Web Programming Group Coursework Extra Feature Report

Group 20
Normantas, Lukas
Pogonowski, Andrzej
Kobalasingam, Ageeban

Website Address: http://webcw-g20.apps.devcloud.eecs.qmul.ac.uk/
GitHub: https://github.research.its.qmul.ac.uk/ec14099/groupcw_web

Admin Details --
Email: admin@admin.com Password: @dmin098

Sample User Details (Frontend)
John Doe - Email: 1@1.com:123
Peter Doe - Email: 2@2.com:123
Harry Doe - Email: 3@3.com:123

The entire front end is written in Vue.js, a progressive JavaScript Framework. Vue.js has a very simple structure that is easily extendable by using Vuejs plugins and other JavaScript libraries. Vue.js uses a component structure model, which allowed us to decouple elements into multiple atomic components. The components are able to talk to one another using props and events, this allowed us to keep the menu component separate from the article render component, while still maintaining connectivity between them. Vue provides its own build and compile tools that takes .vue files and compile them into minified Javascript code that then gets sent over by Django. This is why the Vue code is separate to the Django python code. The front end communicates entirely via asynchronous requests using the Axios (AJAX) JavaScript library, this means that our Newspaper is a single page application. The application only refreshes if the user logs out, in order to clear all the session variables. Apart from Axios we also use Vue-Sessions in order to keep the state of the user. Cookiesjs is used to grab the CSRF token. We also use VueBootstrap, which is a Vue plugin that allows us to implement Boostrap components and CSS easily through Vue syntax. Using Vue and VueBootstrap we were able to do some front end validation. This means that the user (unless he/she modifies the Javascript code) cannot submit an invalid form. The Django Web API back end handles data validation and sanitation. Lastly we use VueAwesome, which is an icon library for Vue.

The front end is completely decoupled from django, apart from passing the Javascript files to the client. In order to implement front end to back-end communication we used the django-rest framework to implement a Web API. We used the Django rest framework token authentication. This means that the user only logs in once and then uses the token stored on the front end, in order to communicate to the server. The web api uses @api_view decorator in order to determine what actions can be taken and use @permission_class decorator to determine if the action can be called by an anonymous user. Actions which require the user to be logged in are protected using that decorator, and we use Django rest token authentication middle ware in order to reference the user, when we filter objects by the user object.

We also added a Data Layer (data.py) in order to separate the data filtering from the view logic. Lastly we added some general tests to test the data layer. That way we were able to verify that our data layer is returning the correct information.

The application is deployed on Openshift and uses MySQL as its database.