Newt /n(y)o͞ot/ - small slender-bodied amphibian. cute but poisonous (toxic) .

Phase 1 - Progress Report

Optimistic Newts

Overview

Throughout phase 1, our team has continued to work effectively towards fulfilling our Specification and moving our program to a Web App. Our team has continued to work well in communicating amidst ourselves and in dividing the work between us. :)

Specification: Updated

Since Phase 0, we've been hard at work implementing the remaining User Stories, and making the transition to a WebApp.

Remaining specification:

- Interacting with Conversation Queues for sorting conversations based different algorithms
- Changing User IDs to be unique integer IDs for easier sorting/management in the database

Along the way, we reworked:

- Request and Response Models for requests between the controllers and interactors
- Our Data Access Objects for interacting with the persistence layer
- Our controller layer to work with Spring Boot and our updated interactors

As a User, I want to be able to...

- Create, Log in and Log out of an account
- Safe and secure user and password protection
- User profile personalization and editing
- Browse through and communicate with other users
- Find, join, engage and contribute to interesting conversations and meet other users with similar interests
- Discover new topics which may interest me through recommendations, and filter-out topics that I am not interested in seeing
- Look for conversations both in my city and internationally to meet people with different perspectives
- Find, add, and remove people as friends

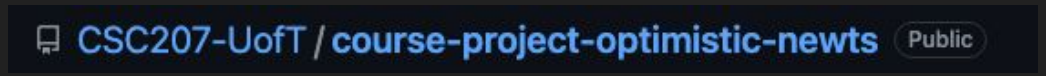
Web App

After phase 0, we decided to transition our project to a web app using Spring Boot and PostgreSQL!

Things we worked through:

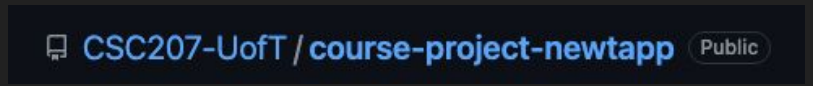
- Moving to and configuring Spring Boot
- Installing PostgreSQL on each of our systems (there were some entertaining roadblocks)
- Implementing and configuring a database for our application using PostgreSQL
- Adjusting our Presenter and Response Model to let Spring Boot take care of it
- Linking our entities to our database so Spring Boot can handle Data Access for us

Along the way, we created a new repository for our project, in order to simplify the Spring Boot configuration.



: Old

New :



(no longer optimistic)

Design Decisions

Along the way, we made some important design decisions.

- ## Implementing a Web App

  A key design decision our group made was to focus our Social Media Platform towards a Web Application, to maximize the reach and scope of our app. The technologies we chose to assist us in this are:

  Spring Boot - a Java-based framework for our Back-end API server, responsible for handling client requests, as well as reading and writing from a Database

  PostgreSQL - a relational database management system - the database itself

- ## Unique Usernames and Integer IDs

  We initially wanted to have usernames be cosmetic and user ids to be what distinguishes users from each other. But after trying a few different approaches of this, we decided it would be easier to enforce that each username is unique and use user ids for mostly backend purposes including being the Primary Key of our users table in the database.

Design Patterns

We have done our best to ensure our program uses design patterns where applicable to solve problems effectively and efficiently in our code. Here are some examples:

- ## Facade Design Pattern:

  The Facade design pattern provides a simple interface to a complex subsystem, containing many moving parts.

  Conversations and Users have a lot of small interactors, so to simplify our code, we created “manager” facade classes for Conversations and Users to delegate calls to specific interactors as needed, rather than having all of these small interactor methods together in one class.

- ## Strategy Design Pattern:

  The Strategy Design pattern facilitates defining family of algorithms in separate classes, while still making their objects interchangeable.

  An important feature of our app is recommending new conversations based on a User’s interests. So, to sort through which Conversations to recommend we have a few different algorithms in mind. We employed the Strategy Design Pattern for this, creating a ConversationSorter interface then creating implementing classes for each of our specific sorting algorithms. This way they are easily interchangeable.

Open Questions

A few things we're still considering :

- Are 32 bit integers sufficient for unique User ids?

We noticed in Evan's "Intro to Web Apps" example that Java's long type was used for ids. What are the pros and cons to using long instead of int for ids?

- Using Database-set IDs vs setting our own database ids

We found PostgreSQL's implementation of int IDs to be very useful for database organization, however, some sources on the web we came across suggested against this because using Database generated IDs delegates an important aspect of the program's domain to third party software. Should we consider handling userIDs ourselves? What considerations must we make if we choose to do this?

- Making our API publicly available

We hope to have our API publicly available for Phase 2. Where can we host our app? What changes need to be made to our app and database to support running remotely? What security concerns must we address in making our API public?

Looking Forward

We’re all enjoying seeing the fruits of our labour begin to form a working application, and are excited to see where Phase 2 takes us!

Here are some ideas of the next steps in our development:

- **Finalizing our back end**
- **Deciding how we want to handle our front end**
- **UI/UX Design choices**

:

:

:

Bringing our app live!

DEMO

THANKS *for* listening :)