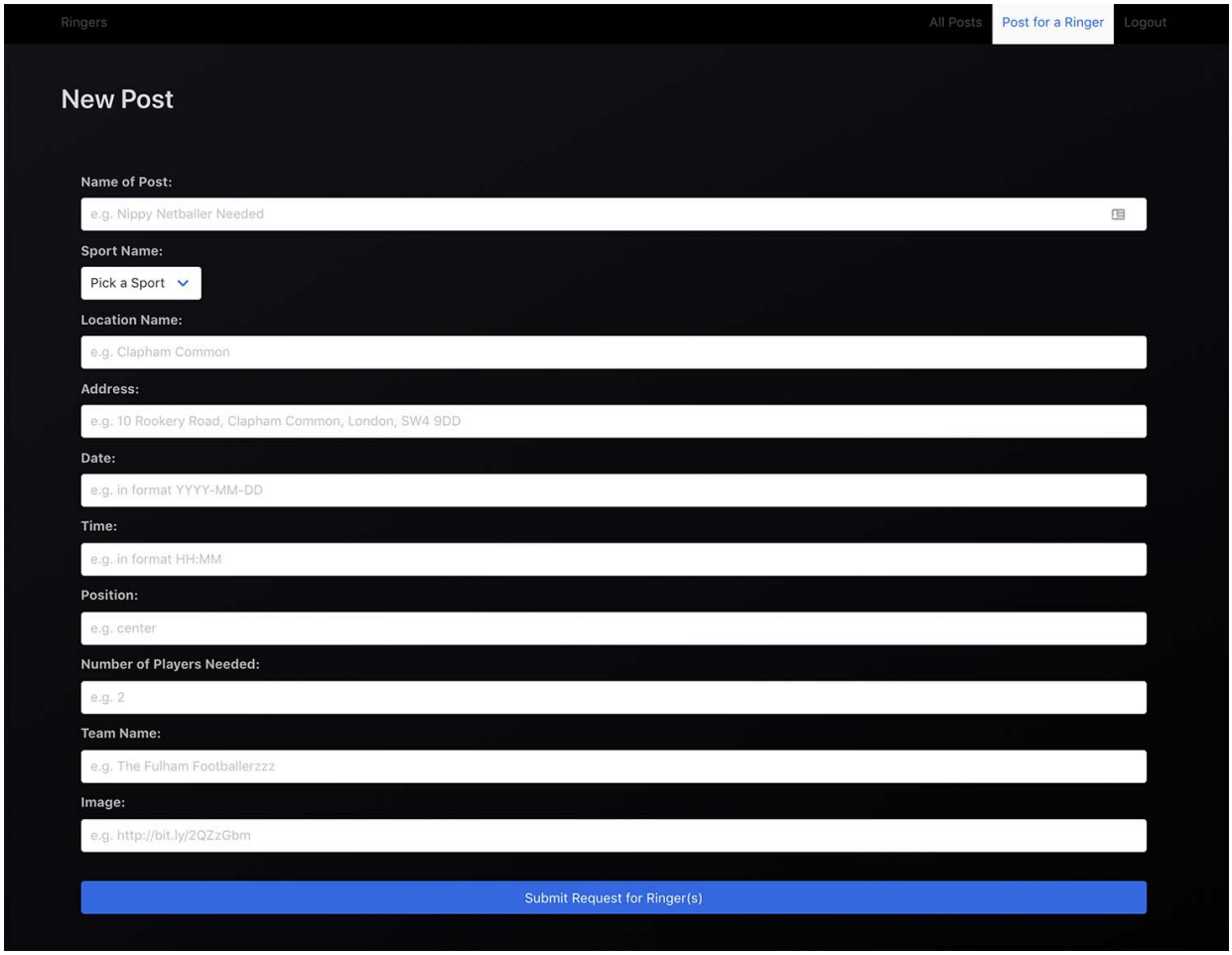
- navigation bar with different page links when signed in/out
- All posts page that shows all ringers needed with the date time and location
- Users can log in to create, edit and delete their own posts (add screenshots of index and show page, create)

Website Architecture

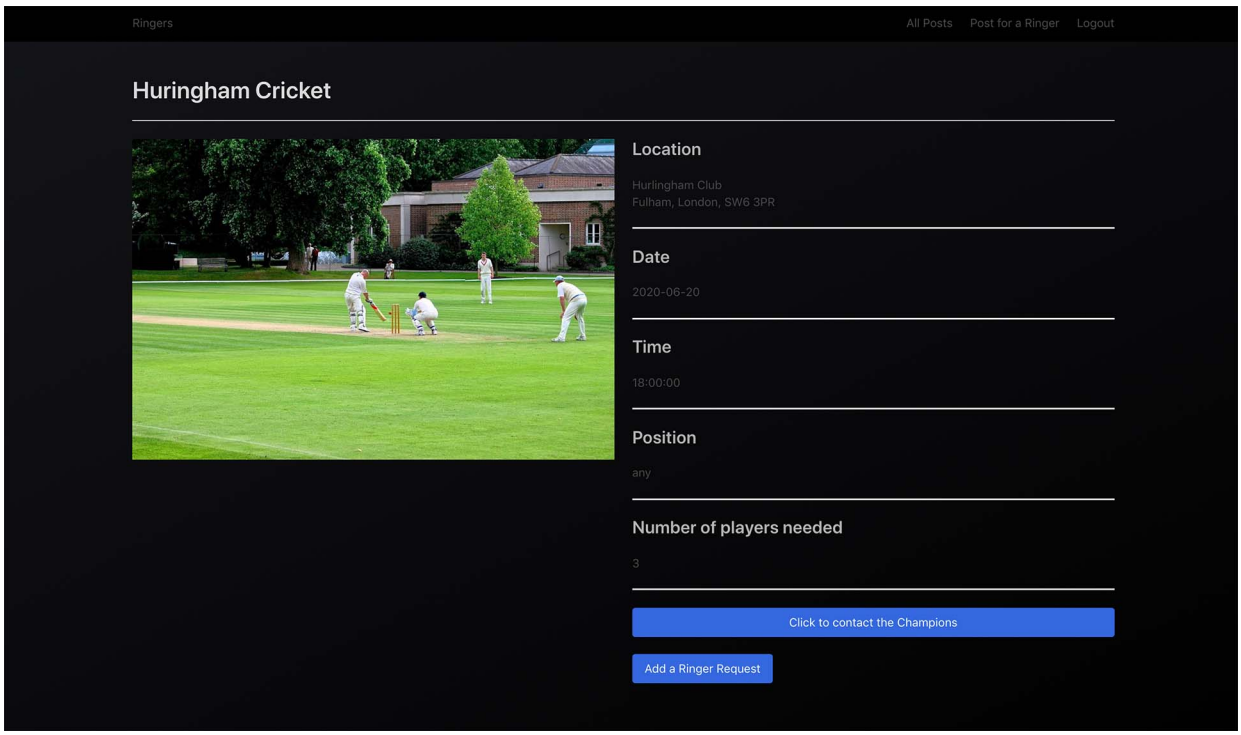
(Home Page at top of page)

Index Page to follow when deployed 

New Post Form



Show Page



Approach

Day 1

One of the nice things about this brief was that we had the option to work in a group or individually. Project 3 was a group project and I wanted to check that I was able to go through the all motions myself before leaving the course so I opted to go it alone.

We were told this should be the most ambitious project to date so I spent way too long brainstorming and deciding on what to build. But came up with lots of ideas which I will likely try and build after the course.

Day 2

After discussing my ideas with a few people in class, I decided to go with Ringers as it seemed most fitting with the CRUD functionality and the MVP seemed most achievable in the timeframe.

I started building the backend. I found it pretty inflexible, for example, when I needed to add or change the name of the fields in the model this caused errors and time fixing the errors. Initially set me back so far with time that I started the build from scratch.

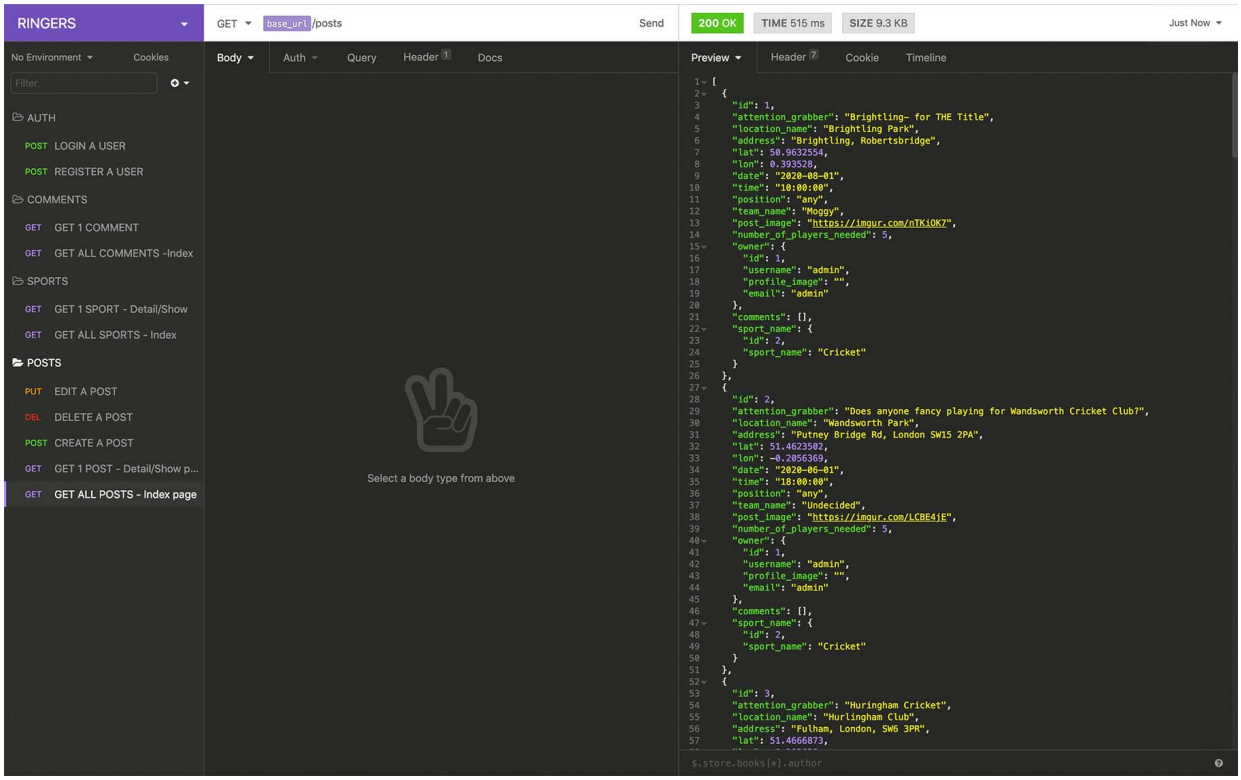
Created models, serializers and hooking up the urls.

I liked that Django had it's own inbuilt version of Insomnia.

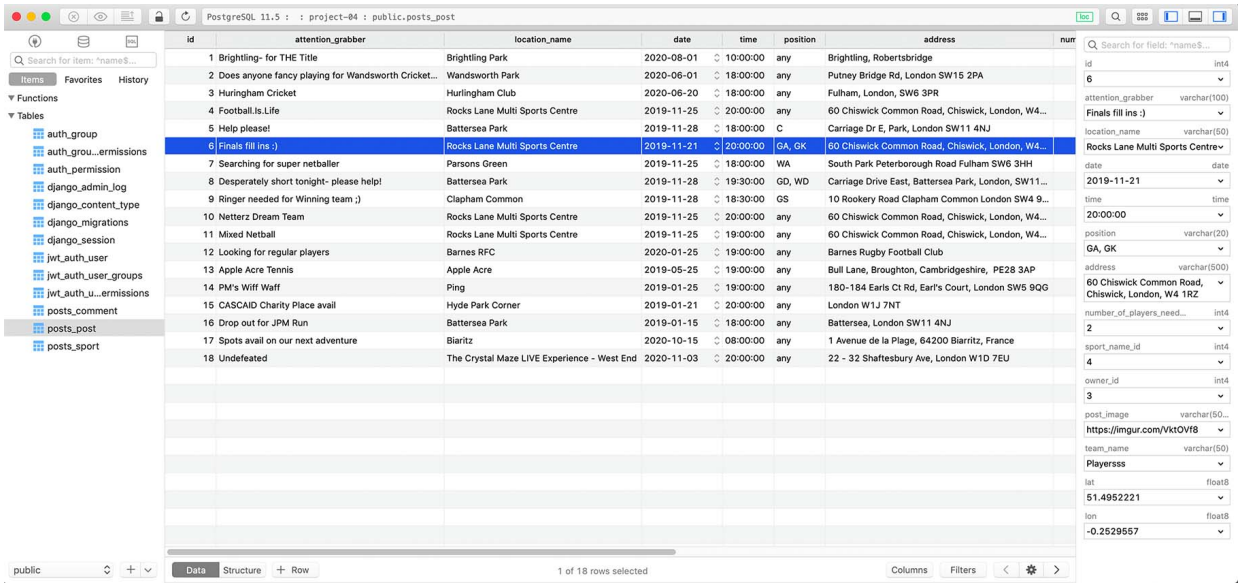
Day 3

I used insomnia and table plus to check everything was working on the back end.

Insomnia



TablePlus



Added nesting to the db as we had in the example in class, later I realised this wasn't the best way to do this so re wrote this on day 4 & 5.

I found making migrations particularly difficult.

Day 4

Added seeds.Found reseeding difficult. Lived in fear of accidentally deleting everything. Added edit and more fields to the default user model. Added forms and error handlers

Day 5

Added a front end. Feels a bit foreign having not looked at React for a couple of weeks. Added Bulma styling initially as worried about for speed but keen to do my own CSS Deliberate over

this for 24 hours. Should have added forms after adding Bulma.

Day 6

Work on Authentication. Toggle nav bar so it only shows relevant when logged in/out. Delete and edit button only for user who created.

Day 7

Attempted to hook up assets file to show images then learnt it is best practise to host externally so used imgur. Tried saving images for seeds in assets files then had a CORBs issue.

"Cross-Origin Read Blocking (CORB) blocked cross-origin response with MIME type text/html. See for more details."

Test and fix bugs. Work more on last minute styling.

Takeaways

As it happened this exposed the gaps in my knowledge. I spent too much time adding elements to the back end that I didn't get as far as needing on the front end. I didn't leave nearly enough on the front end so struggled to finish on time.

Overall, I was really impressed by the speed in which it was possible to create the back end using Django. There was lots of setup required (creating the virtual environment, running PostgreSQL, installing Django Rest Framework etc.) and it was important to complete the steps in order but I think I will become more comfortable with that over time. Initially I was thinking of adding comments so built this into the back end and started building it in the front end but thinking about it - it didn't make sense users to be able to comment- better to force them to email.

All in all I enjoy using python as the code looks much clearer without the brackets and with the indent format. Also loving the additional built in functionality. I found it frustrating using Django as it was quite rigid and the setup is very specific -didn't allow any room for error for a junior so I was frustrated by how long something that seemed like following a recipe on paper, took me.

I need to focus on keeping the MVP super simple and adding in styling much earlier so I can visualise the priorities.

Challenges Overcome

- Setting up in Django
- Linking to the front end
- Seeding and migrations
- Toggling the navigation bar
- Manipulating Bulma (responsive design)
- Allowing only hosts to edit and delete their posts
- Adding contact email links

Challenges Still To Overcome

- Using Imgur to show images
- Pre populating the edit form
- Getting the styling to a place I'm happy with

Future Improvements

- Add Mapbox to visually locate events nearby
- User can search for events by sport, location, date, time
- Team pages with scores and stats
- Player ratings with recommendations, positions, availability etc.
- Book into leagues - payment system
- Adding tests
- Sending out alerts

---

sophieturnell.com
linkedin.com/in/sophieturnell/
