Project 6 Report: MapGeist

Alex Costinescu, Ayden Blotnick, Lucas Sward

Final State of System:

Features:

- Add Events to the map as a user by double clicking on the map
    - Also implemented a third-part DateTime picker for the Start/End times
    - Submitting Events asynchronously sends them to the server and displays a message on success.

- Login as a Moderator (User authentication)
    - Username/password authentication using session tokens.
    - Some views/API calls are protected and only available to logged in Moderators.

- View Events staged for approval on the map as a Moderator.
    - Approve/Deny Events asynchronously using the buttons.
    - Reviewed Events are removed from the list and map.
    - Events to review are automatically replenished on login.
        - Synchronized to ensure Events are only added to one Moderator's working queue.

- Active Events (Approved and ongoing/upcoming) are pulled from the database and placed on the public-facing map asynchronously on page load.

- Viewing Events:
    - Hovering on a marker on the map highlights the Event in the list.
    - Hovering on an Event in the list centers the map on the marker for the Event and highlights it.

Excluded Features:

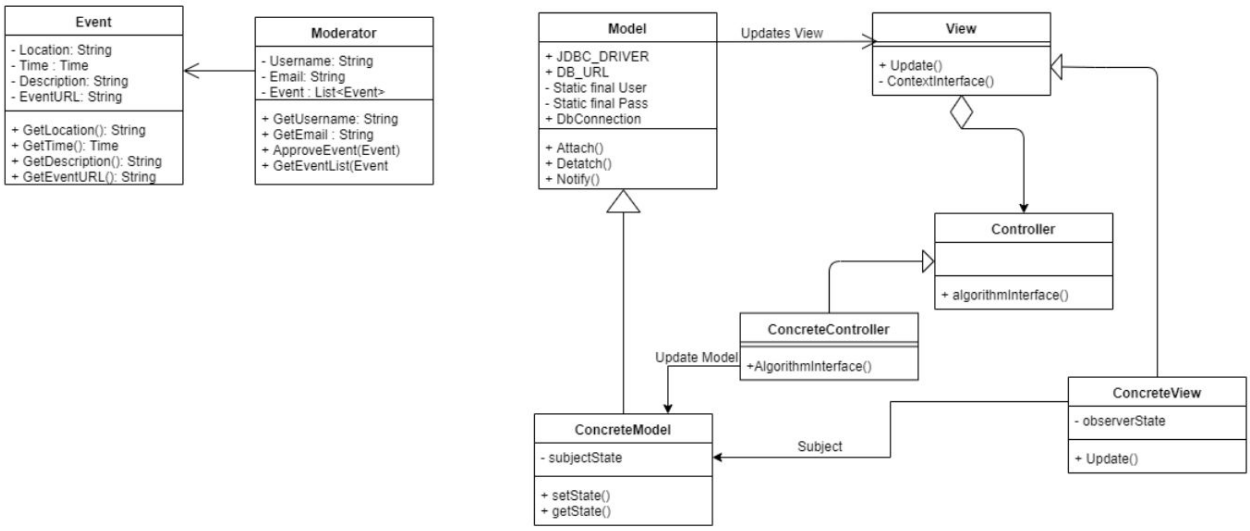
- Public User Accounts
    - We decided this wasn't necessary - it's more convenient to be able to submit Events anonymously and have Moderator's filter out bad/fake Events.

- Web Scraping
    - This feature was removed due to the time in which it took in order to complete it. We looked into two separate libraries to scrape the https://www.colorado.edu/events/ including htmlUnit and Jsoup. Jsoup does not allow one to correctly parse html generated by javascript at runtime. The htmlUnit libraries were experimented with, however, we did not have adequate time to complete it.

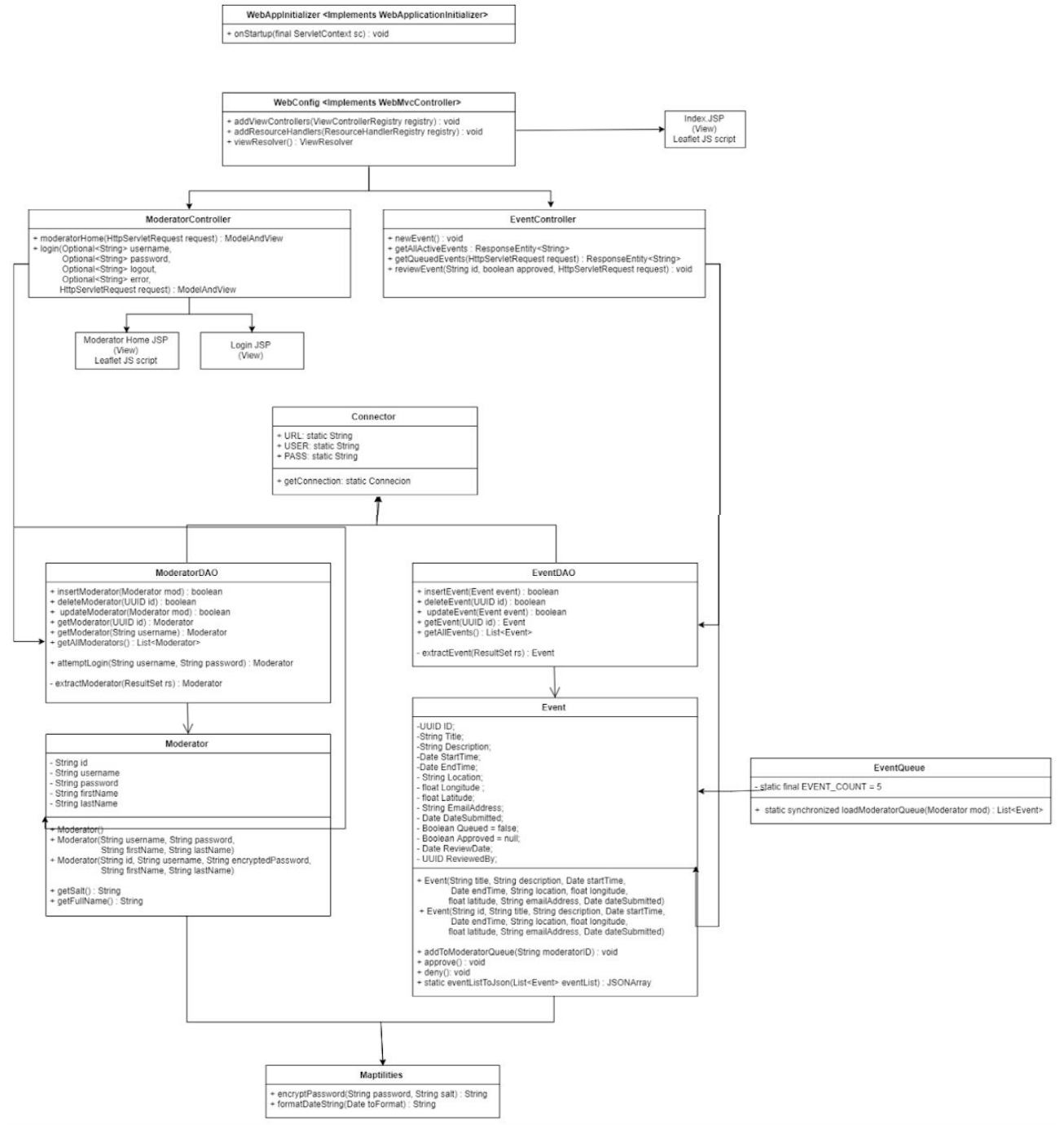
- Mobile View
    - This was again removed due to time constraints. We found that trying to build/debug a mobile view that would work on all of the different versions of iOS/Android would take too much time and decided instead to focus on making the desktop view as good as possible.

Final Class Diagram and Comparison Statement

Project 4 Class Diagram:



Final Semester Project Diagram:



Key Changes Statement:

As exemplified in the UML class diagram above, we decided to move toward a spring MVC framework instead of building our own. We decided to utilize the Spring MVC framework to create the connections between the model, view and controller classes. Additionally, we split the model into both Moderator and Event objects with additional ModeratorDAO and ObjectDAO data access objects that act as the intermediary between the SQL database and our Java code. Additionally, We added a connector object that has a static method which initializes the connection to the SQL database. This is used within the DAO objects to open and close connections with the database easily. Lastly, we created an event queue object that handled moving the events to be approved into the moderator's view.

Third Party Code Usage:

Overall we utilized a few tools listed below to create the framework for our web application. While these tools were used to build the backend and initial framework of our project, the majority of the code was written independently with reference to various online sources such as https://stackoverflow.com/.

Spring MVC Framework:

https://docs.spring.io/spring/docs/current/spring-framework-reference/web.html
Was utilized to create an MVC connection between the view (index.jsp) the model (eventDAO, Event, Moderator, Moderator DAO) and the controller (EventController, ModeratorController).

Leaflet for Map Generation:

https://leafletjs.com/examples/quick-start/
This link assisted us in creating the basic interactive map on the web page via javascript.

Handling HTTP Requests via Javascript:

https://www.javascripture.com/XMLHttpRequest
This link was useful to ensure the webpage didn't need to be reloaded once new event information was posted to the web server.

Tomcat:

https://crunchify.com/step-by-step-guide-to-setup-and-install-apache-tomcat-server-in-eclipse-development-environment-ide/
This tutorial was helpful to create a webserver that can host a java web server.

JDBC Connections and SQL Queries/Updates:

https://dzone.com/articles/building-simple-data-access-layer-using-jdbc
This tutorial provided the proper libraries and APIs we needed to utilize in order to connect to a SQL database and perform queries.

Statement on OOAD process for Semester Project:

Key Design Process Elements or Issues:

1. Web development uses multiple components and various languages such as Java, Node.JS, HTML, and CSS. Each of these languages must be understood and practiced in order to create a web page that is suitable to current standards.

2. It is often best to utilize a tool (such as spring MVC or JDBC) that has significant documentation and online resources instead of creating code entirely from scratch. This also applies to many Javascript plugins - it's not often a good use of your time to develop your own, for example, DateTime picker when there are many, good implementations that already exist.

3. We ran into an issue when it came to extracting the Event form data from the webpage and Posting it to the spring EventController. The data submitted was not in the correct format to be received by the Spring framework. Through multiple iterations of debugging and multiple HTML submission formats, we found that the data needed to be properly encoded from a Uniform Resource Identifier (URI) component by replacing each instance of certain characters by one, two, three, or four escape sequences representing the UTF-8 encoding of the character.