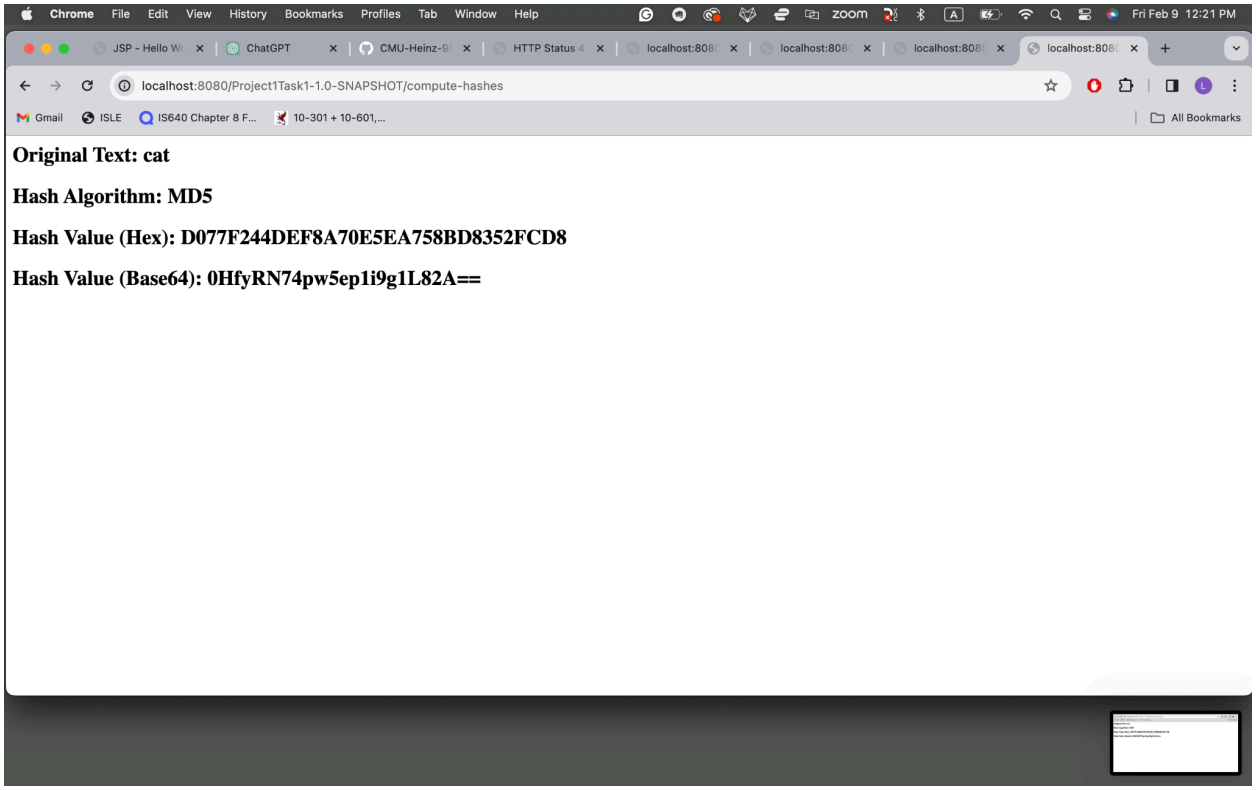
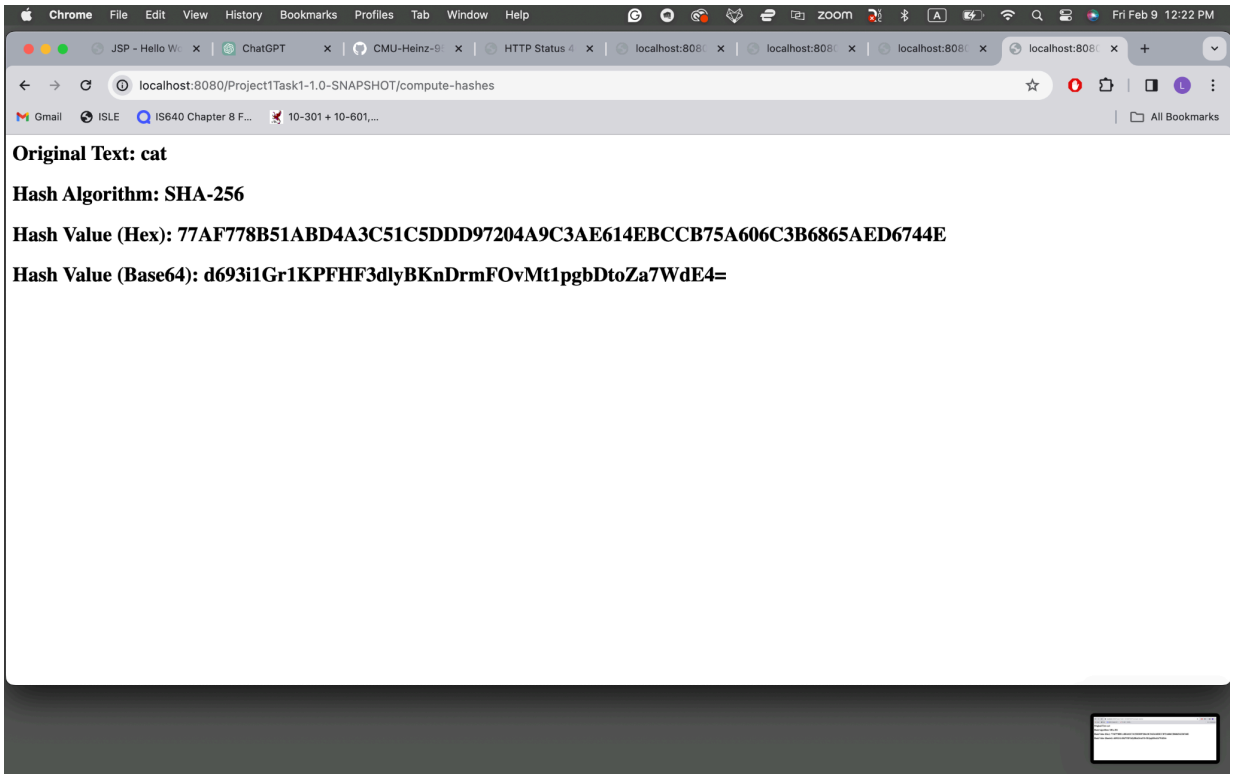
Lawrence Hua
Project 1
Task 1

Code Snippet:

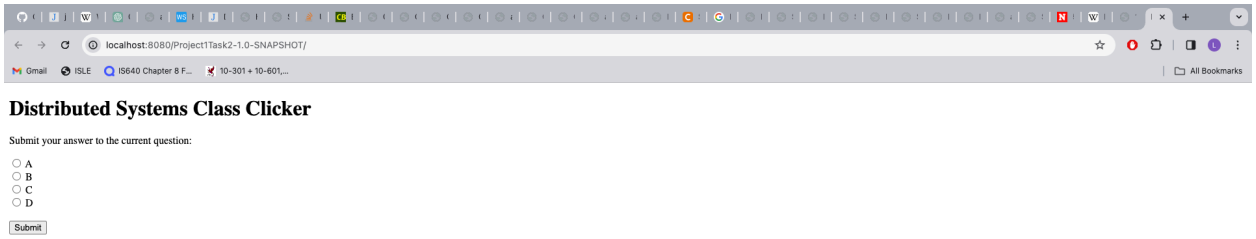
```
private String computeHash(String inputText, String algorithm, String encoding)
{
   //computes the hash of the input text by using the given methods
   try {
       //digest determines which of either hash function to use
       MessageDigest digest = MessageDigest.getInstance(algorithm);
       //hashBytes is the input turned into a byte array using the respected
hash algorithm
       byte[] hashBytes = digest.digest(inputText.getBytes());
       if (encoding.equals("hex")) {
           return DatatypeConverter.printHexBinary(hashBytes);
       } else if (encoding.equals("base64")) {
           return DatatypeConverter.printBase64Binary(hashBytes);
       }
   } catch (NoSuchAlgorithmException e) {
       e.printStackTrace();
   }
   return "Error computing hash!";
}
```

To summarize, compute hash is used by a doPost method within the same file. This doPost is what is returned after the jsp form is submitted. Below are the screenshots of the result screens:

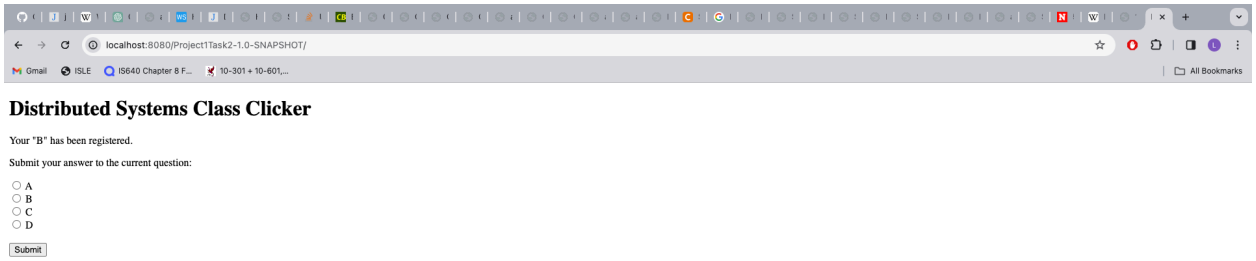

Chrome File Edit View History Bookmarks Profiles Tab Window Help
Fri Feb 9 12:21 PM
localhost:8080/Project1Task1-1.0-SNAPSHOT/compute-hashes
Gmail ISLE IS640 Chapter 8 F... 10-301 + 10-601,...
All Bookmarks
Original Text: cat
Hash Algorithm: MD5
Hash Value (Hex): D077F244DEF8A70E5EA758BD8352FCD8
Hash Value (Base64): 0HfyRN74pw5ep1i9g1L82A==


Chrome File Edit View History Bookmarks Profiles Tab Window Help
Fri Feb 9 12:22 PM
localhost:8080/Project1Task1-1.0-SNAPSHOT/compute-hashes
Gmail ISLE IS640 Chapter 8 F... 10-301 + 10-601,...
All Bookmarks
Original Text: cat
Hash Algorithm: SHA-256
Hash Value (Hex): 77AF778B51ABD4A3C51C5DDD97204A9C3AE614EBCCB75A606C3B6865AED6744E
Hash Value (Base64): d693i1Gr1KPFHF3dlyBKnDrmFOvMt1pgbDtoZa7WdE4=

Project 1
Task 2
Input Screen 1)



Input Screen 2/2)



Output Screen 1)

Distributed Systems Class Clicker

The results from the survey are as follows

A: 1

B: 1

C: 1

D: 1

Output screen 2)

Distributed Systems Class Clicker

The results from the survey are as follows

A: 0

B: 0

C: 1

D: 0

Output Screen 3)

Code Snippet task 2)

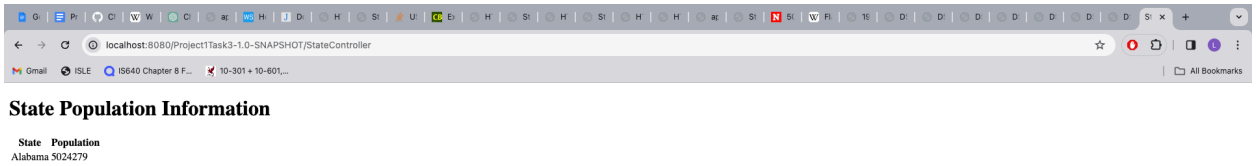
```
<%--
iterates through the map's key and value pairs to display on resultsScreen
page, since we are only allowed a,b,c,d
we will specifiy for each value if it contains a count then display, else
display 0.
However, if the map is null (happens if you just hit submit without hitting any
option), itll print no results.
--%>
<%
   Map<String, Integer> results = (Map<String, Integer>)
request.getAttribute("results");
   if (results != null) {
        int countA = results.containsKey("A") ? results.get("A"):0;
        int countB = results.containsKey("B") ? results.get("B"):0;
        int countC = results.containsKey("C") ? results.get("C"):0;
        int countD = results.containsKey("D") ? results.get("D"):0;
%>
<p>The results from the survey are as follows</p>
<p>A: <%= countA %></p>
<p>B: <%= countB %></p>
<p>C: <%= countC %></p>
<p>D: <%= countD %></p>
<%
} else {
%>
```

```
<p>There are currently no results.</p>
<%}%>
```

Task 3)

Input Screen)

Output Screen)



Task 3 Code Snippet going through census api and outputting it:

```
private static final String CENSUS_API_URL =
"https://api.census.gov/data/2020/dec/pl?get=P1_001N";
public static Map<String, String> getStatePopulation(String selectedState) {
   Map<String, String> statePopulationMap = new HashMap<>();
   try {
       String apiUrl = getAPIURL(selectedState);
       String jsonData = fetchJsonData(apiUrl);
       JsonParser parser = new JsonParser();
       JsonArray dataArray = parser.parse(jsonData).getAsJsonArray();
       //puts data found into a map and returns that map to post
       for (int i = 1; i < dataArray.size(); i++) {
           JsonArray stateData = dataArray.get(i).getAsJsonArray();
           String stateName = getStateNameByFips(stateData.get(stateData.size()
- 1).getAsString());
           String population = stateData.get(0).getAsString();
           statePopulationMap.put(stateName, population);
       }
   } catch (IOException e) {
       e.printStackTrace();
   }
   return statePopulationMap;
}
```

```
%
Map<String, String> statePopulationMap = (Map<String, String>)
request.getAttribute("statePopulationMap");
if (statePopulationMap != null && !statePopulationMap.isEmpty()) { %>
<table>
 <thead>
 <tr>
   <th>State</th>
   <th>Population</th>
 </tr>
 </thead>
 <tbody>
 <% for (Map.Entry<String, String> entry : statePopulationMap.entrySet()) { %>
 <tr>
   <td><%= entry.getKey() %></td>
   <td><%= entry.getValue() %></td>
 </tr>
 <% } %>
 </tbody>
</table>
<% } else { %>
<p>No state population information available.</p>
<% } %>
```

Used chat gpt to generate model java file and state.jsp.
Prompts are as follows:

START OF PROMPT
help me generate the code to:
Create a web app using information about the 50 United States. The information you choose to display is (somewhat) up to you, with the requirements given below.

You app will first let the user choose a state by name using a drop down menu. Next, it will show the user some choices for state data from the list below.

Your app must meet these criteria:

Show the state population using the U.S. Census Bureau's API at https://api.census.gov/data/2020. This API returns very simple JSON data. You'll have to check out their examples to see the query format. See the section below about using JSON.

You must use the gson library to handle JSON records.

Provide other information about the state. This must include two of the state's official things, like the state flower, the state flag, the state bird, and so on. Every state has some number of these things. Besides labeling the thing, it must include a picture of each thing.

You must web scrape this data. See Figure 6 for an example. And you cannot use Flickr for the web scraping.

Figure 6

Figure 6 Pennsylvania State Bird: Ruffed Grouse

Credit:Mad Tinman at https://en.wikipedia.org/

Include at least two other facts about the state, obtained either by scraping or some API.

Add citations for each website you used (see Figure 6), including the Census Bureau.

Be able to handle different user input - in other words, do not show the same information in your output each time - it must be dynamic, not static. It must also be repeatable: if a user enters the same input data another time, the same output should be presented.

Be coded primarily in Java and JSP. If you use something else, give a brief but convincing argument why this was necessary and not just a convenience (like you copied the code from github, or you got ChatGPT to generate it).

Present a web site that requires user interaction using a text field and these: - button - drop-down menu - radio buttons

And use one page for input and at least two web pages for your output.

Use the Model-View-Contorller (MVC) pattern and good separation of concerns

Handle error conditions in a "reasonable" way.

Be well documented - the code, the usage of the app, and the structure/design of the app, any AI-related things as described above.

What sites you use for state is up to you (except for the required Census Bureau data)- but be careful of copying other students' ideas; this must be your own work. The overall design of the web app is up to you (subject to the criteria above). If you have questions about sites, citations, or output, ASK.

Again, you MUST use the MVC pattern for Task 3.

Notes and hints
Screen Scraping
Screen scraping is programmatically processing the HTML that typically is displayed by a browser and can be a useful tool when your data source does not have an API that provides structured data. Instead, you can search or parse the HTML to find and extract the data that you need. For more information, see

https://en.wikipedia.org/wiki/Web_scraping

Your application should work similarly to InterestingPicture, but instead of searching Flickr, it will use the sites mentioned above.

You are allowed to and encouraged to build your solution based on the InterestingPicture code you have been given in class. You MUST refactor it, however, so that it has project, variable, and class names that make sense for your application. For example, you will lose points if your class is still named InterestingPictureServlet.

You do not need to, but you are welcome to, use jsoup (https://jsoup.org/) which is a Java HTML Parser, to do the scraping. It is the Java version of Beautiful Soup, which you may have used in Python. The downsides of using jsoup are you will first need to understand the Document Object Model (DOM) and CSS Selectors. These are both useful to know. The upside of using jsoup is that it makes it much easier to find and select content from HTML (i.e. screen scrape). Refer to the JSON Maven notes below; adding jsoup will require a similar process.

HTML
Refer to http://www.w3schools.com for good help on the basic HTML you need for this task. This has examples of drop-down boxes.

JSON and gson
JSON records are text records containing tag-value pairs, where the tag is the field name - think of it as a dictionary or map with nesting. It is much shorter than XML. In order to find what you need, use the JSON library GSON. To use GSON, download the gson v.2.10.1 jar file to a place you'll remember. To add it to your project, go to File->Project Structure->Modules, choose the Dependencies tab, click the + icon at the bottom choose Jars or Directories, navigate to where you put the jar file, click that, then Apply and OK. It should show up in your pom.xml file as the last entry in as:

```
<dependency>
   <groupId>com.google.code.gson</groupId>
   <artifactId>gson</artifactId>
   <version>2.10.1</version>
</dependency>
```

If this does not appear, add the above lines manually to pom.xml.

Finally, reload the Maven dependencies to have this new dependency take effect - an icon will probably appear in the pom.xml window, but if you don't see it, got to the Project View window, find the pom.xml entry (it should be near the bottom of the tree), right click it, choose Maven -> Reload Project. (FYI, Maven is a build management tool, different from the usual Gradle build; Maven uses the Project Object Model (pom) file to keep track of properties and dependencies.)

Please use gson and not some other JSON library.

SSLHandshakeException
Most modern sites require you to make https, not http requests. When you do so from your Java program, you will hit an SSLHandshakeException. We will be covering SSL and related topics in a few weeks. In the meantime, you will have to deal with this exception.

If you use jsoup, you should use validateTLSCertificates(false). (Refer to the jsoup API to understand this when you need it.)

If you do not use jsoup, here is a code to replace the fetch method in InterestingPictureModel to ignore the exception. The parameter "certType" should be set to the string "TLSV1.3".

```
private String fetch(String searchURL, String certType) {
   try {
      // Create trust manager, which lets you ignore SSLHandshakeExceptions
      createTrustManager(certType);
   } catch (KeyManagementException ex) {
```

```
            System.out.println("Shouldn't come here: ");
            ex.printStackTrace();
        } catch (NoSuchAlgorithmException ex) {
            System.out.println("Shouldn't come here: ");
            ex.printStackTrace();
        }

        String response = "";
        try {
            URL url = new URL(searchURL);
            HttpURLConnection connection = (HttpURLConnection) url.openConnection();

            // Read all the text returned by the server
            BufferedReader in = new BufferedReader(new
InputStreamReader(connection.getInputStream(), "UTF-8"));
            String str;
            // Read each line of "in" until done, adding each to "response"
            while ((str = in.readLine()) != null) {
                // str is one line of text readLine() strips newline characters
                response += str;
            }
            in.close();
        } catch (IOException e) {
            System.err.println("Something wrong with URL");
            return null;
        }
        return response;
    }

    private void createTrustManager(String certType) throws KeyManagementException,
NoSuchAlgorithmException{
        /**
         * Annoying SSLHandShakeException. After trying several methods, finally this
         * seemed to work.
         * Taken from:
http://www.nakov.com/blog/2009/07/16/disable-certificate-validation-in-java-ssl-connections/
         */
        // Create a trust manager that does not validate certificate chains
        TrustManager[] trustAllCerts = new TrustManager[] {new X509TrustManager() {
            public X509Certificate[] getAcceptedIssuers() {
                return null;
            }
            public void checkClientTrusted(X509Certificate[] certs, String authType) {
            }
```

```
        public void checkServerTrusted(X509Certificate[] certs, String authType) {
        }
    }
    };

    // Install the all-trusting trust manager
    SSLContext sc = SSLContext.getInstance(certType);
    sc.init(null, trustAllCerts, new java.security.SecureRandom());
    HttpsURLConnection.setDefaultSSLSocketFactory(sc.getSocketFactory());

    // Create all-trusting host name verifier
    HostnameVerifier allHostsValid = new HostnameVerifier() {
        public boolean verify(String hostname, SSLSession session) {
            return true;
        }
    };

    // Install the all-trusting host verifier
    HttpsURLConnection.setDefaultHostnameVerifier(allHostsValid);
}
```

END OF PROMPT

ChatGPT
To create a web app that meets the provided criteria, we need to follow the MVC (Model-View-Controller) pattern and ensure good separation of concerns. Here's a general outline of how we can structure the code:

Model: This part handles the business logic and data manipulation. It should include methods to fetch data from the Census Bureau's API, scrape information from websites about the states, and handle any other necessary data processing.

View: The view represents the user interface. It should include JSP (JavaServer Pages) files to generate the HTML content that will be sent to the user's browser. These pages will contain the dropdown menu, text fields, buttons, and other elements for user interaction.

Controller: The controller acts as an intermediary between the model and the view. It handles user input, interacts with the model to fetch or manipulate data, and determines which view to render based on the user's actions.

Now, let's break down the steps to implement each component:

Model:

Create a Java class to handle data fetching and scraping. This class will contain methods to interact with the Census Bureau's API and scrape websites for state information. You can use libraries like Gson for JSON parsing and Jsoup for web scraping.
Implement methods to fetch state population data from the Census Bureau's API and scrape information about state symbols (e.g., state flower, state bird) from relevant websites.
Ensure error handling for API requests, web scraping, and any other potential issues.
View:

Develop JSP files to create the user interface. These files will include HTML, CSS, and JavaScript code to render dropdown menus, text fields, buttons, and other UI elements.
Design the pages to display state information dynamically based on user input.
Ensure the layout is user-friendly and responsive across different devices.
Controller:

Implement Java servlets or controllers to handle user requests and manage the flow of data between the model and the view.
Define routes or mappings to map incoming requests to specific servlets or controller methods.
Parse user input, call appropriate methods in the model layer to fetch data, and pass the retrieved data to the view for rendering.
Handle exceptions, errors, and edge cases gracefully to provide a smooth user experience.
Additional considerations:

Document the code thoroughly, including comments and javadocs for methods, classes, and important code blocks.
Follow best practices for coding standards, naming conventions, and project structure.
Test the application thoroughly to ensure it functions correctly and handles various scenarios.
Include citations for any external sources or libraries used in the project, as per the provided requirements.
This is a high-level overview of how you can approach building the web app. Each component will require further implementation details, but this should give you a solid starting point. Let me know if you need further assistance with any specific aspect of the implementation!

After this, I carried on debugging my code. Unfortunately, I ran out of time and would like to just submit what I have for task 3. I believe I got most of the information set in this except the 4 fun facts and pictures.