Crime Maps

Team Members:

Alex Yin, alexyin@usc.edu

Yihan Lin, yihanlin@usc.edu

Eliot Chang, eliot.chang@usc.edu

Frank Pang, fpang@usc.edu

Table of Contents

Proposal **3**

Project Proposal 3

Team Members 3

Meeting Times 3

High Level Requirements **4**

Technical Specifications **5**

Web Interface 5

MySQL Database 5

MySQL Backend Driver 6

Web Scraping 6

Backend Sockets 6

Multithreading 6

Networking 7

Detailed Design 7

Sitemap **8**

GUI Mockup 8

Class Diagram and/or Inheritance Hierarchy 8

Database Schema 9

Hardware and Software Requirements 9

Server/Client/Multithreading Classes 10

Key Algorithms 11

Testing 11

Black Box Testing 11

White Box Testing **12**

Unit Testing 12

Stress Testing 12

Regression Testing 12

Deployment 13

13

13

13

FP - Project Proposal, Team Members, and Meeting Times

Team members and emails:

	Eliot Chang	Eliotcha@usc.edu
	Yihan Lin	yihanlin@usc.edu
	Frank Pang	fpang@usc.edu
	Alex Yin	alexyin@usc.edu
TA:		
	Adina Kruijssen	kruijsse@usc.edu

Weekly Meeting time:

Time: Friday 11:00 am - 1:00 pm Location: Leavey Library

Project Proposal:

We want to create a web application that helps people understand the safety of neighborhoods in Los Angeles. As USC is located near Downtown LA and many students are not LA natives, USC students would be a good example of people who would want to use this kind of app. Currently however, there are no simple ways of assessing the safety of neighborhoods online. USC students receive crime alert emails but they do not have a sense of how close they are to crime sites. People can look up crime rates and individual crime incidents online but these websites cannot provide a straightforward impression of which areas are safer. Our project proposal is as follows. We want to create an intuitive, graphical display of crime data in the greater Los Angeles Area to help LA residents better understand their personal safety and neighborhoods. Our first goal is to be able to show a heat map representing safety based primarily on the severity and frequency of crimes committed in an area in a way similar to the below image. We want to create a web application, because the primary problem with assessing neighborhood safety is that it requires a large time commitment and research. We want to streamline that process to be as simple as possible.

FP - High-Level Requirements

General idea and goals:

For our project, we want to create a web application that helps people understand the safety of neighborhoods in Los Angeles. USC students receive crime alert emails but they do not have a sense of how close they are to crime sites. As USC is located near Downtown LA and many students are not LA natives (there is no simple way of assessing safety around LA), USC students would be a good example of people who would want to use this kind of app.

When users go on the website, they will have the option of logging in to their account, registering, or continuing as a guest. Guests will have the functionality of viewing general crime data within a certain radius on the map. To register, guests need to enter their current address. If guests login or register, they will have the added functionality of having personalized crime information based on their current address, save and view past search history even after logging out, and they will also be able to filter the map based on certain criteria. Data will be of individual crime incidents over the past few weeks/months and are organized based on types.

Overall, we want to create an intuitive, graphical display of crime data in the Greater Los Angeles Area to help LA residents better understand their personal safety and neighborhoods. The data that we gather will be pulled from online API sources that already store crime data. We will be focusing around LA. We will be storing the data we pull and displaying the information to the user as required.

Registered users will be able to invite other users to a “room” and in that room, they can collaborate search results (filters, dragging maps) and all users will display the updated display based on what a certain user searched. They can also talk on a call with one another at the same time.

For the overall flow of the website, we will have two main pages: a login page and a map page. On the login page, our app will figure out whether a person is a user or a guest, then display the proper map page. From there, we want to be able to show a heat map representing safety based primarily on the severity and frequency of crimes committed in an area. By clicking on regions on the map, people would be able to read further details about a region, like median house value or how most people commute.

<u>FP - Technical Specifications</u>

Web interface

- Wireframing*(2 Hours)*
    - Put together the login, sign up, and main functional user flows using Balsamiq, Figma, or other wireframing platform is best.
- Low Fidelity Mockups*(2 Hours)*
    - Create low fidelity mockups to figure out what elements will be on each page.
- High Fidelity Mockups*(2 Hours)*
    - Once low fidelity mockups are finalized, implement the high fidelity mockups with the finalized graphics and visuals.
    - Content will contain more specifics of what certain parts of the application will look like/contain.
- Front-end coding *(6 Hours)*
    - Starts with a Login Page that will redirect based on buttons to:
        - Sign Up Page
            - User will be asked to sign up with their name and password.
            - After they sign up, they will be redirected to the Main Page
        - Guest Page
            - User will be redirect automatically to the Main Page, but without the additional functionality of a user (interacting with multiple people in rooms)
    - Main Page
        - The main page will contain a search bar to ask the user what location they would like to search within LA (additional locations will be provided if time allows). After the search has been completed the heat map will be updated and they will have the option to toggle parameters to filter the data depending on what they would like to see.
        - Possible filters:
            - Type of crime
            - Time frame to view (day, week, month)
            - Enable AirBnB functionality of being able to see rental prices depending on your viewing area

MySQL Database (3 hours)

- User
    - The User table is used to store user's username and password. It is used in verifying the user's validation. It will contain a userID, a username, and a password

- SearchHistory
    - The search history table will be used to store the searches of a registered user.
    - It will contain an auto incremented id column, along with a cityname column, and a long/lat column.

MySQL Backend Driver (4 hours)

- Create a JDBCDriver class for database operation
- Class is static and contains the following functions
    - Connecting
    - Closing
    - Adding users
    - Updating users
    - Validate user-login/register
    - Add search history
    - Find search history

Web Scraping (5 hours)

- Information from the API will be pulled and as a backup, if all fails, we will have a Python program scrape the information online to later store in MySQL

Backend Servlets (5 Hours)

- Create a servlet for each page
- Register Servlet
    - Verify user input
        - Check if the username has been taken
        - Check if the username and password field is empty
    - If the input is invalid
        - Return to the register page and send error information
    - If the input is valid
        - Return to the home page with user logged in
        - Add new user to the database
        - Add new user to the session
- Login Servlet
    - Validate input
        - Not empty field
        - Check if the username exists
        - Check if the input password matched the password in the database
    - If the input is invalid

    - Return to the login page with the error message
  - If the input is valid
    - Add user to session
    - Return to the home page with user logged in
- Logout Servlet
- Search Servlet
  - Add current search history into the user profile if session has user logged in
  - Get method gets the area postal code and return with the corresponding crime data

Multithreading (8 hours)

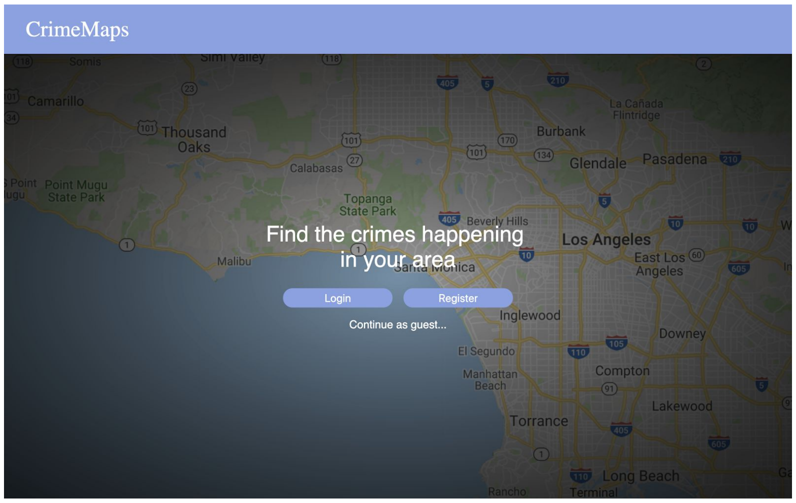
- Multithreading will be done with registered users having an additional functionality of entering a 'room' with other registered users. In this room, they will be able to collaborate on the same screen by toggling filters or making searches. Screens of all users of the room will update depending on these changes.

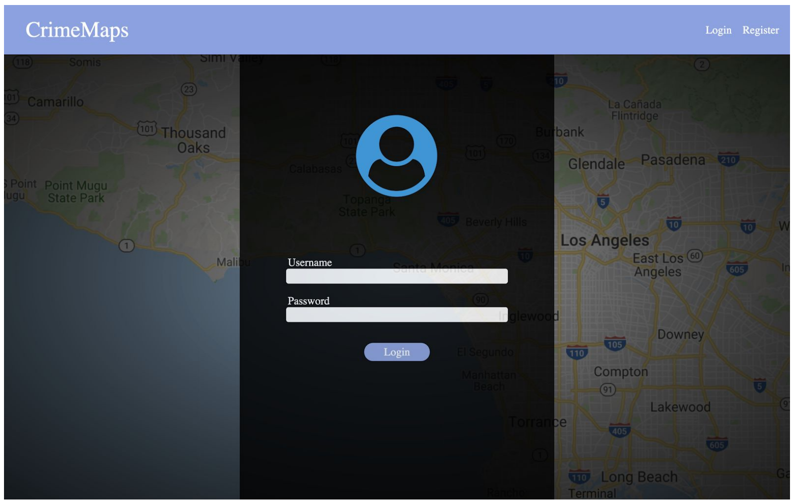
Networking (2 Hours)

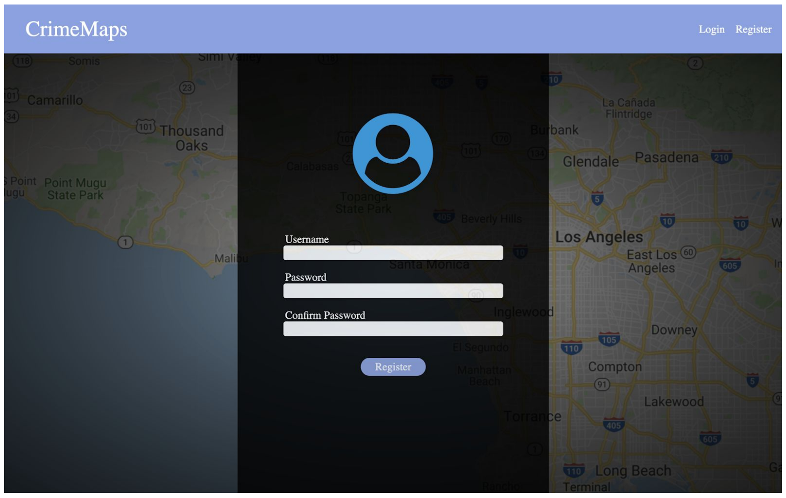
- Map updates are pushed through the websocket from client to server

FP - Detailed Design

Login/Registration User Flow



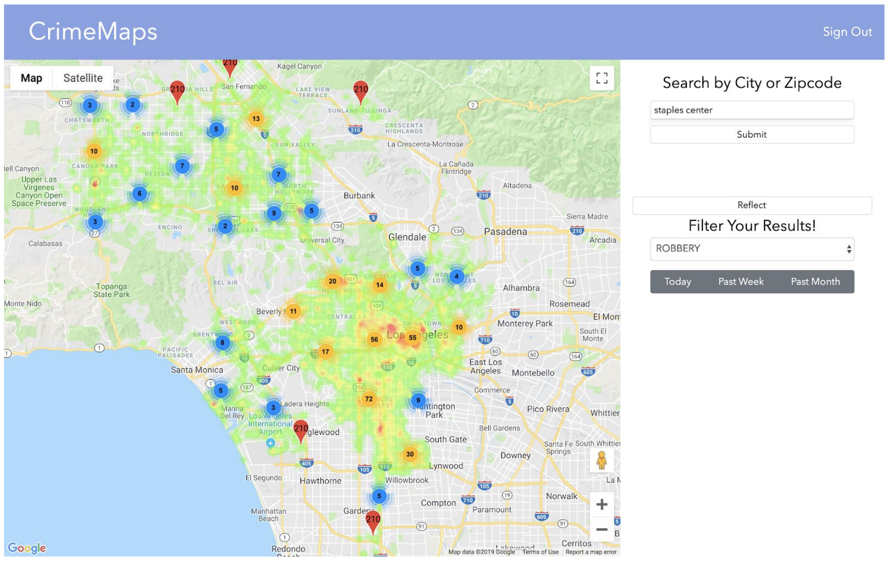




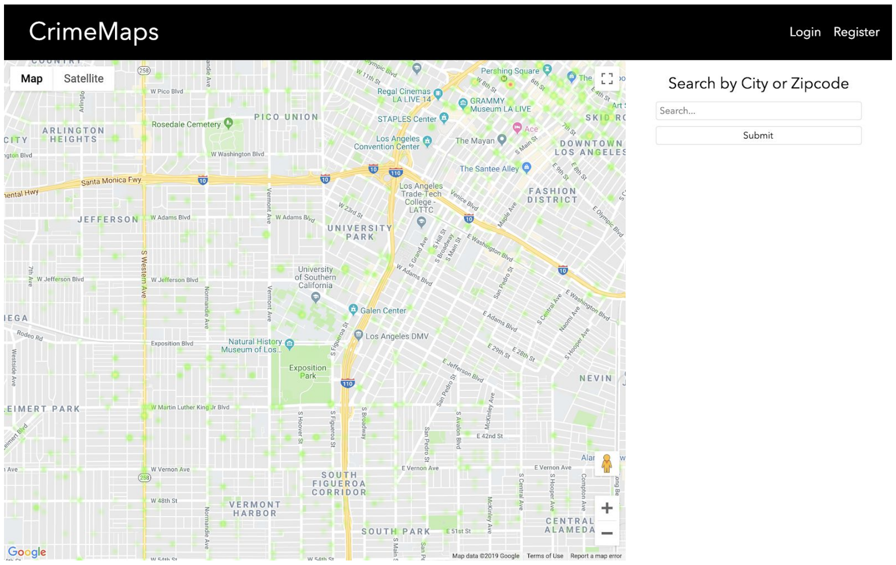
The UserAuthentication class will access the User table in a MySQL database named userLogin using the jdbc:mysql driver. The authenticate() method will have the user input a username and password and verify whether or not the user is in the database. If it is, the function will return true.

Crime Visualization

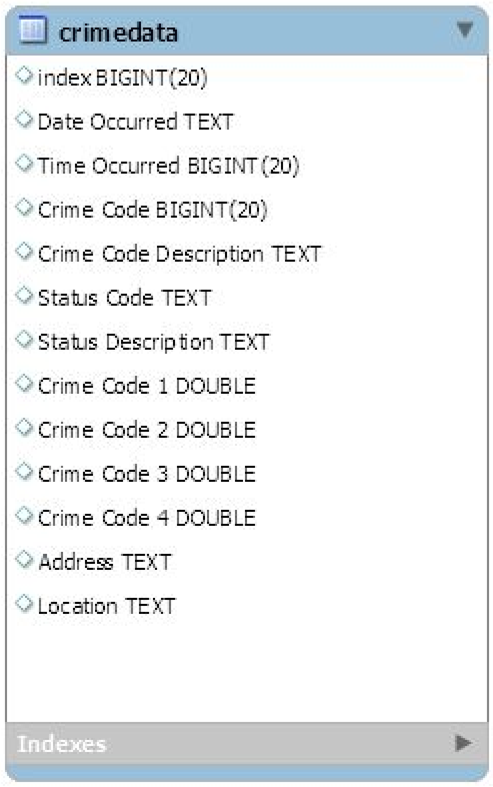
Users can toggle on and off the categories in order for all the users currently on the page to see a change in crime density visualization.



Guests can only search for crime with the heatmap



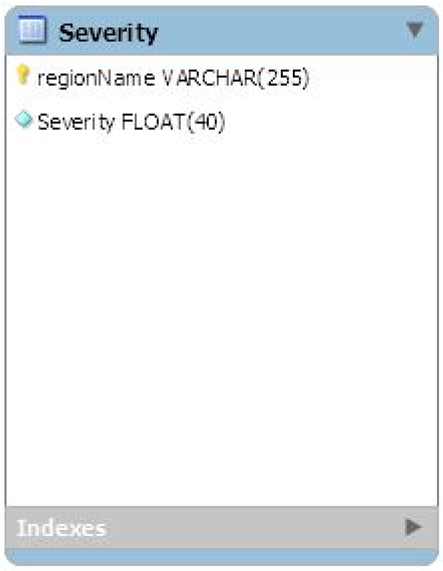
Database Schema



crimedata

- index BIGINT(20)
- Date Occurred TEXT
- Time Occurred BIGINT(20)
- Crime Code BIGINT(20)
- Crime Code Description TEXT
- Status Code TEXT
- Status Description TEXT
- Crime Code 1 DOUBLE
- Crime Code 2 DOUBLE
- Crime Code 3 DOUBLE
- Crime Code 4 DOUBLE
- Address TEXT
- Location TEXT

Indexes



Severity

- regionName VARCHAR(255)
- Severity FLOAT(40)

Indexes

Using data collected from the city of Los Angeles with the above categories (https://www.opendatanetwork.com/dataset/data.lacity.org/y8tr-7khq), our database will store data on the crimes in Los Angeles in the 'crimedata' table. Using these categories, most significantly being Crime Code and Location, our algorithms will calculate a severity (0 to 1) for every neighborhood/ region of Los Angeles which will be stored in the 'Severity' Table. The heatmap and other frontend functions will utilize the 'Severity' Table.

Class Hierarchy and Inheritance Hierarchy

Package: driver

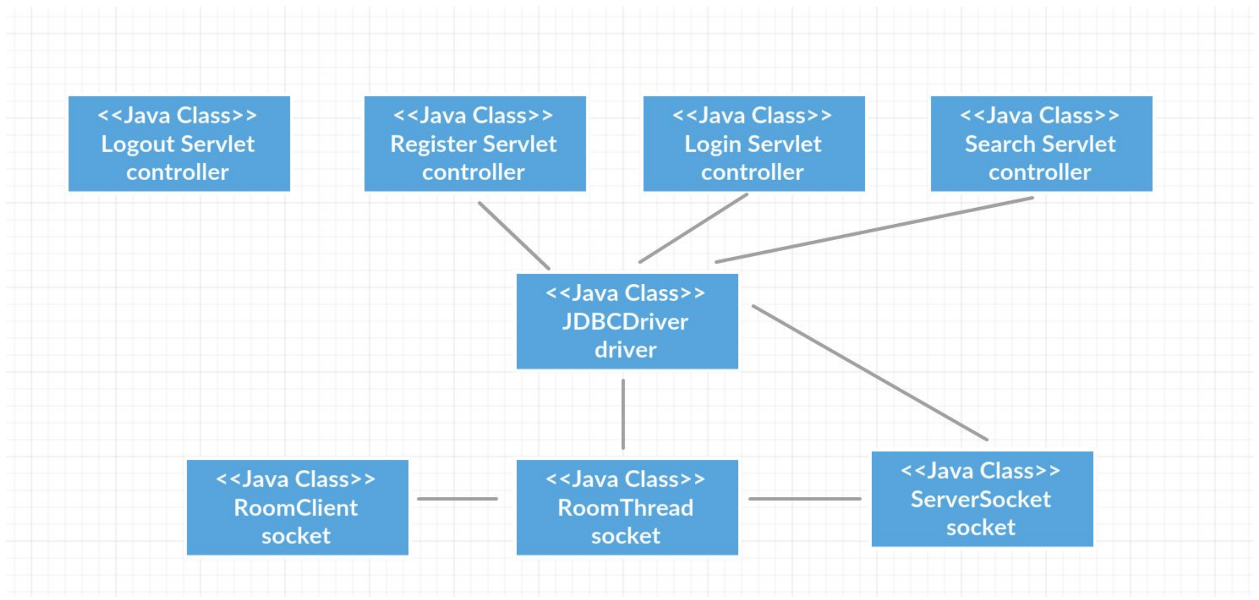
- JDBCDriver

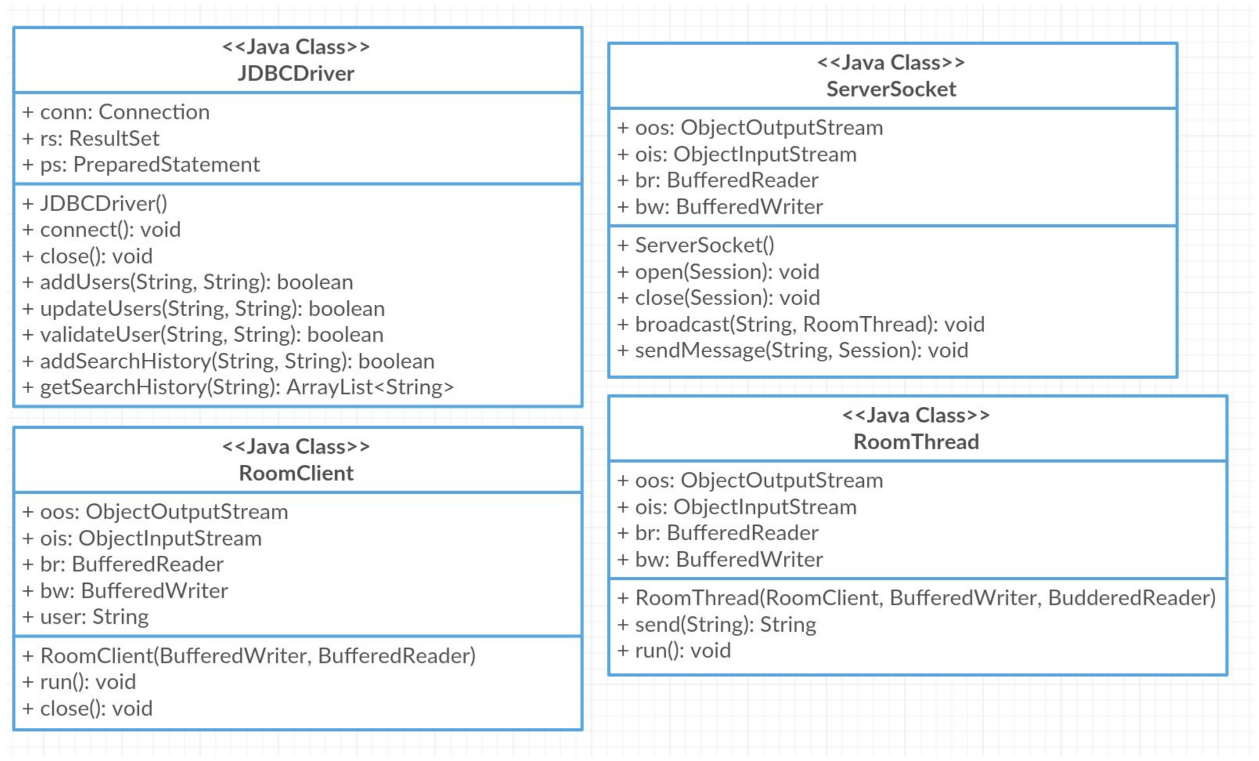
Package: controller

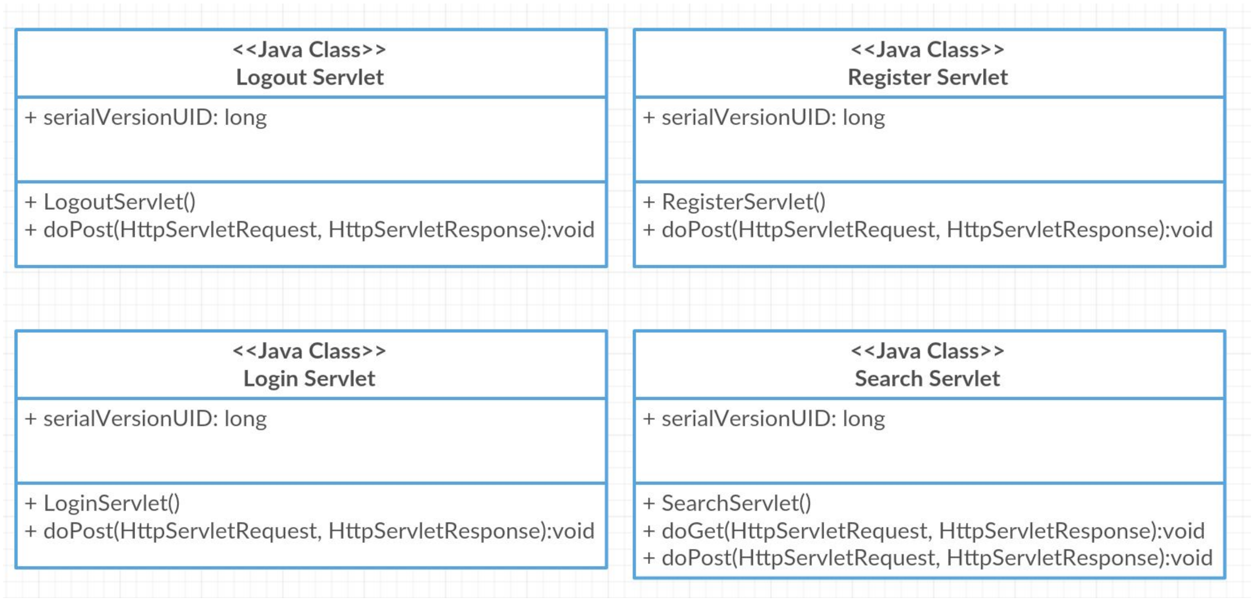
- Register Servlet
- Login Servlet (UserAuthenticate)
    - authenticate()
- Logout Servlet
- Search Servlet

Package: socket

- RoomClient
- RoomThread
- ServerSocket


<<Java Class>> Logout Servlet controller
<<Java Class>> Register Servlet controller
<<Java Class>> Login Servlet controller
<<Java Class>> Search Servlet controller
<<Java Class>> JDBCDriver driver
<<Java Class>> RoomClient socket
<<Java Class>> RoomThread socket
<<Java Class>> ServerSocket socket


<<Java Class>>
JDBCDriver
+ conn: Connection
+ rs: ResultSet
+ ps: PreparedStatement
+ JDBCDriver()
+ connect(): void
+ close(): void
+ addUsers(String, String): boolean
+ updateUsers(String, String): boolean
+ validateUser(String, String): boolean
+ addSearchHistory(String, String): boolean
+ getSearchHistory(String): ArrayList<String>
<<Java Class>>
ServerSocket
+ oos: ObjectOutputStream
+ ois: ObjectInputStream
+ br: BufferedReader
+ bw: BufferedWriter
+ ServerSocket()
+ open(Session): void
+ close(Session): void
+ broadcast(String, RoomThread): void
+ sendMessage(String, Session): void
<<Java Class>>
RoomClient
+ oos: ObjectOutputStream
+ ois: ObjectInputStream
+ br: BufferedReader
+ bw: BufferedWriter
+ user: String
+ RoomClient(BufferedWriter, BufferedReader)
+ run(): void
+ close(): void
<<Java Class>>
RoomThread
+ oos: ObjectOutputStream
+ ois: ObjectInputStream
+ br: BufferedReader
+ bw: BufferedWriter
+ RoomThread(RoomClient, BufferedWriter, BuddederedReader)
+ send(String): String
+ run(): void



Hardware and Software Requirements

- Hardware: none
- Tomcat Server
- Java/Java Servlets for backend
- GSON for parsing class
- MYSQL JDBC Driver 8
- Python (Scraping)
- HTML/CSS/JavaScript
- Eclipse

Key Algorithms

- The key algorithms that we will be using would be the authentication and registration functions that are from the login/logout servlets. Additionally, the search servlet will be used heavily and will return the search query that was requested by the user. These will heavily utilize the JDBC Driver class to access MySQL. Additionally, the RoomClient, RoomThread, and ServerSocket will be super important in making the multithreading work for the users.

FP- Testing Documentation

Black Box Testing

- *Register button*
  - Directs users to the register page.
- *Login Button*
  - Directs users to login page.
- *Continue as guest button*
  - Directs users to the maps page without needing an account.
- *Map searching* functionality.
  - Outputs the crimes in the area searched.
- *Map filter* functionality.
  - Outputs the crimes in the area searched and filter out the respective information depending on the filter pressed.

White Box Testing

- Login
  - *Valid Credentials Test.* On the login page, they will specify a username and password that is in the database. After pressing login, the user should now be redirected to the main page as it was valid.
  - *Invalid Credentials Test.* Specify a username and password that is not in the database. User should not move from the page, but should be asked to enter valid login credentials. Display error message telling what went wrong to the user.
- Register
  - *Valid Credentials Test*. On the registration page, they will specify a username and password that is not in the database and follows the login criteria. After pressing register, the user should now be redirected to the main page as it was valid.
  - *Invalid Credentials Test.* On the registration page, they will specify a username and password that does not follow the criteria. After pressing register, the user stays on the same page and is displayed an error message.
  - *Taken Credentials Test*. On the registration page, they will specify a username and password that does follow the criteria, but has credentials that already exist. After pressing register, the user stays on the same page and is displayed an error message.
- *User Functionality Test.*
  - After testing the login functionality, a user must be able to save their searches and filter their results. Verify this with checking that the filters work and that searches are saved on their profile.
- *Guest Functionality Test.*

    - Do not login or register as a user. Guests must not be able to filter their results or save their searches. Attempt a filter and ensure they are not able to filter. Attempt a search and make sure that it is not saved.
- Map Search
    - *Valid Zip Code Test.* Input a valid zip code and pull information from the database to be displayed in the map. Should output the map information to be displayed on the page.
    - *Invalid Zip Code Test.* Input an invalid input (e.g. not a valid zip code, words, weird characters, etc) and do not pull information from the database to be displayed in the map. User should not attempt to load new information to be displayed on the page.

Unit Tests

- JDBC.Driver
    - addUser(username, password) → check for both existing and non-existing users
    - deleteUser(username) → check for both existing and non-existing users
    - loginUser(username, password) → check for both existing and non-existing users
    - registerUser(username, password) → check for both existing and non-existing users
    - getCrimes(zipcode) → check with both valid zipcode and invalid input
    - displayHeatMap(zipcode) → check with both valid zipcode and invalid input
- Backend Database Request Test
    - *Large Request Test.* Test large queries to the database for performance time and reliability. Should verify that application can handle larger requests smoothly.
    - *POST Response Test.* Individually test each type of POST that may be returned to the client from the backend. Check that the client never receives malformed data
    - *Invalid Request Test.* Individually test all the types of invalid requests that can be made to the database and ensure uniform behavior when errors occur. Should verify that errors occur rarely and with understandable reasoning (ex. No data for a particular area)
- Parallel Functionality Testing
    - *New Changes Sending Test.* Individually test each type of change (new search, new filter). Should verify that the changes are broadcast to every user.
    - *New Changes Updating Test.* Individually test each type of change (new search, new filter). Should verify that the changes have been received by every user and then displayed on each page.
    - *Multiple Changes Sending Test.* Individually test each type of change (new search, new filter). Have multiple changes done consecutively. Should verify that all changes that are valid still persist and are broadcast to every user.

  - *Multiple Changes Updating Test.* Individually test each type of change (new search, new filter). Have multiple changes done consecutively. Should verify that all changes that are valid still persist and are correctly displayed to every user.
  - *Multiple Users' Spontaneous Changes Sending Test.* Individually test each type of change (new search, new filter). Have multiple changes done by random users nonconsecutively. Should verify that all changes that are valid still persist and are broadcast to every user.
  - *Multiple Users' Spontaneous Changes Updating Test.* Individually test each type of change (new search, new filter). Have multiple changes done by random users nonconsecutively. Should verify that all changes that are valid still persist and are correctly displayed to every user.

Stress Testing

- Out of the scope for this project (Will not being having that many users)

Regression Testing

- Will update if necessary

FP - Deployment Document

1. Download the .zip file for the project
2. Download and setup Tomcat
3. Open the .zip file in Eclipse
4. Import the following libraries under the project
    a. Crime_Data_from_2010_to_Present.csv
    b. Python 3.6.4
    c. pandas version 0.24.2
    d. sqlalchemy version 1.3.1
    e. mysql-connector version 2.1.4
5. Setup the database and reorganize Database to be more easily accessible
6. Change to the connect path of the MySQL in the JDBCDriver class
7. Start the MySQL server
8. Start with the login page