Case Study: Real Web Apps

Q1. Introduction to Redash

For your final assignment, you will investigate two real-world web applications, and answer questions about how they work. You'll get to see for yourself the similarities and differences between the applications you've worked with so far in this course and the codebases for living applications with thousands of users.

Redash

Redash is an open-source browser-based data tool. From the description:

> Redash is designed to enable anyone, regardless of the level of technical sophistication, to harness the power of data big and small.

Redash is a large, active Flask application with years of development.

Open the repository: **[Redash on Github](https://github.com/getredash/redash)**

Instructions

- Follow the link to read more about the application
- Answer the questions about the application

Q2. Redash: History

Browse the Github interface or use git log to answer the following questions:

Q2.1. When was the first commit?

Q2.2. How many commits have there been?

Note: the codebase is alive, so this number will change as more commits are added. Enter the number of commits right now, you'll still get points if the number changes.

Q2.3. How many different committers?

Q2.4. How many issues have been opened? How many are currently open?

Q2.5. How many pull requests have been merged?

Q3. Redash: Size

Clone the Redash repository. Navigate to it in the terminal or VSCode and answer the following questions.

Tip: Do not try to count things manually! Use command line tools like ls, tree, wc, find, or Powershell's [Measure-Object](https://superuser.com/questions/345849/how-to-count-all-the-lines-of-code-in-a-directory-recursively-on-windows) to do the counting. (See [this page for tips](https://devconnected.com/how-to-count-files-in-directory-on-linux/))

Q3.1. How many files are there in the project?

Q3.2. How many lines of code?

Q3.3. What file lists the project's dependencies?

Note: Redash has lots of dependencies for different kinds of installations and users. Just include the core dependencies.

Q4. [Optional] Run Redash

As an optional task, install the dependencies and get Redash up and running locally. The steps to get it running can be tricky, so ask for help in Discord if you get stuck.

Q5. Ghost: History

Ghost

Ghost is an open-source blogging and newsletter platform, built on Express.

> Turn your audience into a business. Publishing, memberships, subscriptions and newsletters.

It's one of the go-to options for building a newsletter or blog. You can also learn a lot about how a large, widely-used Express app works by reading the source.

Open the repository: **[Ghost on Github](https://github.com/TryGhost/Ghost)**

All the following questions are about Ghost.

Q5.1. When was the first commit?

Q5.2. How many commits have there been?

Note: the codebase is alive, so this number will change as more commits are added. Enter the

number of commits right now, you'll still get points if the number changes.

Q5.3. How many different committers?

Q6. Ghost: Size

Clone the repository: **[Ghost on Github](https://github.com/TryGhost/Ghost)**

Navigate to it in the terminal or VSCode and answer the following questions. Tip: Do not try to count things manually! Use command line tools like ls,tree, wc, and find to do the counting. (See this page for tips)

Q6.1. How many sub-projects are there within the ghost project?

Hint: how many subdirectories are there in the `ghost` directory?

Q6.2. What file lists the dependencies for ghost/core?

Note: there are several subprojects within the Ghost repository. Find the listing for the dependencies within `ghost/core`. Enter the full path to the file.

Q7. The Ghost Application

Q7.1. Initialization

In the projects you've seen so far in class, there is just one app.py or app.js file, and there is no application code dedicated to loading modules and starting the application. Ghost has lots of different pieces, so startup is more complicated.

Look at `ghost/core/core/boot.js`. What are the steps of the initialization process (Hint: Look for the comments that say "Step N")

Q7.2. External API

Ghost uses the Stripe API to handle customers and payments.

The file `ghost/stripe/lib/StripeAPI.js` exposes a limited part of the full Stripe API to the rest of the Ghost application. Looking at the type definitions at the top of the file, which 6 Stripe resources does Ghost use?

Q7.3. Link Redirects

Ghost has a link redirection feature similar to the first assignment: the url-shortener. This enables tracking links that are clicked from newsletter emails.

Tracing through the code:

- `ghost/email-service/lib/email-renderer.js` replaces urls before emails are sent out
- it uses services to do link replacement and add tracking urls specific to each recipient of the email
- when a link in an email is clicked, the redirection logic is handled in `ghost/link-redirects/lib/LinkRedirectsService.js`. It looks up the url to redirect to, tracks the click event, and responds with the redirection.

What line of code in the `LinkRedirectsService.js` actually issues the redirect?What HTTP status code does Ghost use when sending this redirect? You may need to look up `redirect` method in the [Express docs](https://expressjs.com/en/api.html#res.redirect).

Q8. [Optional] Run Ghost

As an optional task, install the dependencies and get Ghost up and running locally. The steps to get it running can be tricky, so ask for help in Discord if you get stuck.