Part 10 & Part 11

Introduction

Self Evaluation: Mastery / Progressing / Needs Improvement

Due date: 2021-11-12

Completion: 2021-11-13

GitHub: https://github.com/ZDGharst/FullStackOpen (part10)

https://github.com/ZDGharst/full-stack-open-pokedex (part11)

Part 10 – React Native

In this section of the course, I create an iOS/Android/mobile app using React Native. React Native is a framework to develop native Android and iOS apps using JavaScript and React. The application builds down to a native application specific to the device that is being deployed to. You could use a framework such as Cordova that allows development to happen with HTML, CSS, and JavaScript that is run in a webview; however, you lose access to native features, and it isn't very performant. React Native's main feature is the speed of development: any React engineer for web development can quickly transition and learn the native environment when React Native is used.

There are many options for running the mobile app in this course. For this, we use Expo to build the React Native app into a mobile app. You can run the build in a web browser, through an Android or iOS emulator, or my preferred method which was to use the Expo Go app on my Android device and run it through that app on my device itself.

The course covers configuring the project with needed tools, using native components to create the frontend and actions to perform functionality such as presses, and testing and debugging. I add style to the application via inline styles and themes, I add pages via routing, create forms for logging in, validate forms and handle their state, implement infinite scrolling and cursor-based pagination, and operate on the backend endpoints with GraphQL queries and mutations.

This part of the course covers a lot of material; I spent around 30 hours on this part alone. These past few parts are becoming quite time intense, as there is a lot of material to cover in only a single part.

Additional notes can be found in the Part 10 README.

Part 11 – CI/CD

This entire part of the course is about Continuous Integration and Continuous Delivery. Topics that I cover are branching etiquette and procedures, pull requests and merges, packaging, building and deployment, and rules and pipelines. I cover a few of the important principles: CI/CD is not the goal, the goal is better and faster software development with fewer preventable bugs. I learn about covering cases that something that is “creative” and correct should probably still fail safely if it’s unexpected, how the same thing should happen every time in the same order, and code should always be deployable with the latest deployed version being easily known and changeable. The course talks about both Jenkins (self-hosted CI/CD system) and GitHub Actions (cloud-based CI/CD system) and when to pick which type. I picked another language other than JavaScript to examine their CI/CD system(s). I chose Go for the built in tools from the standard library such as the formatter, builder, and tester.

Part11b gets me started working with GitHub actions. I created a pipeline yaml configuration for a workflow which runs the linter, the build, and the test to make sure nothing goes wrong. Along each step, I allowed the workflow to run then I fixed the issues that came up from the lint and tests that happened. I also created End-to-End testing (E2E) using Cypress to add that test to the workflow.

I also learn what a deployment system should do and what could go wrong if it isn’t covered by your deployment server. I set up a deployment server to Heroku, and set up a workflow so that after a commit is made on the master branch, the new development version is pushed to Heroku. I purposefully broke the application to see the rollback features in action.

Part11d is called “Keeping Green”, and the part is about automating workflows on pull requests (blocking pull requests without approval), reasons for pull request, what to focus on during a pull request, and configurations to make the most out of the pull request process such as testing and linting. Versioning, with both semantic versioning and hash versioning, is covered as well. I created a workflow such that when a pull request is merged into master, the version number is automatically changed. I also add a condition on some pull requests such that they may not be deployed to production under certain criteria.

The last part of the entire course is *Part11e* is short, but it covers how to make issues visible and underestandable, setting up notifications, metrics that can be tracked, and automating periodic tasks.

Additional notes can be found in the Part 11 README.

Deliverables

For a completed list of exercises, please view the commits on the master branch here. There is a history browser, diff viewer, and the git messages explicitly define exercises.

In addition to the exercises in this module, exercises 9.22-9.27 were completed in this section as overflow from the previous section.

The exercises in *part10d* were not completed due to time constraints. These will still be completed.

Supplementary Material

No supplementary material in this module.