FULL STACK FRAMEWORK WITH DJANGO PROJECT BRIEF

Congratulations on making it this far!
Now for the final project, you're going to build a web application, and you will draw on the knowledge and skills you have gained as a result of completing lessons in all nine modules. You can either choose to use the example brief below, or you can use your idea for the website.

BUILD AN ISSUE TRACKER
Now that you're a full-fledged web developer you've decided its probably time for you to start your very own cool, modern startup, offering the extremely awesome UnicornAttractor web app to your users. It's really amazing, but we don't care about it at all in this project. The exciting thing is the business model that you've decided upon – you chose to offer the service and bug fixes for free, but ask for money from your users to develop additional features.

The primary entity in the Issue Tracker is a ticket describing a user's issue, and similar to Github's issue tracker, you should allow users to create tickets, comment on tickets, and show the status of the ticket (e.g. 'to do,' 'doing,' or 'done'). As mentioned, issues come in two varieties – 'bugs' (which you'll fix for free, eventually), and 'features' which you'd only develop if you're offered enough money. To help you prioritize your work, your users will be able to upvote bugs (signifying 'I have this too'), and upvote feature requests (signifying 'I want to have this too'). While upvoting bugs is free, to upvote a feature request, users would need to pay some money (with a minimum amount of your choice) to pay for your time in working on it. In turn, you promise always to spend at least 50% of your time working on developing the highest-paid feature.

To offer transparency to your users, you decide to create a page that contains some graphs showing how many bugs or features are tended to on a daily, weekly and monthly basis, as well as the highest-voted bugs and features.

Add any additional pages that would help you attract users to the Issue Tracker (and have them pay you well). To make the users participate as much as possible in your online community, make sure that your UI/UX is sublime. Feel free to add additional features, such as a blog, extra perks for active participants, etc.

If you want to have some more fun with this, feel free also to add pages describing your fictional UnicornAttractor application.

And of course, as this project is going to be the lifeblood of your company, it's essential that new developers that join the company will be able to get up and running as quickly as possible. Documentation is the best way to achieve this.

CREATE YOUR OWN PROJECT
If you choose to create your project outside the brief, the scope should be similar to that of the example brief above. If you want some ideas, please ask your mentor for advice and direction.

FULL STACK FRAMEWORK WITH DJANGO PROJECT GUIDELINES

Use the following guidelines when developing your project:

Build a web app that fulfils some actual (or imagined) real-world need. This can be of your choosing and may be domain specific.

Write a README.md file for your project that explains what the project does and the need that it fulfils. It should also describe the functionality of the project, as well as the technologies used. Detail how the project was deployed and tested and if some of the work was based on other code, explain what was kept and how it was changed to fit your need. A project submitted without a README.md file will FAIL.

The project must be a brand-new Django project, composed of multiple apps (an app for each reusable component in your project).

The project should include an authentication mechanism, allowing a user to register and log in, and there should be a good reason as to why the users would need to do so. e.g., a user would have to register to persist their shopping cart between sessions (otherwise it would be lost).

At least one of your Django apps should contain some e-commerce functionality using Stripe. This may be a shopping cart checkout, subscription-based payments or single payments, etc.

Include at least one form with validation that will allow users to create and edit models in the backend (in addition to the authentication mechanism).

The project will need to connect to a database (MySQL or Postgres) using Django's ORM

The UI should be responsive, use either media queries or a responsive framework such as Bootstrap to make sure that the site looks well on all commonly-used devices.

As well as having a responsive UI, the app should have a great user experience.

The frontend should contain some JavaScript logic to enhance the user experience.

Whenever relevant, the backend should integrate with third-party Python/Django packages, such as Django Rest Framework, etc. Strive to choose the best tool for each purpose and avoid reinventing the wheel, unless your version of the wheel is shinier (and if so, consider also releasing your wheel as a standalone open source project).

Make sure to test your project extensively. In particular, make sure that no unhandled exceptions are visible to the users, under any circumstances. Use automated Django tests wherever possible. For your JavaScript code, consider using Jasmine tests.

Use Git & GitHub for version control. Each new piece of functionality should be in a separate commit.
Deploy the final version of your code to a hosting platform such as Heroku.

PLAGIARISM
It is each student's responsibility to ensure that when they include (directly or indirectly) the work of others that this contribution is fully and adequately acknowledged and documented.
You are encouraged to ask your mentor/tutor for advice about your project work but note that your project code should not include any code written by others unless it is explicitly credited to them with a comment above. Any such code provided by your mentor/tutor would not contribute to your assessment. Failure to attribute credit to code that isn't yours will be considered as plagiarism and will result in a failing grade.