Final Project for Stream Three

In this unit the students are given the guidelines for their stream three project

**PROJECT BRIEF**

The project guidelines are as follows:

Congrats on making it this far!

Now for the final project you’re going to have to build a web app that will include as much of the topics that we’ve covered throughout the three streams of the fullstack course as possible.

* **Build An Issue Tracker**

Now that you’re a full fledged web developer you’ve decided it’s probably time for you to start your very own cool, modern startup, offering the extremely awesome UnicornAttractor webapp to your users. It’s really really amazing, but we don’t care about it at all in this project. The interesting 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.

To manage the tracking of bugs and feature requests, you decided to create an Issue Tracker that will allow your users to submit and track any issues (bugs or feature requests) related to using the UnicornAttractor.

The basic 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 prioritise 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 to always 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 and 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, additional perks for active participants, etc…

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

And of course, as this project is going to be lifeblood of your company, it’s important 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.

* **Choose Your Own Project (Recommended)**

If you choose to create your own project, the scope should be similar to that of the example brief above, or the we\_are\_social project from earlier in this stream. If you want some ideas, please ask your mentor for advice and direction.

**PROJECT GUIDELINES**

* Build a web app that fulfills some actual (or imagined) real-world need. This can be of your own 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 fulfills. 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 off other code, explain what was kept and/or 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 in order to persist their shopping cart between sessions (otherwise it would be lost).
* At least one of your Django apps should contain some kind of e-commerce functionality using Stripe and/or Paypal. This may be a shopping, or subscriptions, 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 an SQL database using Django’s ORM, or to a Document-Oriented database (e.g. MongoDB) using pymongo.
* 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 Disqus, 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, in 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.

**ADDITIONAL ADVICE**

* We advise that you create wireframes before embarking on full blown development (you can use Balsamiq or any other tool, including just pen and paper)..
* Incorporate as much as you have learned in our lessons.
* The site can also make use of CSS, Javascript and Python libraries/frameworks, just make sure you maintain a clear separation between the library code and your own code
* Please refer back to your mentor to get constant feedback as you progress
* Don’t forget to use your mentor’s help. Your mentor can act as:
  + Your client
  + Your technical consultant

Good luck and enjoy!