Gather Documentation

Table of Contents

1. Project Description 1
2. Process 1
3. Requirements and Specifications 2
   3.1. Use Case Diagram 2
   3.2. Use Cases 3
   3.3. User Stories 4
4. Architecture and Design 6
   4.1. System Architecture 6
   4.2. Application Architecture 7
      4.2.1. Model 7
      4.2.2. Controller 8
      4.2.3. View 9
   4.3. REST API 9
   4.4. Sequence Diagrams 10
   4.5. Framework Reflection 12
5. Reflections and Lessons Learned 13

1. Project Description

Gather is a web application that allows both individual hobbyists, hobby groups, or general interest groups to display and filter their specific interests in the application based on location and time. In this way, people can easily meet new friends with similar interests, especially when they are new to an area. Established hobby and interest groups can also attract new friends to join their group.

2. Process

Our team followed the Extreme Programming (XP) process as studied in Software Engineering I (CS 427) at the University of Illinois. During every iteration the development team divided into the three pairs and each pair selected a number of user stories to work on as part of the planning game. The team members in each pair met frequently during each iteration to complete tasks and then discussed their progress during one or more weekly full team meetings to aggregate all the work with other pairs.

We followed the XP process fairly closely with one exception, we did not utilize Test Driven Development (TDD) across our entire project. There were some elements that were developed using TDD, but our cycle for many features was to build, test, ensure features and tests aligned with the requirements, and then refactor. This cycle is reflected in many of the commits throughout our repository.

To streamline our process, our team spent a good deal of the early iterations developing and refining our workflow. We utilized Bitbucket for our Git repositories and also used their issue tracking system. Additionally, we used our class

provided VM and set up our own Jenkins server to track our Bitbucket repository. Jenkins would automatically poll our repository, run our tests and then deploy our app to 1 of 8 different production or development servers based on the branch name. This workflow was a great boon to our development and is summarized in the following graphic:

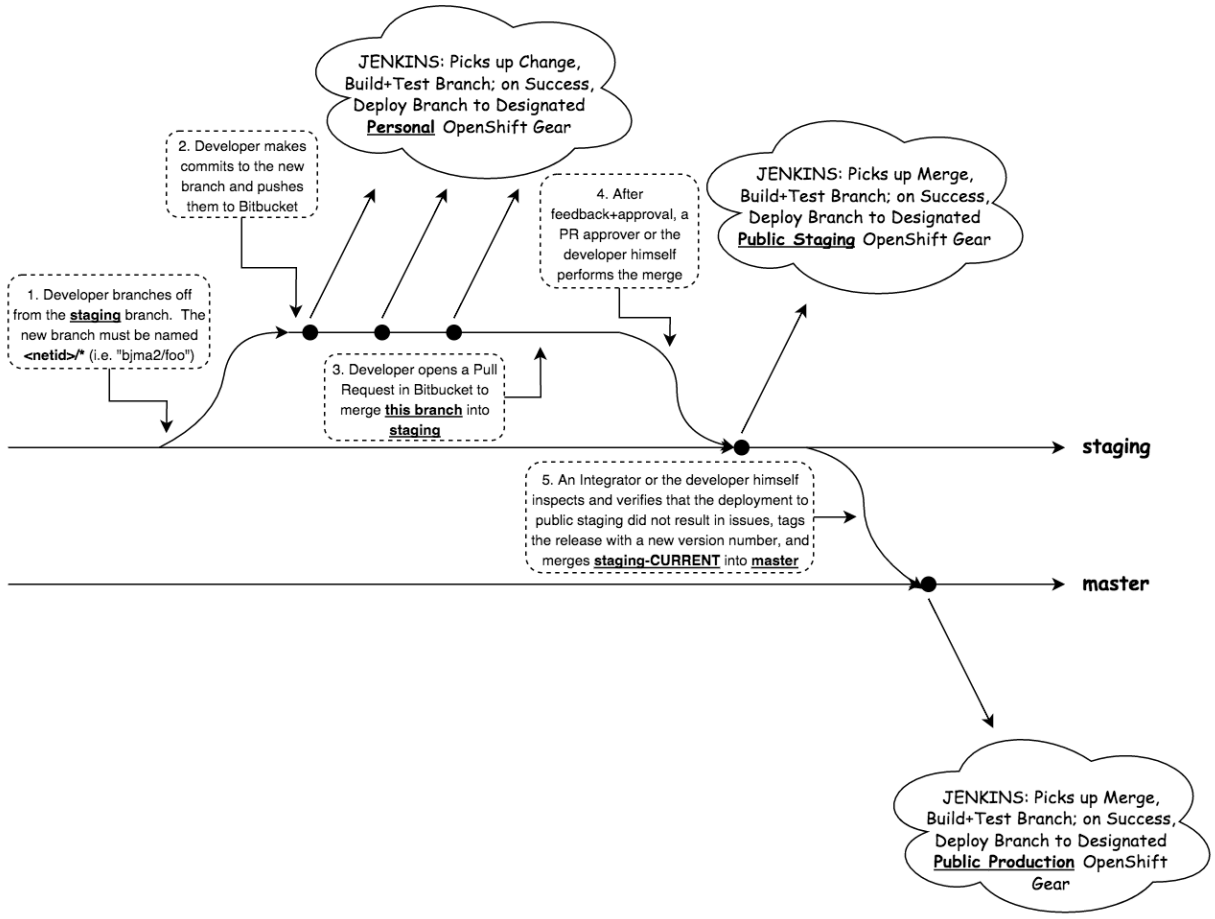

Figure 1 - Developer Workflow

3. Requirements and Specifications

3.1. Use Case Diagram

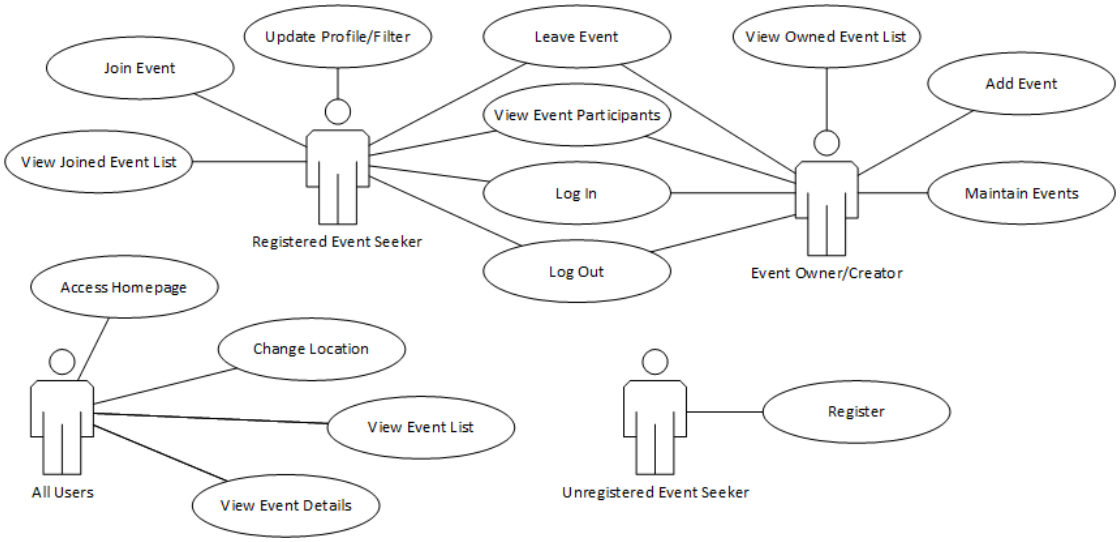

Figure 2 - Use Case Diagram

3.2. Use Cases

#	Use Case Title	Actor	Task-level goal
1	Access Homepage	All Users	A user accesses the system and is presented a map view displaying the events in their local area.
2	Register	Unregistered Event Seeker	An unregistered user elects to register. Providing a username and password. Registration can be successful or unsuccessful and the appropriate notification is provided.
3	Login	Registered Event Seeker, Event Creator/Owner	A registered user elects to log in. They either successfully log in and are presented a map view displaying the events in their local area that match their default filter, or they fail to login due to user/pass issues and are notified.
4	Change Location	All Users	A user can change the location they are searching by either moving the map, or entering a new zip code.
5	View Event List	All Users	A user can view a list based presentation of the events. The list can be sorted by distance from user, alphabetical, time, or rating...
6	View Event Details	All Users	A user can view the detailed information about an event, as well as see any ratings and reviews left by other users. A user can also view the upcoming sessions of an event in a calendar view.
7	Update Profile/Filter	Registered Event Seeker	A registered user can change their current event interest profile (category, time, location, ...). Once applied, they are returned to the map view displaying the appropriate events.
8	Join Event	Registered Event Seeker	A registered user can elect to join an event.
9	Leave Event	Registered Event Seeker, Event Creator/Owner	A registered user can elect to leave an event. They will no longer be listed as a participant and the event will not be listed in their "My Event List". If the member is an owner, they must appoint at least one co-owner before they can leave the event.
10	View Joined Event List	Registered Event Seeker	A registered user can view a list of events that they have joined.
11	View Created Event List	Event Creator/Owner	A registered user can view a list of even they have created/own.
13	View Event Participants	Registered Event Seeker, Event Creator/Owner	A registered user can view the list of other users who have joined a particular event.
15	Add Event	Event Creator/Owner	A registered user can create a new event, providing a location, time, name and description. They will also decide what categories this event will fall under (multiple categories acceptable).
16	Maintain Event	Event Creator/Owner	An event creator/owner can edit the information for events they created. An event creator/owner can mark an event recurring or manually schedule upcoming occurrences. An event creator/owner can select other registered users to be added as co-owners for their event. An event creator/owner can remove an event, effectively deleting it.

25	Logout	Registered Event Seeker, Event Creator/Owner	A user will be able to sign out of the application so that their state is not remember in their browser session.

3.3. User Stories

Use Case #	User Story Description	Use Case Title
1	As a user, I want to be able to access the application homepage so that I can see what the app offers.	Access Homepage
1	As a user, I want the system to detect my location automatically so that I don't have to enter my address.	Access Homepage
1	As a user, I want to be able to enter my zip code so that the application can show me local events in case it could not detect my location automatically.	Access Homepage
1	As a user, I want to see a map of my location.	Access Homepage
2	As a user, I want to access the registration page so that I can register on the website.	Register
2	As a user, I want to choose my username and password and successfully register so that I can login to the app.	Register
2	As a user, I want to be stopped from successfully registering if I enter a username which is already taken or the password chosen doesn't meet the requirements so that I don't share username with someone else and my account is secured.	Register
3	As a registered user, I can log in to the system by entering a valid username and password.	Log In
3	As a registered user, I will be notified of a failed login due to incorrect username/password.	Log In
4	As a user, I want to change the location by navigating the map.	Change Location
4	As a user, I want to change the location by changing the zip code.	Change Location
4	As a user, I want to change the location and be able to see the nearby events either by changing zip code or navigating the map.	Change Location
5	As a user, I want to view a sortable list of nearby events so that I can easily find the events I am interested in based on certain criteria.	View Event List
5	As a user, I want to sort the event list by distance from user, name, time, rating, or category.	View Event List
6	As a user, I want to click on any event shown on the map so that I am able to see details of the events.	View Event Details
6	As a user, I want to select an event from the event list so that I am able to see details of the events.	View Event Details
7	As a registered user, I want to save a list of event categories I prefer so that the application automatically filters what it displays to only include these categories.	Update Profile/Filter

7	As a registered user, I want to save my zip code so I never have to enter it in case the system cannot automatically detect my location.	Update Profile/Filter
7	As a registered user, I want to save a default time window (as in X hours from now) so that the application automatically filters what it displays to only include events in my window.	Update Profile/Filter
8	As a registered user, I want to join an event, so that I am listed as a participant and the event is added to my joined event list.	Join Event
8	As a registered user, after electing to join an event, I want to be informed if I fail to join the event and why (e.g. the event was removed before I could join).	Join Event
8	As an unregistered user, if I want to join an event I am asked to register.	Join Event
9	As a registered user, I want to leave an event so that I am no longer listed as a participant and it doesn't show up in my event list.	Leave Event
9	As an event owner, if I choose to leave an event and there is no co-owner, I want the system to notify me that I must add a co-owner before I can leave the event.	Leave Event
10	As a registered user, I want to see all the events that I have joined so that I don't miss any.	View Joined Event List
11	As a registered user, I want to see a list of all the events that I own so that I can manage them.	View Owned Event List
13	As a registered user, I can view the list of other users who have joined a particular event so that I am better informed about who might be attending.	View Event Participants
15	As an event creator, I want to create a new event providing a location, time, name, description and categories.	Add Event
15	As an unregistered user, if I want to create a new event I am asked to register.	Add Event
15	As an event creator, after entering the details for a new event, I want to be informed if there is an existing event with the same name, location, and time. I expect to have to change one of these items otherwise I cannot create my duplicate event.	Add Event
15	As an event creator, I want to be informed if I have not entered sufficient information to create a new event.	Add Event
16	As an event owner, I want to remove an event I created/own, so that it is no longer displayed in the system and others are aware it may no longer be taking place.	Maintain Event
16	As an event owner, I want to be able to set a recurrence to an event that I have created so that I don't have to create a new event for every future occurrence.	Maintain Event
16	As an event owner, I want to be able to edit the details of my events so that I can convey the changes to event participants.	Maintain Event
16	As an event owner, I want to be notified if I enter insufficient information to update or change an event.	Maintain Event
16	As an event owner, if I cancel editing an event in the middle of change, the event should contain the original information that was there before I elected to make a change.	Maintain Event
16	As an event owner, I can add a co-owner to the event so that another person can manage the event alongside myself or take over completely.	Maintain Events
25	As a user, I want to be able sign out when I no longer want to use the app services.	Log Out

4. Architecture and Design

4.1. System Architecture

Gather is a Java web application, running on Apache Tomcat and hosted on RedHat Openshift. OpenShift is a Platform as a Service (PaaS) offering backed by Amazon Web Services that provides many features like integrated MySQL servers and Jenkins integration. These features are what initially drew us toward OpenShift, but as we continued development we ran into limitations that eventually stopped us from using them. As a result, we are simply using OpenShift's Tomcat server to host and serve our Java Web Application Archive (.WAR).

Gather utilizes the Spring MVC framework and Spring Data tools. To aid us in getting the project up and running quickly we used Spring Boot which reduces a lot of the onerous configuration tasks often associated with Spring applications. With Spring Boot and Spring Data, we were provided an embedded H2 database and object relational mapping courtesy of Hibernate.

For client technologies, we explored several frameworks including React.JS, but ultimately settled on moving forward without them. We did, however, utilize tools such as jQuery and Bootstrap to help us build our pages and interactions. We used Mapbox as our mapping platform alongside various tools made available by Google for geolocation.

The diagram below highlights this overall system architecture as well as how our development process was tied into it:

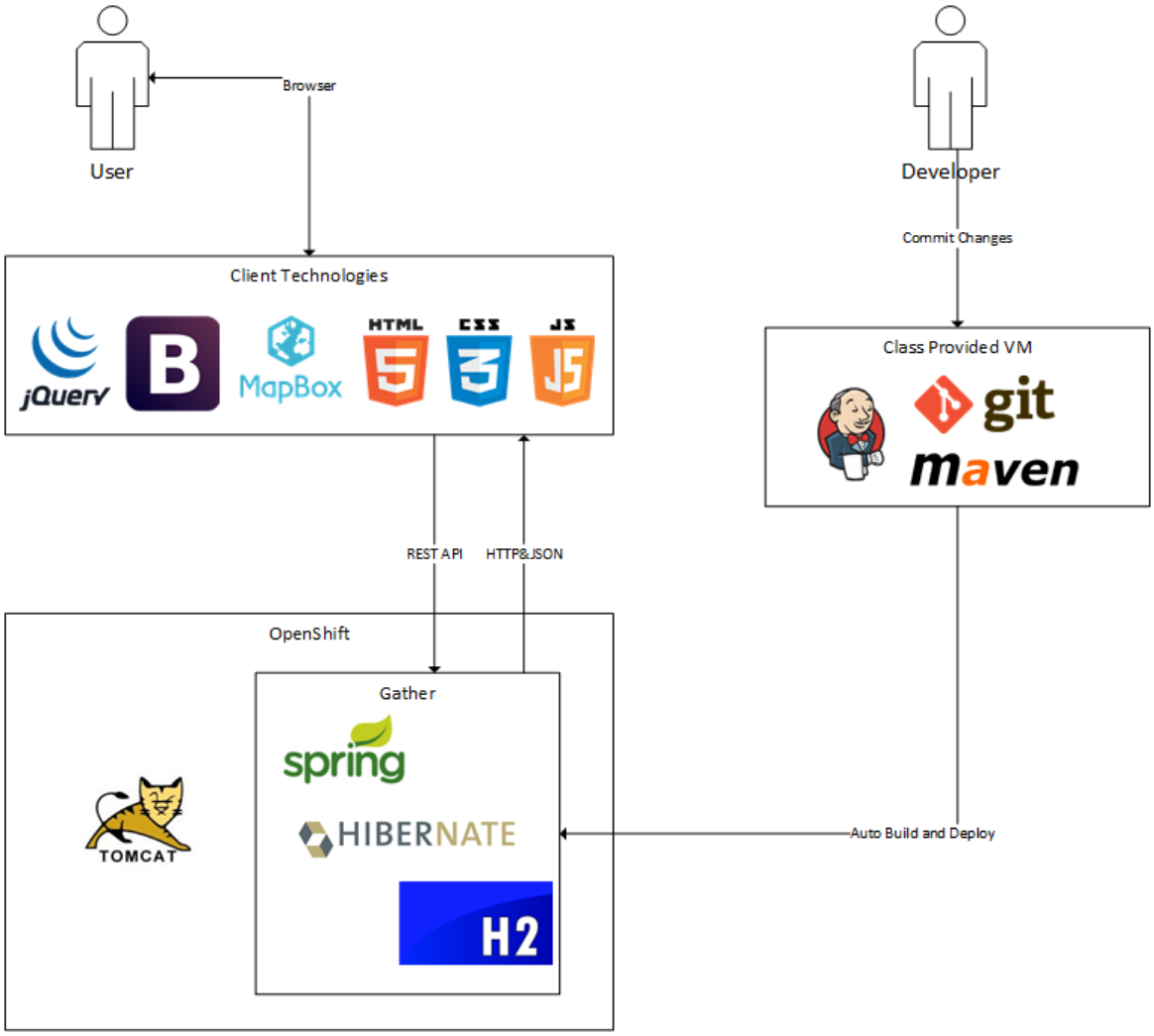

Figure 3 - Architecture and Flow

4.2. Application Architecture

Since we are using the Spring framework, our application utilizes the Model View Controller (MVC) architecture. Our views are constructed with HTML & JavaScript using the client technologies discussed above. The views communicate with our Spring based controllers using HTTP & JSON according to our REST API. The controllers will parse any incoming form data, validate the data, and then interact with the model to perform the desired function. Our model utilizes Java Persistence API (JPA) entity classes and Spring Data provided repositories. The controllers can interact with the entity classes directly, and the repositories are used to retrieve and store entities to the database. This overall architecture is depicted in the figure below:

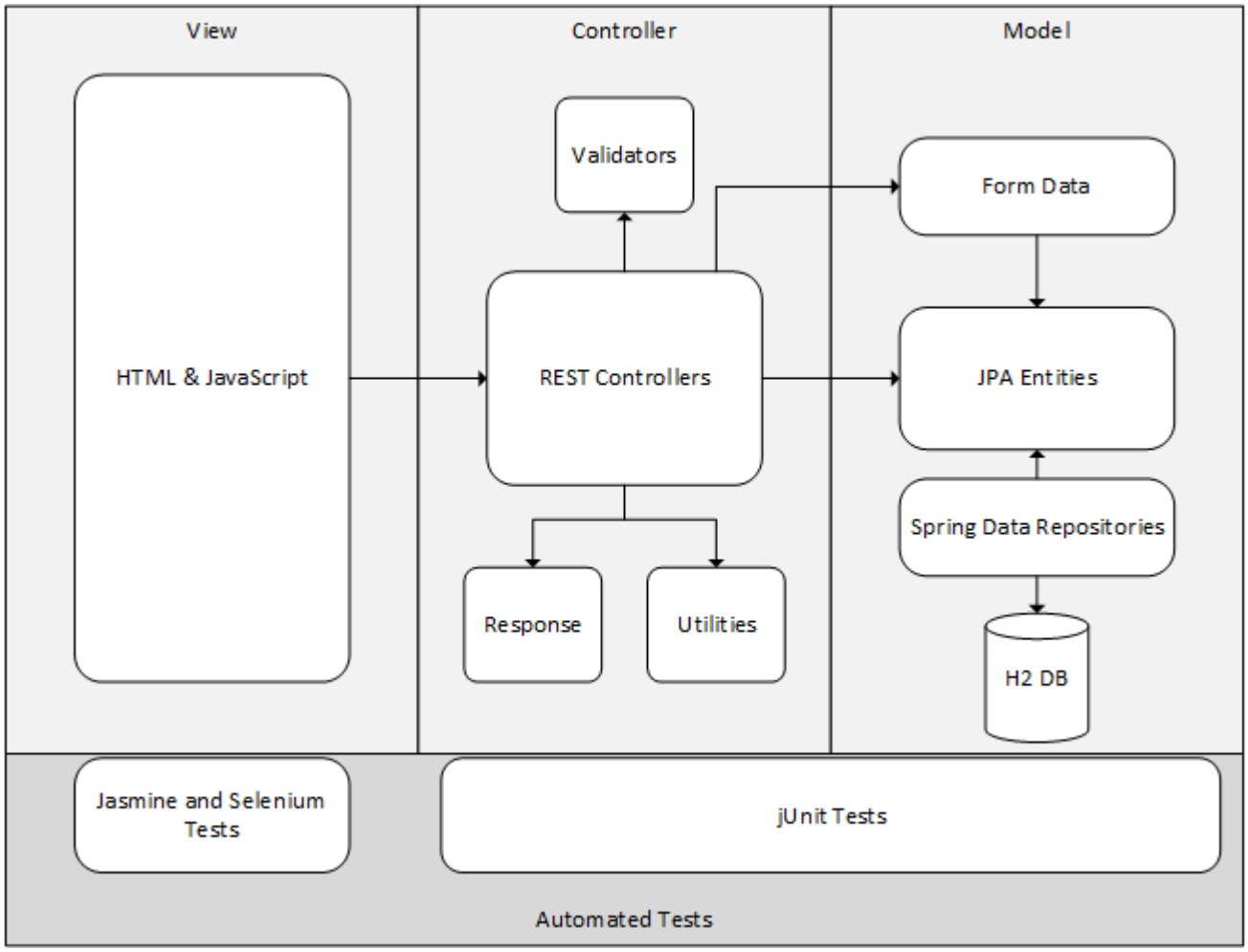

Figure 4 - App Architecture

4.2.1. Model

Our model primarily consists of 6 entity classes that are persisted to the embedded H2 database. These classes are found in the package `cs428.project.gather.data.model` and are shown in the UML class diagram below:

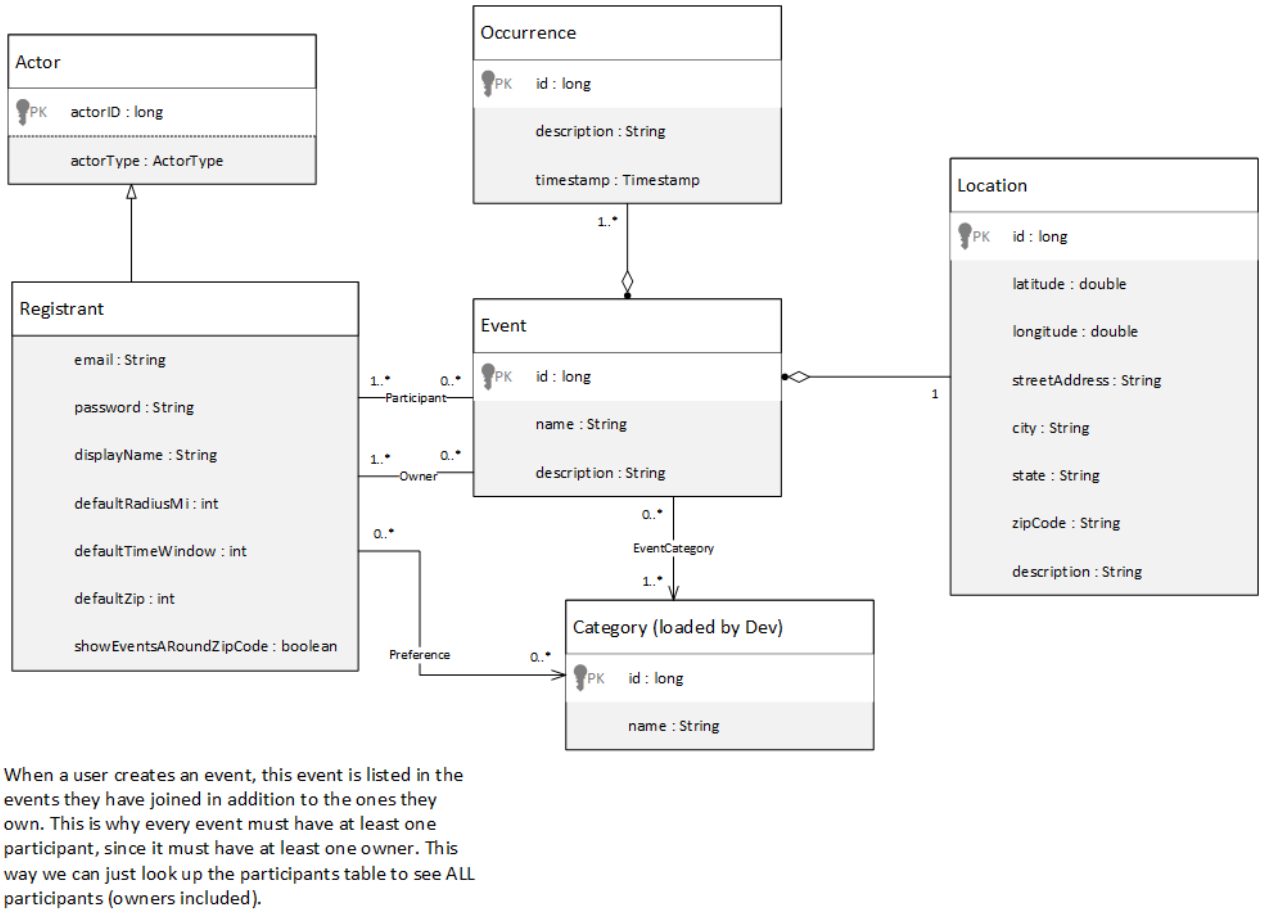

Figure 5 - Model UML

Through JPA annotations, these 6 entities explicitly indicate the relationships depicted above and Hibernate ORM is able to automatically set up all the required tables in the database. We can then query for and save these entities through their respective repositories: `EventRepository`, `RegistrantRepository`, or `CategoryRepository`. Base implementations of these repositories are provided by Spring Data and then extended by our specific query methods.

4.2.2. Controller

Our controller is comprised of several Spring web controllers that are located in the `cs428.project.gather.controllers` package. These classes all extend `AbstractGatherController` which holds some critical variables and helper functions that are common to all controllers. We divided these controllers by function and developed the 5 following classes:

- `HomeController` – Responsible for serving our html templates.
- `SessionsController` – Responsible for signing in, signing out, and returning the session authentication status and username.
- `RegistrantsController` – Responsible for registering new users, updating user profiles, as well as returning user data and a list of other registered users to the client.
- `EventsController` – This is the largest of the controllers. It handles returning the nearby events, adding, updating, and removing events. It is also responsible for handling requests to join or leave an event, or view a list of events that a user owns or participates in.
- `CategoriesController` – Responsible for returning the pre-defined list of categories to the client.

4.2.3. View

Our View component is supported by HTML, JavaScript, and jQuery technology. The code is located at `/src/main/resources/{static, templates}`.

The `templates` are comprised of `header`, `letpane`, `rightpane`, `index.html`, `registerform.html`, and `zipcode.html`.

- `Header` – the top header bar including the user sign form.
- `Letpane` – the left side of the page where the event list is displayed.
- `Rightpane` – the right side of the page where it contains the main map interface and profile form.
- `Index.html` – the main page that loads all the JavaScript and Cascading Style Sheets(CSS).
- `registerform.html` – the user registration
- `Zipcode.html` – the mapping of zipcode and coordinates.

The `static` is comprised of a collection of JavaScript, CSS files, images, and utility scripts. The major implementation is under `scripts` directory:

- `awesomplete.js`– the external autocomplete library.
- `event.js` – the JavaScript to handle dynamic interactions for event list.
- `global.js` – the JavaScript to keep the definition of global variables.
- `interaction.js` – the JavaScript for main interactions on the home paging Style Sheets(CSS).
- `jquery*.js` – the external jQuery library
- `map.js` – the JavaScript to handle dynamic interactions for the main map interface.
- `profile.js` – the JavaScript to handle dynamic interactions for the user profile update interface.

4.3. REST API

REST URL	HTTP Method	Possible HTTP Status Code
/rest/registrants/signin	POST	ACCEPTED UNPROCESSABLE_ENTITY LENGTH_REQUIRED BAD_REQUEST CONFLICT UNAUTHORIZED NOT_FOUND
/rest/registrants/signout	POST	OK BAD_REQUEST
/rest/registrants	POST	CREATED UNPROCESSABLE_ENTITY LENGTH_REQUIRED BAD_REQUEST CONFLICT
/rest/registrants/update	PUT	CREATED UNPROCESSABLE_ENTITY LENGTH_REQUIRED BAD_REQUEST CONFLICT
/rest/registrants/info	PUT	OK BAD_REQUEST
/rest/registrants/displayname /rest/registrants/displayname?page=2	GET	OK BAD_REQUEST
/rest/session	GET	OK

		BAD_REQUEST
/rest/events /rest/events?page=2	PUT	OK BAD_REQUEST
/rest/events	POST	OK BAD_REQUEST
/rest/events/update	POST	OK BAD_REQUEST
/rest/events/remove	POST	OK BAD_REQUEST
/rest/events/join	POST	OK NOT_FOUND BAD_REQUEST
/rest/events/leave	POST	OK NOT_FOUND BAD_REQUEST
/rest/events/userJoined	GET	OK BAD_REQUEST
/rest/events/userOwned	GET	OK BAD_REQUEST
/rest/categories	GET	OK BAD_REQUEST

4.4. Sequence Diagrams

Having discussed the architecture and major components, some UML sequence diagrams might further explain the operation of our application. See below for sequence diagrams for a representative subset of our use cases:

Access Homepage, Change Location, View Event List, View Event Details

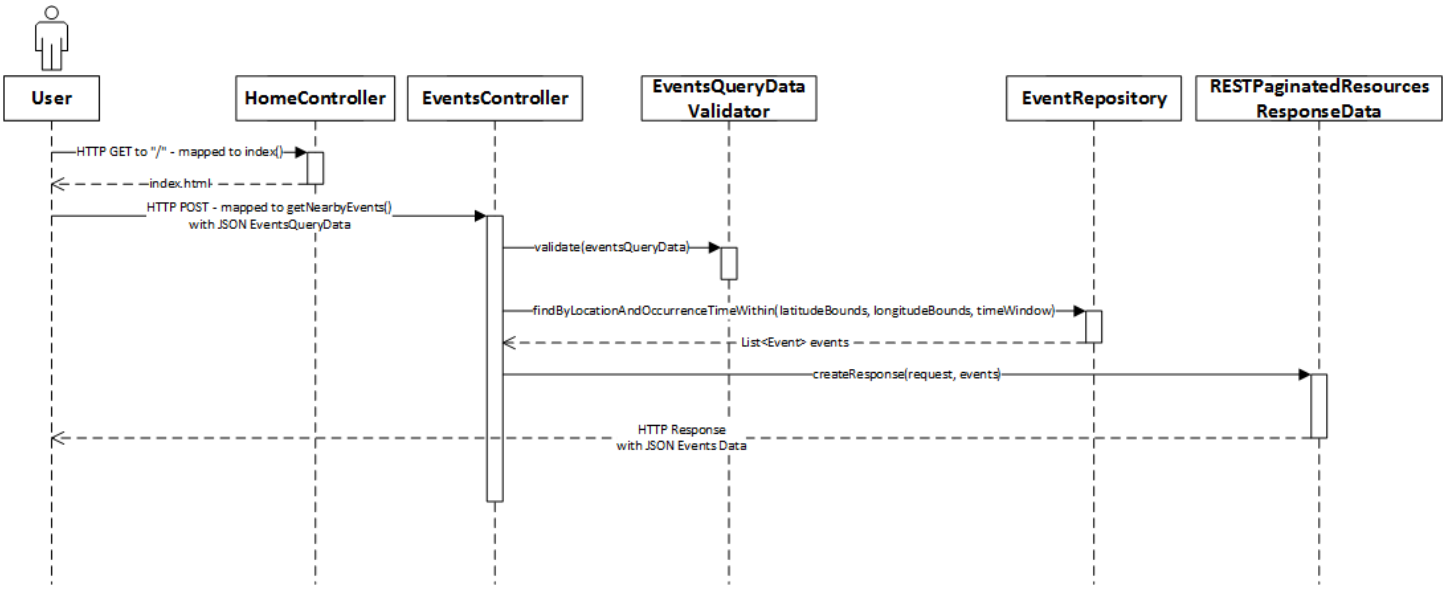

Figure 6 - Access Homepage Sequence Diagram

Register

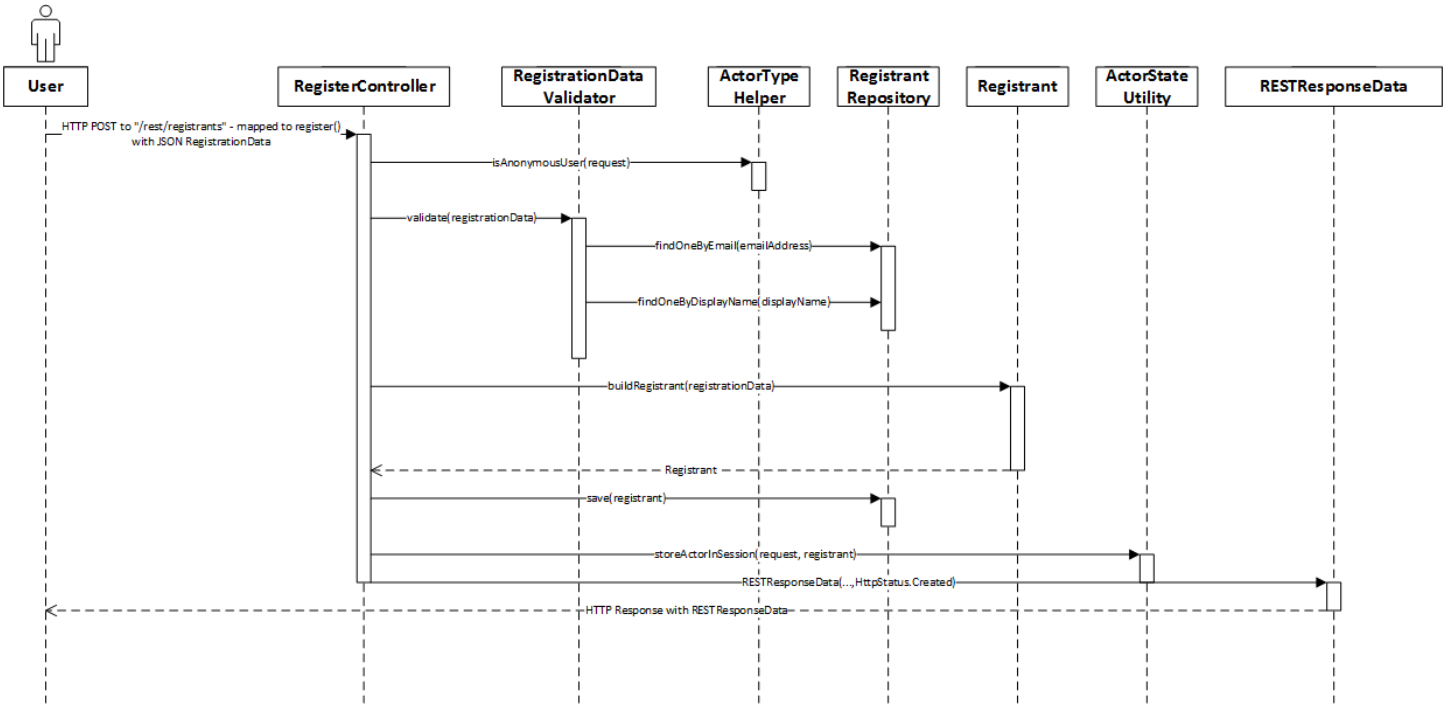

Figure 7 - Register Sequence Diagram

Sign In

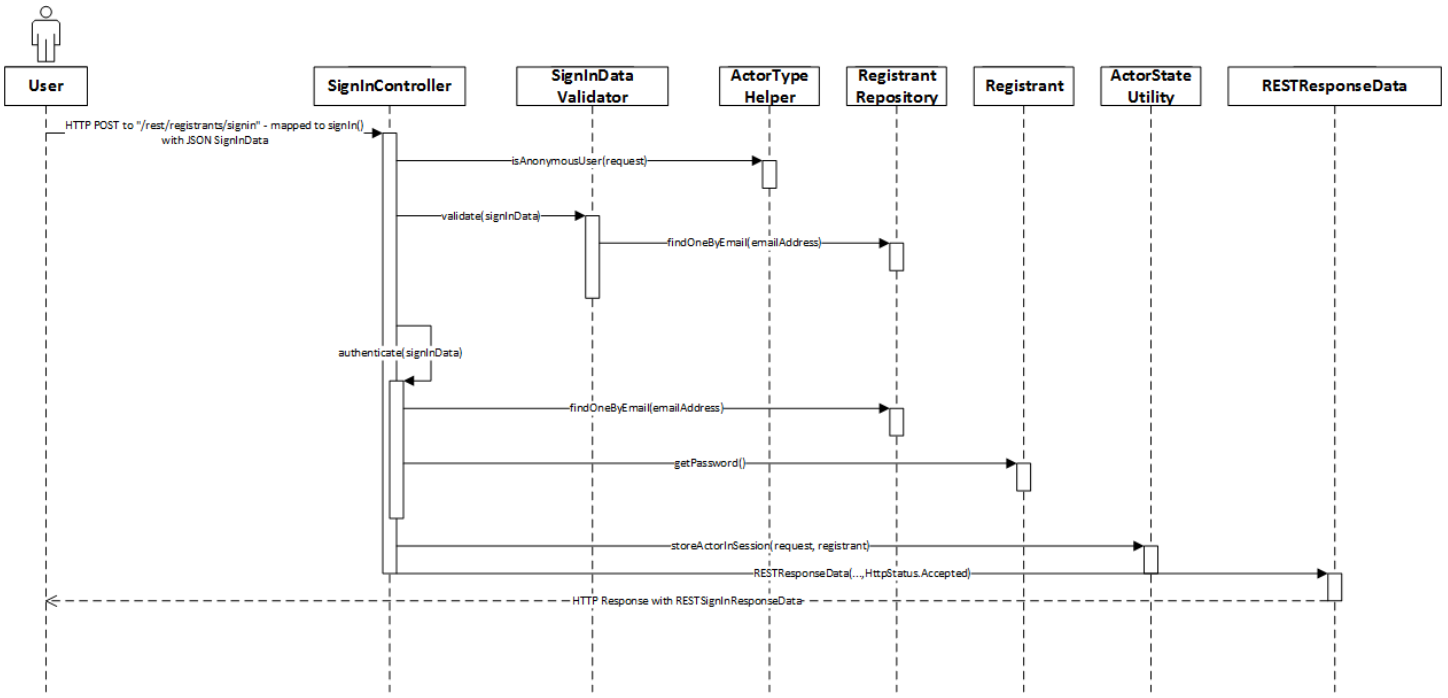

Figure 8 - Sign In Sequence Diagram

Sign Out

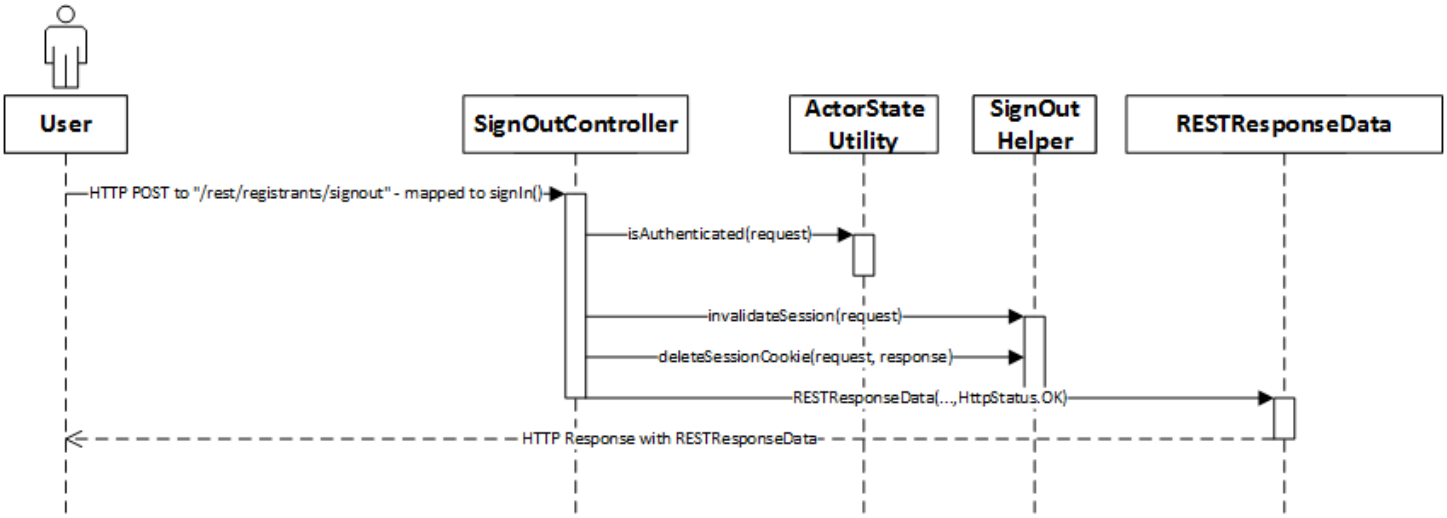

Figure 9 - Sign Out Sequence Diagram

Add Event

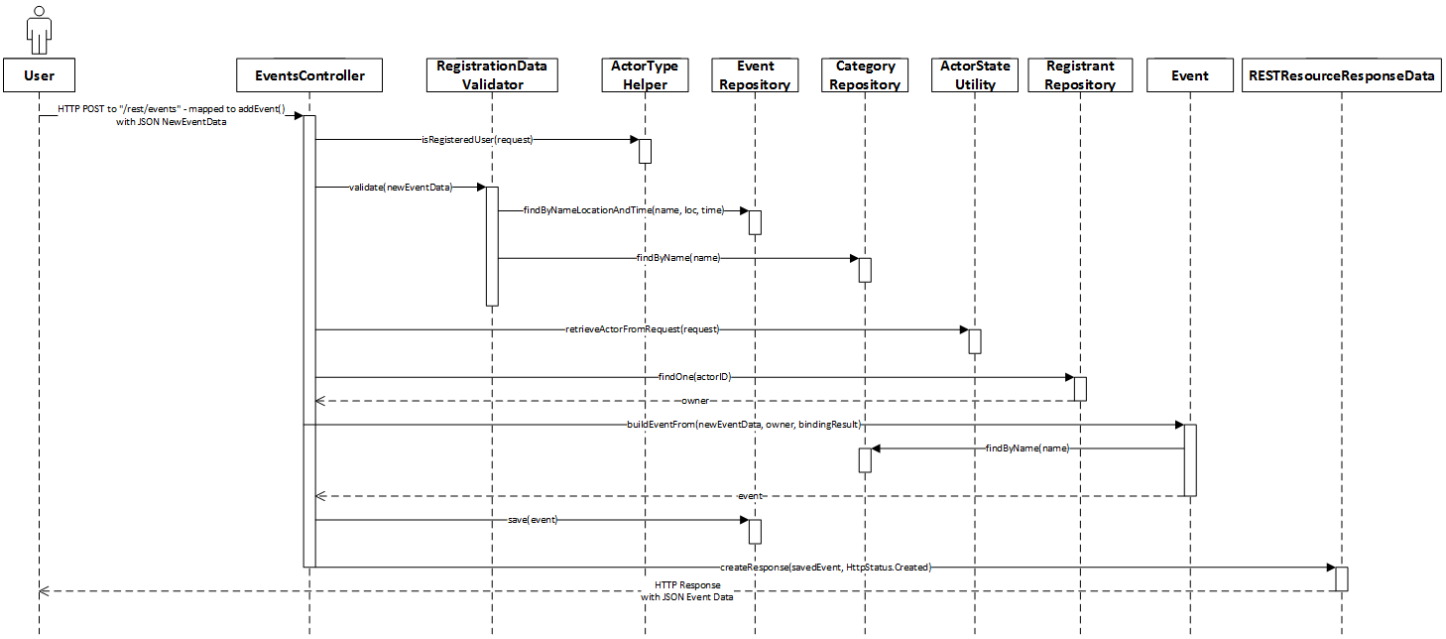

Figure 10 - Add Event Sequence Diagram

4.5. Framework Reflection

The Spring framework heavily influenced our design decisions. We relied on the frameworks overall controller architectures and leveraged the frameworks JPA entity integration to build our model. We appreciated the clean abstraction of Model-View-Controller that was provided, but we did experience challenges developing the JPA entities and getting our desired database interactions. We also had to be patient about the inherited characteristics about Java in general, i.e. heavy amount of the code for each component, verbose exceptions causing difficulties in debugging, and a considerable amount of mocking for unit tests. These challenges became increasingly difficult due to our lack of expertise using Spring. In the end, we learned that it's extremely beneficial for a team to have framework expertise to avoid frustration on debugging framework specific issues and increase iteration productivity

significantly. If we were to build this project again, we likely would go with an alternative framework, like Rails or Django, as suggested by one of our team members.

Specific challenges that we encountered while developing this project include:

- The JavaScript unit testing framework Jasmine that we use was not Java 8 compatible, and so we could not take advantage of Java 8 to program more efficiently.
- Version incompatibilities with the Maven build system on OpenShift prevented us from being able to use the Java MySQL driver, and so we had to rely on an embedded H2 database.
- We could not rely on Spring to auto-generate most of the boilerplate model and controller code because we had a customized model hierarchy and needs that we could not easily express using Spring attributes (Spring's domain specific language). As such, a lot of the server-side code is manually-written boilerplate code as opposed to business logic.
- In the case of complex model queries, we had issues building the queries using Spring attributes and JPQL. So for some queries, a superset of records must first be pulled from database into memory for a second-level in-memory query. This solution does not scale, but with limited time we did not think it wise to focus on learning the intricacies of Spring Data, JPA, or Hibernate.

5. Reflections and Lessons Learned

Heeho Park

Few years back, I taught myself web design languages such as html, PHP, jQuery, CSS, MySQL, and others to create a website called TallyOp.com. That took me about two years to start from scratch without any web language knowledges on my free time with a full time job. The experience from that project was much help for this project to contribute in the front-end development. But I never realized there were such frameworks that you can build websites on top of them to speed up the development process. In our case, we used Spring Boot to connect the interactions of users in the frontend to the backend database. There was a learning curve in the beginning of the project, but once learning the concepts, the work was easily managed. It was new experience and motivated me to find out about other frameworks a team member recommended, Ruby on Rails. When I built TallyOp.com I was a novice, I did not have any testing done for the application, and did not have automated testing/deployment system. This is something that I learned to be crucial as well. I'm actually testing much more at my work for Python wrapper and Fortran code development and it's been awesome to catch errors and bugs as I further develop the code.

As for the team experience, it's been amazing to use Slack and Bitbucket as a team communication and repository tools. My industry does not use such dynamic communication system due to security concerns, but I can see why start-ups and small tech companies prefer interfaces like Slack. I had experiences with Bitbucket as a code developer; however, I learned that Bitbucket can be used as a communication tool to cover many aspects of the code development. Also, I only used Mercurial as a code repository, but Git provided much more flexibility in code development.

As a nuclear engineering background, I never such experience like pair programming and agile development. At work we use Gantt chart for almost all projects and can't imagine having two sets of eyes for code development, but it was a new experience. Pros of pair programming was that it was much less prone to make mistakes, but the con was that it was spending two people's time and was actually hindering progress. As for the agile development, there were conflicting ideas about it. One thought that it just had to be done to meet the customer's satisfaction, but it later became burdensome to re-generalize many portions of codes. My argument was that the agile concept should still achieve customer's satisfaction while writing the codes as general as possible.

Overall, this class opened my eyes in many aspects of software development, and gave me an idea of what processes are being done at start-ups and small tech companies.

Benson Ma

As a programmer who works on backend web services daily, I came in with higher expectations and estimated that this project's deliverables could be achieved by the team in a little over a month, or 2-3 iterations. I had hoped to take this project much further; however, I found that there was an experience gap between the different members of the team, which was contrary to my expectation that this course was targeted at students already familiar with the technology aspects but unfamiliar with the project-management aspects of software engineering. Consequently, the challenge for me was in bringing out my real-world work experience to the team and helping my fellow team members familiarize themselves with the software engineering processes and the technologies and idioms around web services. I hope this class provided valuable experience to the other members of the team.

Hiren Bhavsar

Coming from a Systems Engineering Background and having very limited experience in Web Application Development (e.g., ASP.Net), this project was very challenging in terms learning Spring Boot framework and Frontend Programming. Furthermore, setting up the developer workflow was difficult at the beginning but once that was figured out, it was very streamlined and straightforward. Having some experience in continuous integration (e.g., Jenkins) from past semester helped a lot for setting up developer workflow. Mainly, building a web application with J2EE and Spring Boot within academic semester was extremely challenging. However, with great team, we were able to complete this project successfully as planned. Overall I gained very valuable experience in developing 3-tier web application architecture using Spring Boot.

Kai Song

This is the first project that we have a big team whose members all come from completely different backgrounds. We have definitely experienced challenges on communication and learning various frameworks (Spring/Java MVC, JPA, HTML/JavaScript/jQuery, OpenShift deployment, and Jenkins CI). We have various instances that some members are frustrated due to the nature of Spring/Java framework or miscommunication between other members. As a result, we found out we had to work as a team and accommodate each other's schedules, preferences, and knowledge backgrounds. During the process of resolving these challenges, we learned that communication for large project team is extremely important. Sometimes, in order to work together productively, having effective communication is far more important than individual technical knowledge.

Max Gabreski

This project is my second real experience with web apps, and my first with building RESTful web services. As such there was a bit of learning curve regarding the architecture and overall implementation in the beginning. In addition, I was inexperienced in the framework we had proposed but I was excited to pick it up as a lot of enterprises I have interviewed with use it for their web services. We hit a couple of rocky patches as we went through the project and had to deal with greatly varied levels of experience. However, by leveraging pair programming and other strategies we were able to accomplish a lot of our goals. Looking back on this project I wish I had been able to devote more time to learning and working in the frontend, as this is another area where I am lacking experience. This is something I will try and focus on more in future work.

As far as process is concerned, I think our project went quite well. In the early iterations we focused on our workflow and environment to make sure everyone was set up for success. Setting up on own Jenkins server and working through the headaches of tying it to our repository and deploying our app to all of our desired servers seems like valuable real world experience.

Souhayl Maronesy

Coming from an Electrical Engineering background most of the frameworks and languages we used for this project were fairly new to me. The only language used in the project that I had some experience with was Java. Therefore, this project was a great opportunity for me to learn many new things. I gained experience with the full stack of web applications from the database to the JavaScript in the front end. I also gained valuable experience with MVC in general and the Spring Boot Framework in particular. We tried to follow the XP process

as much as possible and all members of the group contributed greatly. The fact that we had to develop a piece of software from scratch and were responsible for choosing all the technologies introduced some complication as the beginning. However, we were able to overcome all obstacles and learned from some of the bad decision we made at the beginning. Overall this project was a learning lesson both from the process aspect and the product aspect.