95702: Distributed Systems

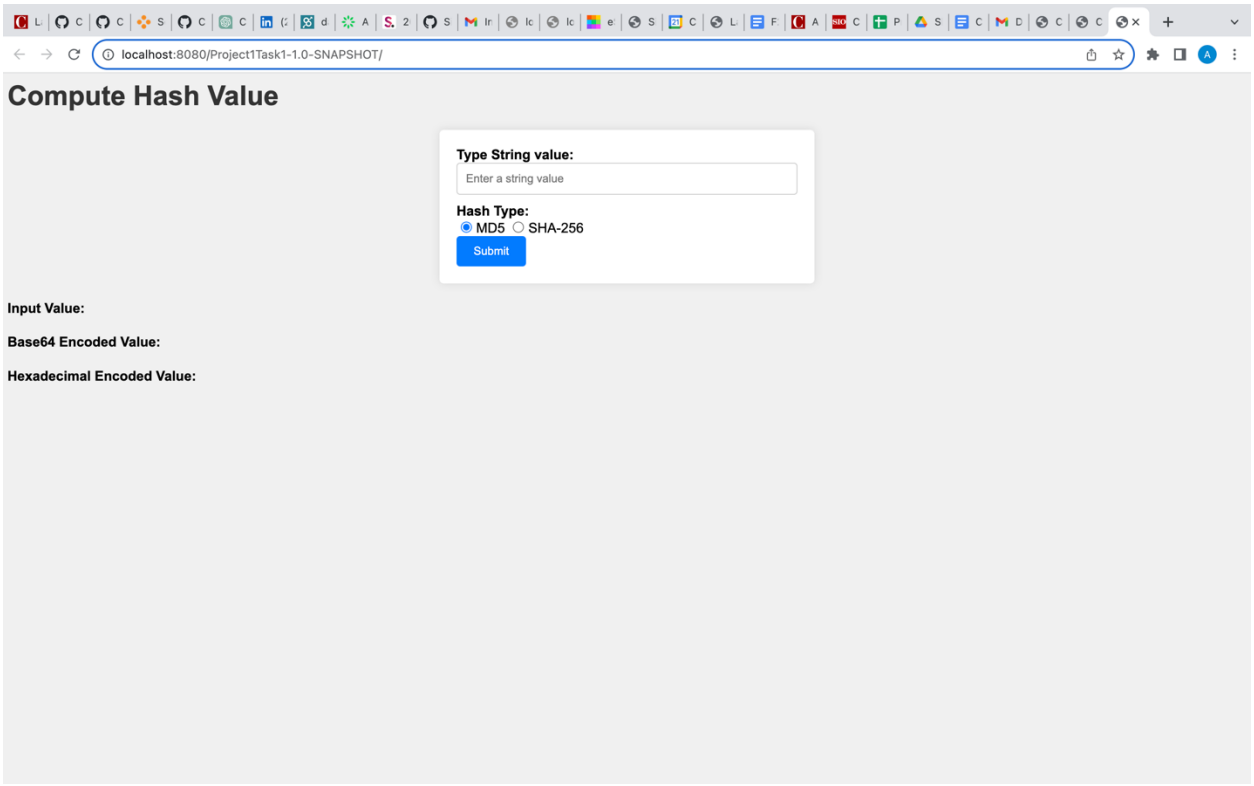
Project 1

Name: Ariane Correa
Andrew ID: ajcorrea

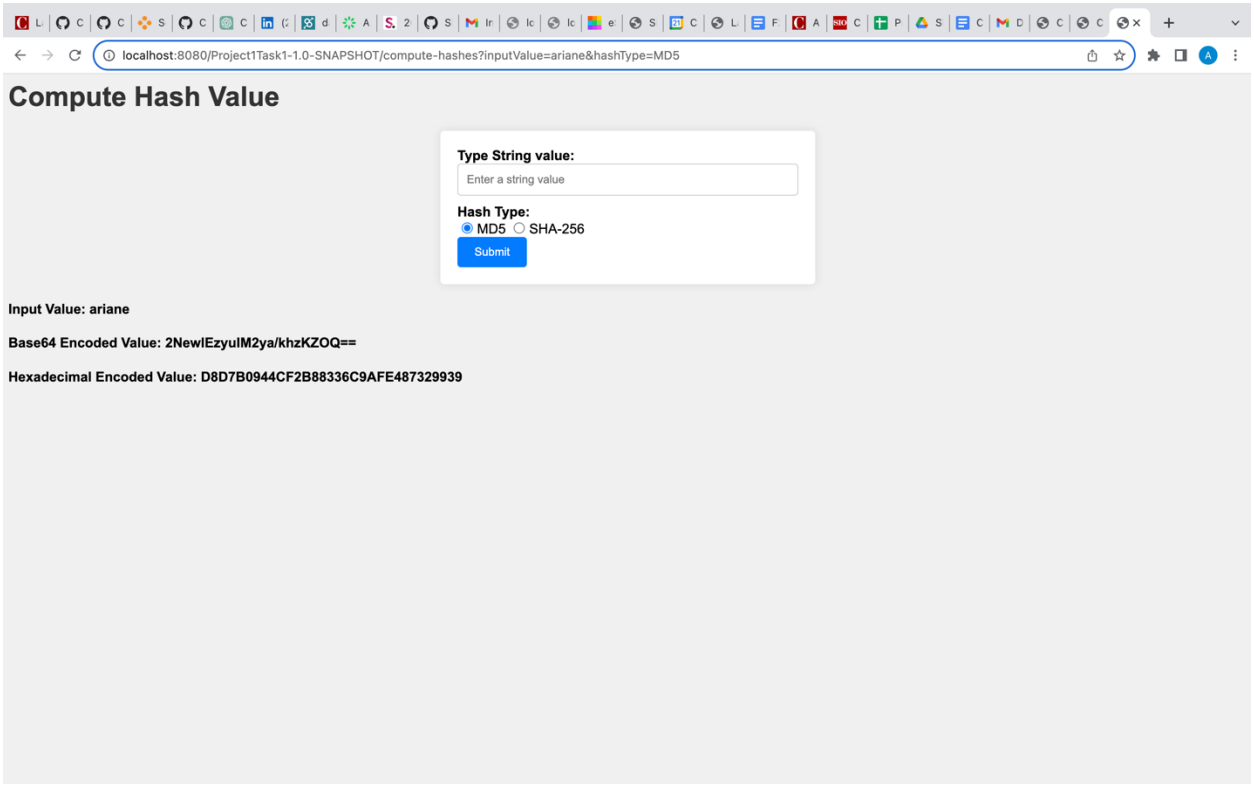
Task 1

Screenshots:

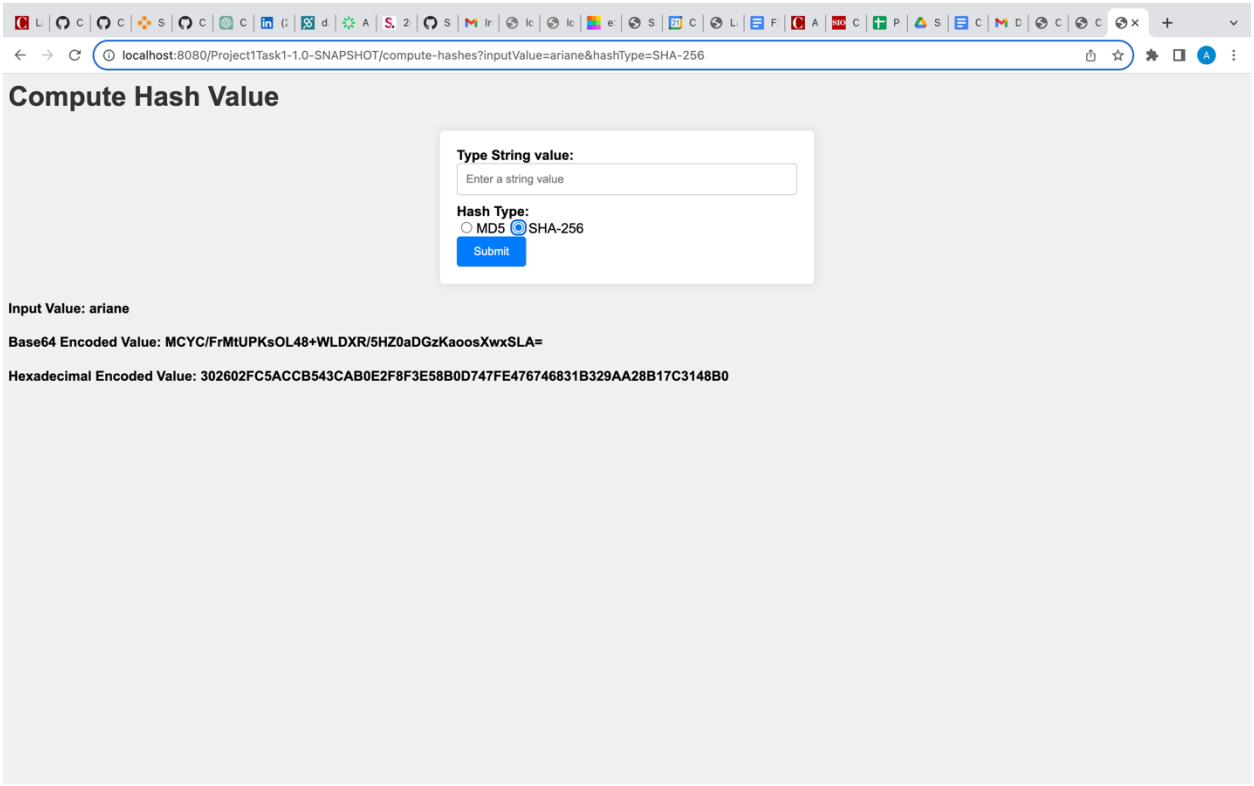
1. Main Page (Before we enter string value)



2. Results for MD5:
   (We enter Type String Value= Ariane, select Hash Type=MD5 and click Submit)

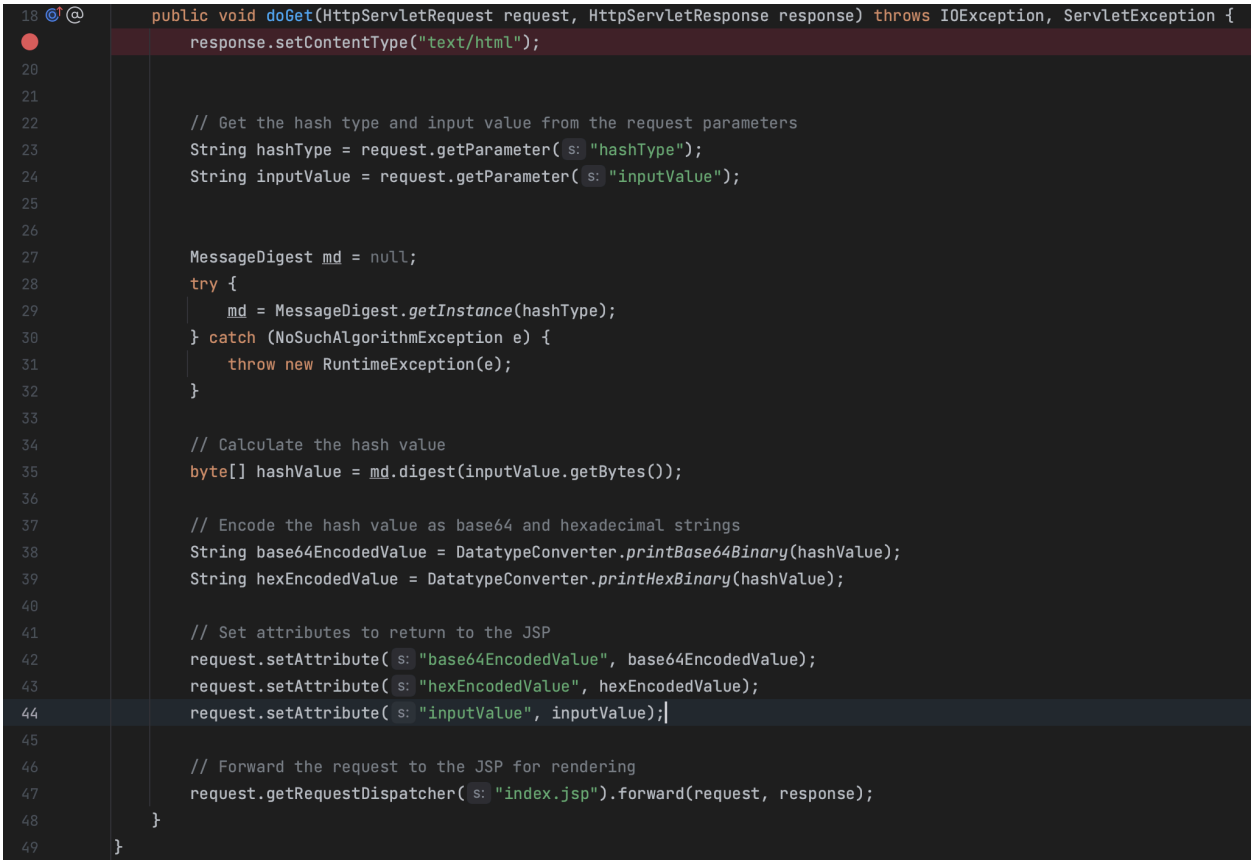


3. Results for SHA-256:
4. (We enter Type String Value= Ariane, select Hash Type=SHA-256 and click Submit)



Code Snippets:

Code from *ComputeHashes.java*: This `doGet` method in a servlet handles HTTP GET requests. It first retrieves the user's input, including the hash type and a string value, from the request parameters. Then, it calculates the hash value of the input string using the specified hash algorithm (e.g., MD5 or SHA-256) and encodes this hash value as both base64 and hexadecimal strings. These encoded values, along with the original input string, are set as attributes to be displayed in the JSP page. Finally, the method forwards the request to the "index.jsp" page for rendering, where the calculated hash values and original input are displayed to the user.



```
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws IOException, ServletException {
        response.setContentType("text/html");



        // Get the hash type and input value from the request parameters
        String hashType = request.getParameter( s: "hashType");
        String inputValue = request.getParameter( s: "inputValue");


        MessageDigest md = null;
        try {
            md = MessageDigest.getInstance(hashType);
        } catch (NoSuchAlgorithmException e) {
            throw new RuntimeException(e);
        }

        // Calculate the hash value
        byte[] hashValue = md.digest(inputValue.getBytes());

        // Encode the hash value as base64 and hexadecimal strings
        String base64EncodedValue = DatatypeConverter.printBase64Binary(hashValue);
        String hexEncodedValue = DatatypeConverter.printHexBinary(hashValue);

        // Set attributes to return to the JSP
        request.setAttribute( s: "base64EncodedValue", base64EncodedValue);
        request.setAttribute( s: "hexEncodedValue", hexEncodedValue);
        request.setAttribute( s: "inputValue", inputValue);

        // Forward the request to the JSP for rendering
        request.getRequestDispatcher( s: "index.jsp").forward(request, response);
    }
}
```

95702: Distributed Systems

Project 1

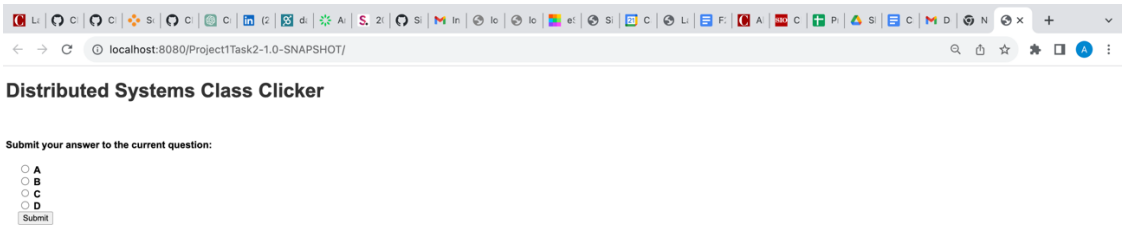
Name: Ariane Correa
Andrew ID: ajcorrea

Task 2

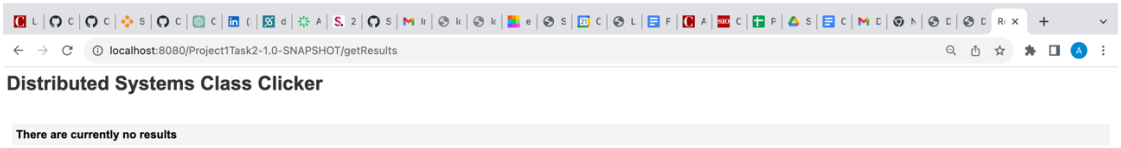
Link to Main Page: http://localhost:8080/Project1Task2-1.0-SNAPSHOT/
Link to Result Page: http://localhost:8080/Project1Task2-1.0-SNAPSHOT/getResults

Screenshots: Desktop View

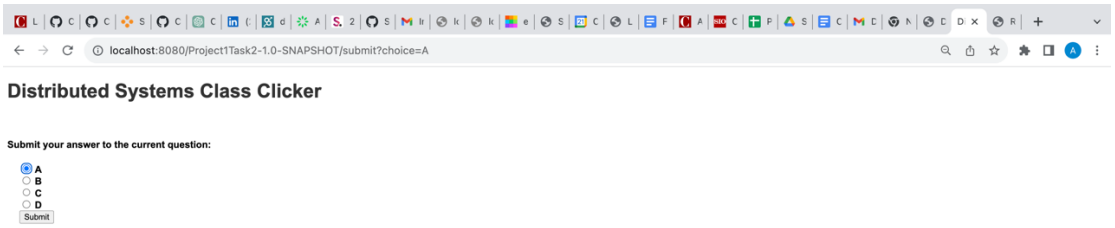
1. Application with the Answer Options (Desktop View)



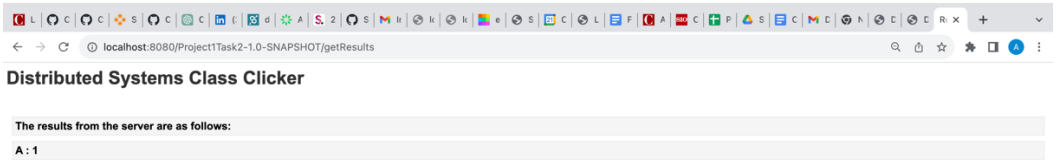
2. Application Result Page (Desktop View)



3. Application when we select a particular option (Here, we select A) and click Submit

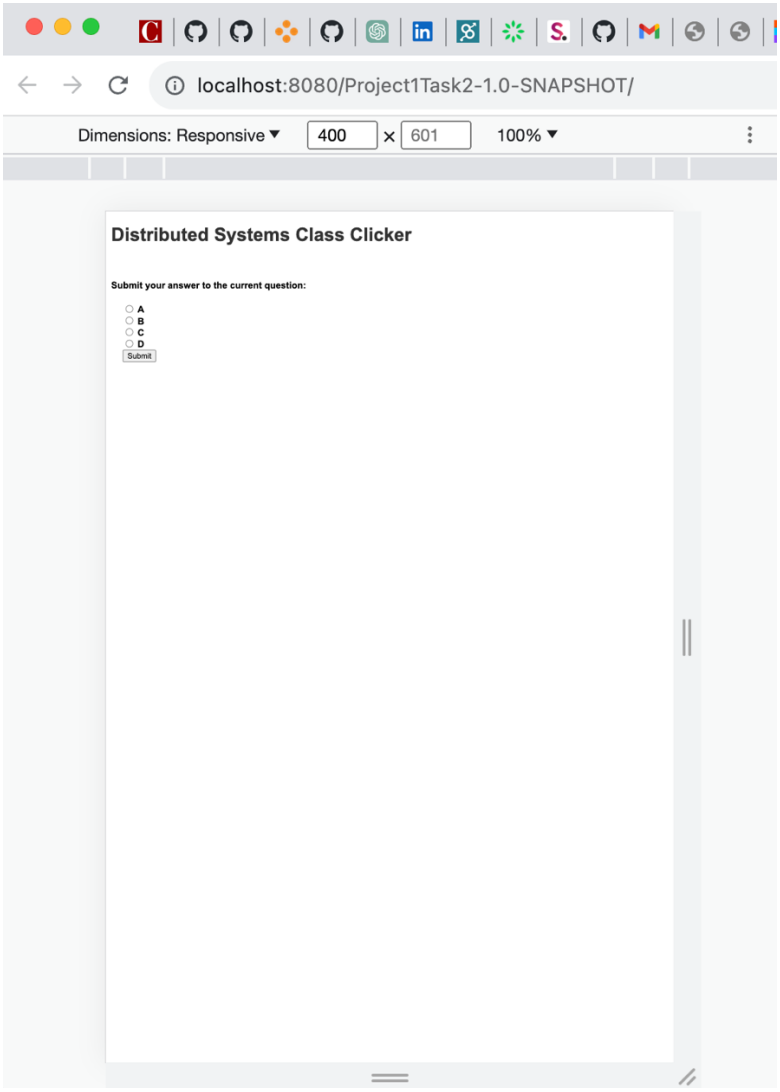


4. Updated Application Result Page (Desktop View)

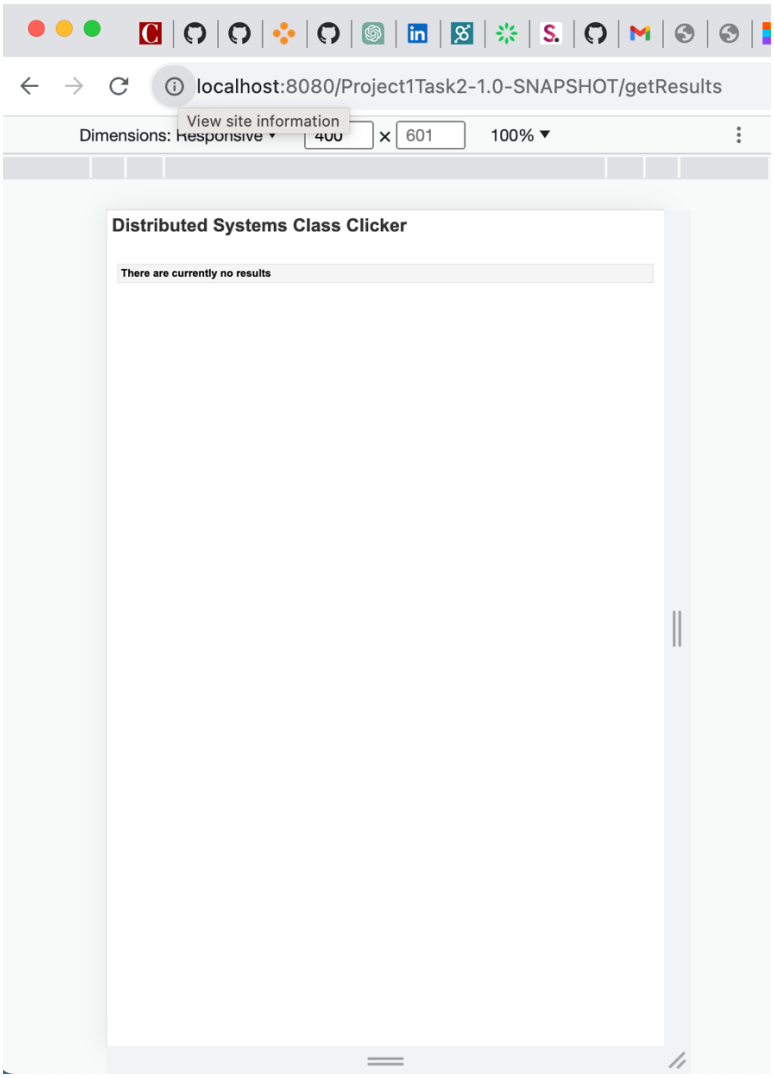


Screenshots: Mobile View

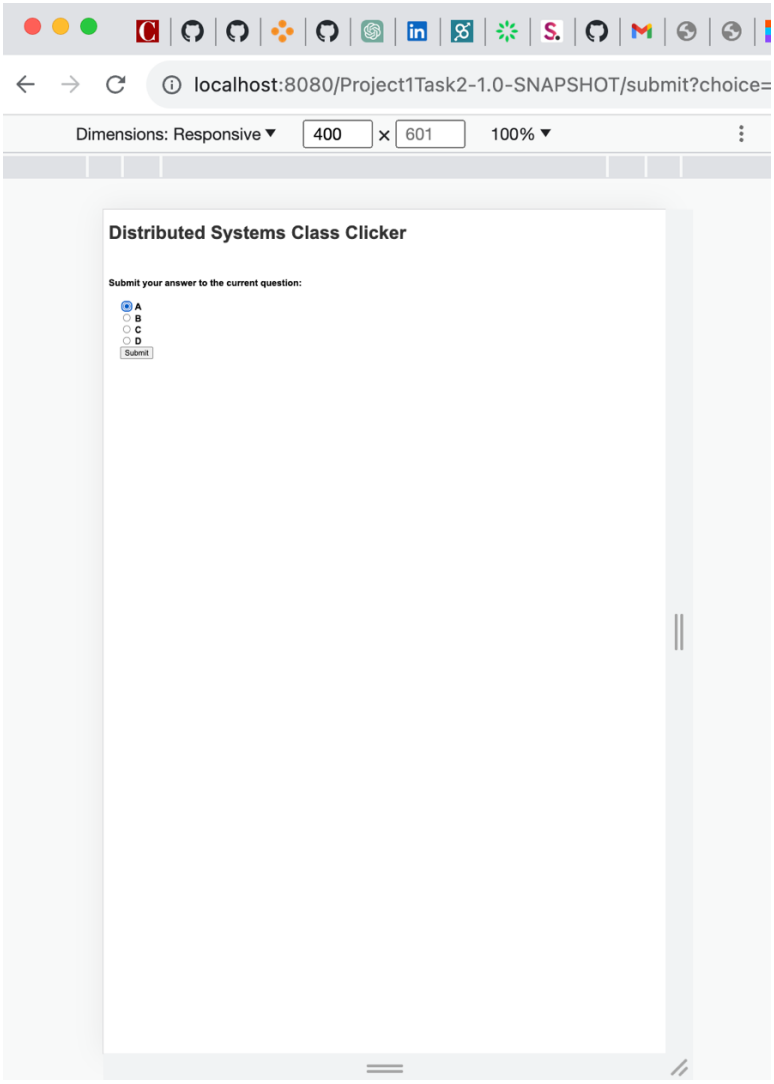
1. Application with the Answer Options (Mobile View)



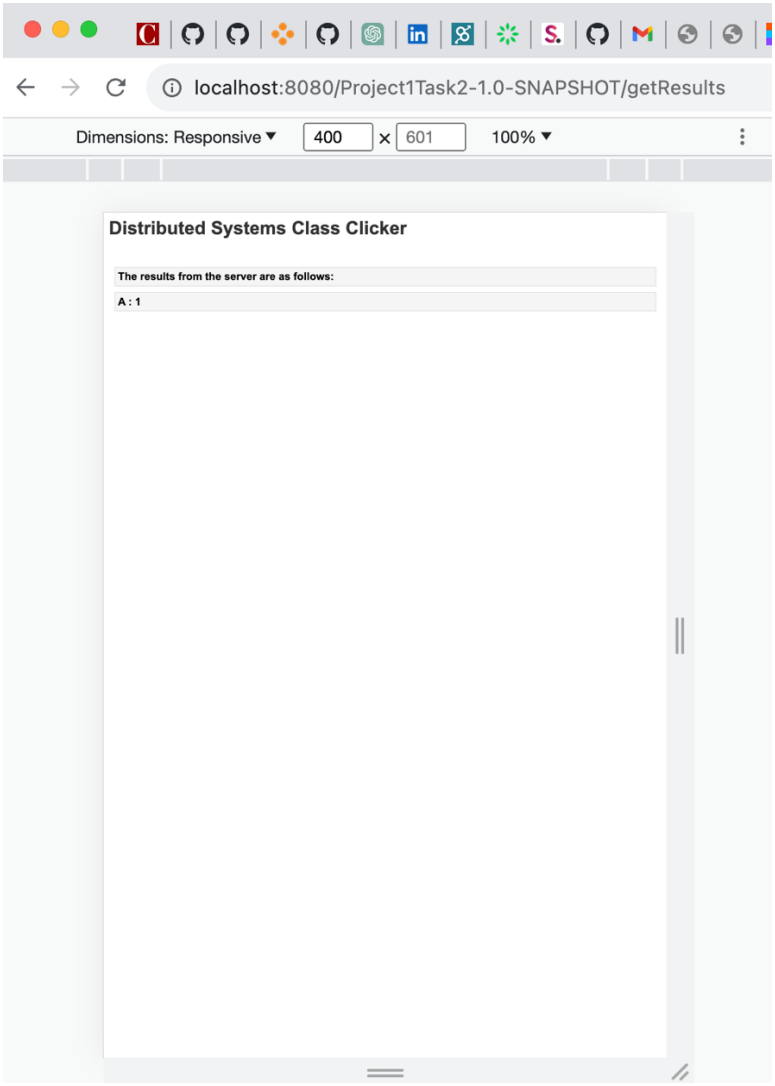
2. Application Result Page (Mobile View)



3. Application when we select a particular option (Here, we select A) and click Submit



4. Updated Application Result Page (Mobile View)



Code Snippets:

Code from *ClickerScorer.java*: Here, we create and maintain a TreeMap that stores values entered by the User in ascending order.

```java
public class ClickerScorer {

    // A TreeMap is used to maintain key values in ascending order
    2 usages
    private Map<String,Integer> score = new TreeMap<>();


    2 usages
    public  Map<String, Integer> getScore() { return score; }

    3 usages
    public void setScore(Map<String, Integer> score) { this.score = score; }
}
```

Code from *ClickerMain.java*: Here, we make necessary initializations of the new object. The String outputText is initialized since we need to send it to the .jsp files for display on the application.

```java
@WebServlet(name = "clicker-controls", urlPatterns = {"/submit", "/getResults"})
public class ClickerMain extends HttpServlet {
    3 usages
    private String outputText;
    5 usages
    private ClickerScorer clickerScorer = new ClickerScorer();

    public void init() {
        outputText = "";
        clickerScorer.setScore(new HashMap<>());
    }
```

Code from *ClickerMain.java*: The logic for producing the results.

- Handling the "/submit" Path: When the servlet receives a request with the "/submit" path, it processes the user's choice and updates or adds it to the maintained map. Additionally, it sends the "outputText" attribute with the User-selected answer back to the JSP file for display.
- Handling the "/getResults" Path: In case the servlet receives a request with the "/getResults" path, it prepares the map object containing the selected options for display. It then forwards the request to a new results JSP page, which

showcases the various selected options. Importantly, this step also clears the map to ensure it maintains new scores for subsequent requests.

```java
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws IOException, ServletException {
        response.setContentType("text/html");

        // If the path is "/submit"
        if (request.getServletPath().equals("/submit")) {
            String choice = request.getParameter( s: "choice");

            // Create an output message for the user
            outputText = "Your \"" + choice + " \" has been registered";
            Map<String, Integer> score = clickerScorer.getScore();

            // Get the value of the "choice" parameter and update the map accordingly.
            // If the value is already contained in the map, update its count; otherwise, add a new entry.
            score.put(choice, score.containsKey(choice) ? score.get(choice) + 1 : 1);

            clickerScorer.setScore(score);
            request.setAttribute( s: "outputText", outputText);
            request.getRequestDispatcher( s: "index.jsp").forward(request, response);

        }

        // If the path is "/getResults"
        if (request.getServletPath().equals("/getResults")) {

            //Set map attribute as object and retrieve in the servlet page
            request.setAttribute( s: "choiceMap", clickerScorer.getScore());
            request.getRequestDispatcher( s: "result.jsp").forward(request, response);
            clickerScorer.setScore(new HashMap<>());
        }

    }
```

Code from *result.jsp*: Code that handles Display in the “/getResults” page.
If the map is empty, the first message is displayed. Otherwise, the JSP page iterates through the map and displays all key-value pairs. Since the Map object used is a TreeMap, the values are automatically presented in ascending order.

```jsp
</head>
<body>
<h1>Distributed Systems Class Clicker</h1>
<%
    Map<String, Integer> choiceMap = (HashMap<String, Integer>) request.getAttribute("choiceMap");
    if (choiceMap.isEmpty()) {
        out.print("<br/>");
        out.print("<h3> There are currently no results </h3>");
    } else {
        out.print("<br/>");
        out.print("<h3> The results from the server are as follows: </h3>");
        for (Map.Entry<String, Integer> entry : choiceMap.entrySet()) {
            out.print("<h3>" + entry.getKey() + " : " + entry.getValue() + "</h3>");
        }
    }
%>
</body>
</html>
```

95702: Distributed Systems

Project 1

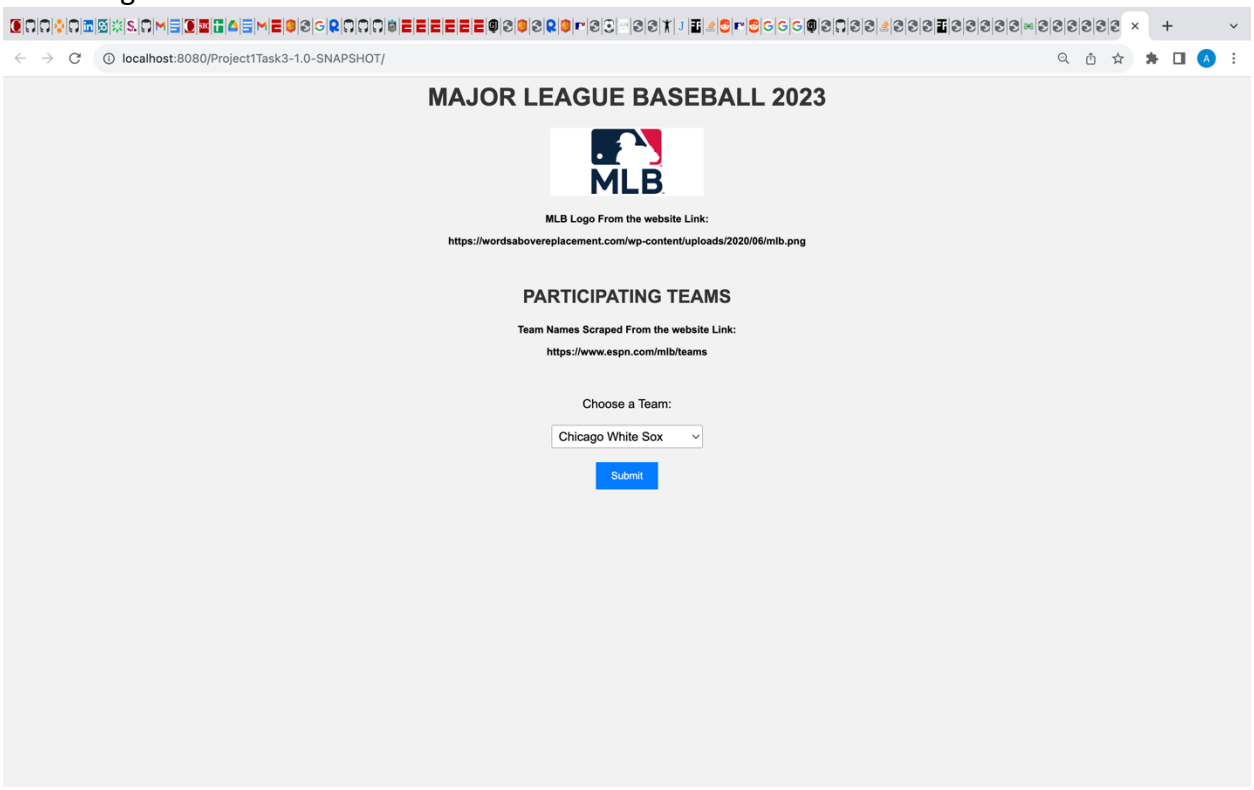
Name: Ariane Correa
Andrew ID: ajcorrea

Task 3

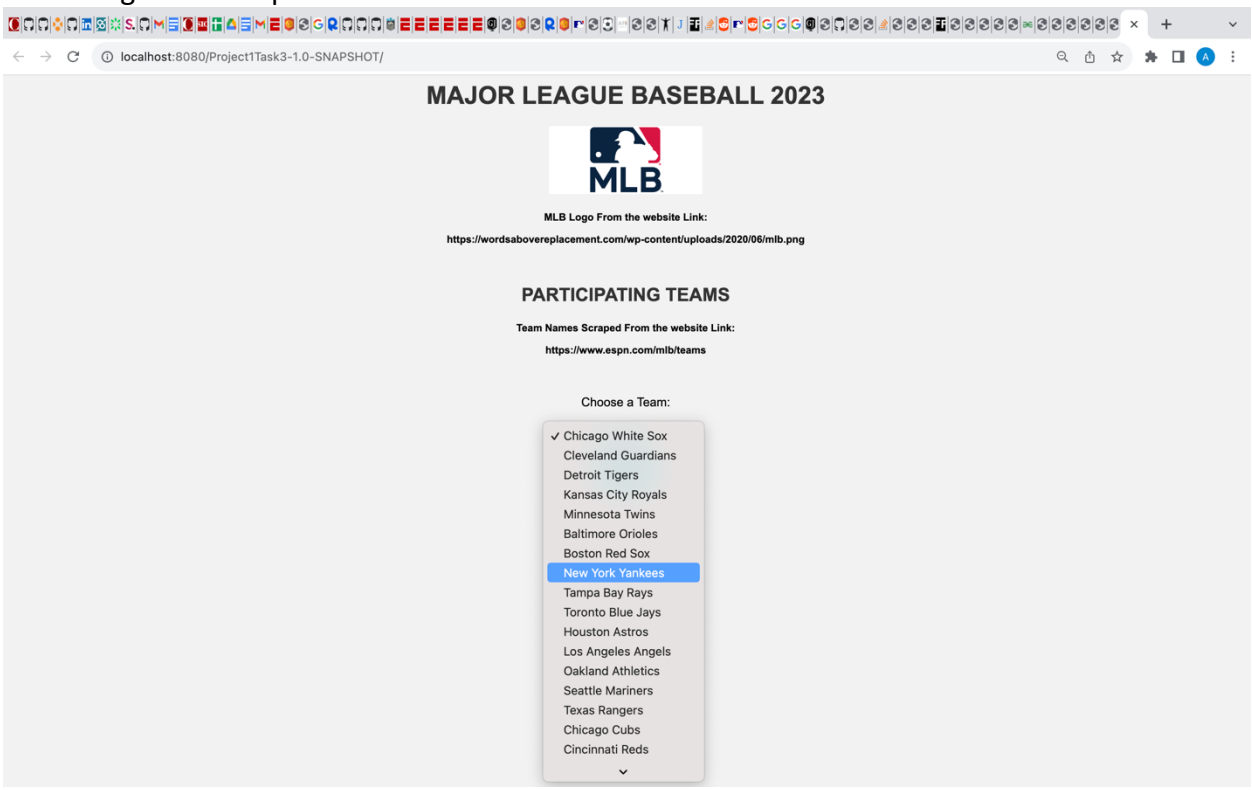
URLS Scraped:
website 1: for list of MLB team names:
https://www.espn.com/mlb/teams
website 2: for MLB team logos:
https://www.sportslogos.net/teams/list_by_year/42023/2023_MLB_Logos/
website 3: for mlb team home stadium info:
https://geojango.com/pages/list-of-mlb-teams
api 1: for MLB team information:
https://statsapi.mlb.com/api/v1/teams
api 2: for mlb team top hitter stats:
https://statsapi.mlb.com/api/v1/stats?stats=season&group=hitting

Screenshots:

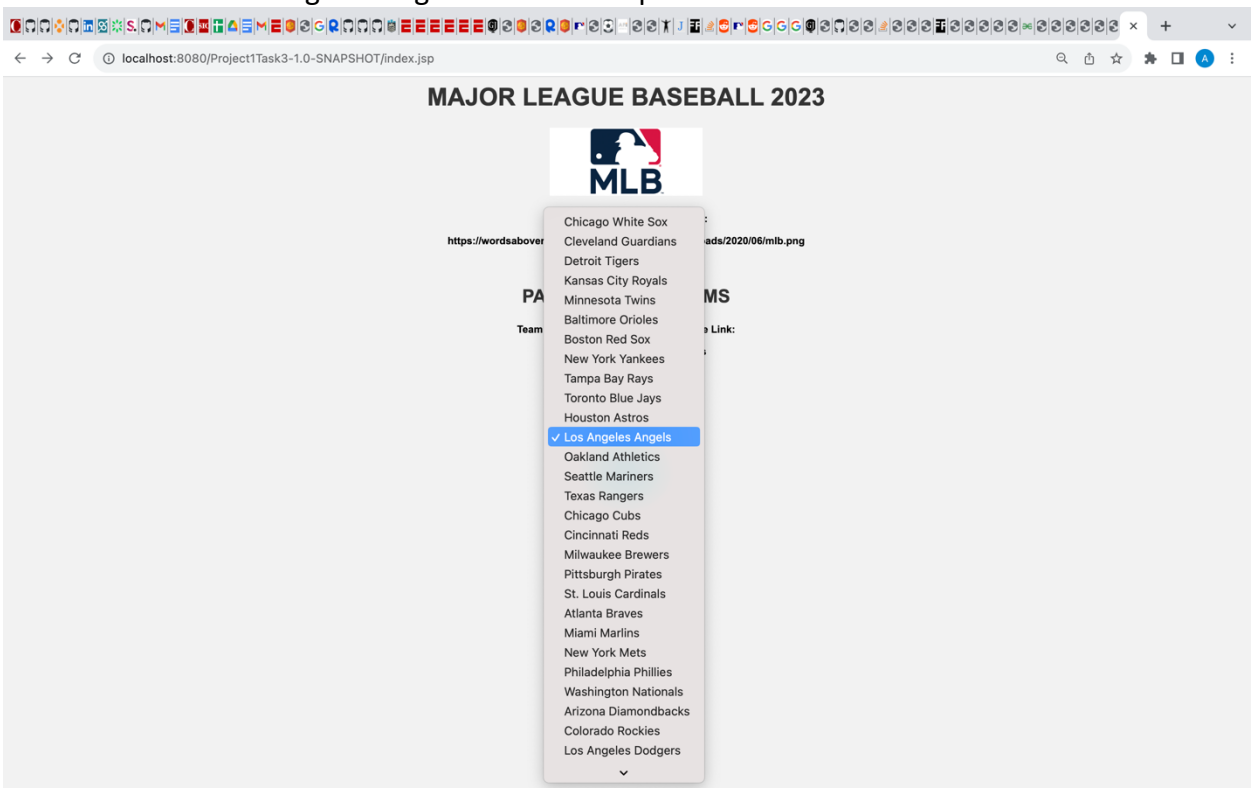
1. Home Page



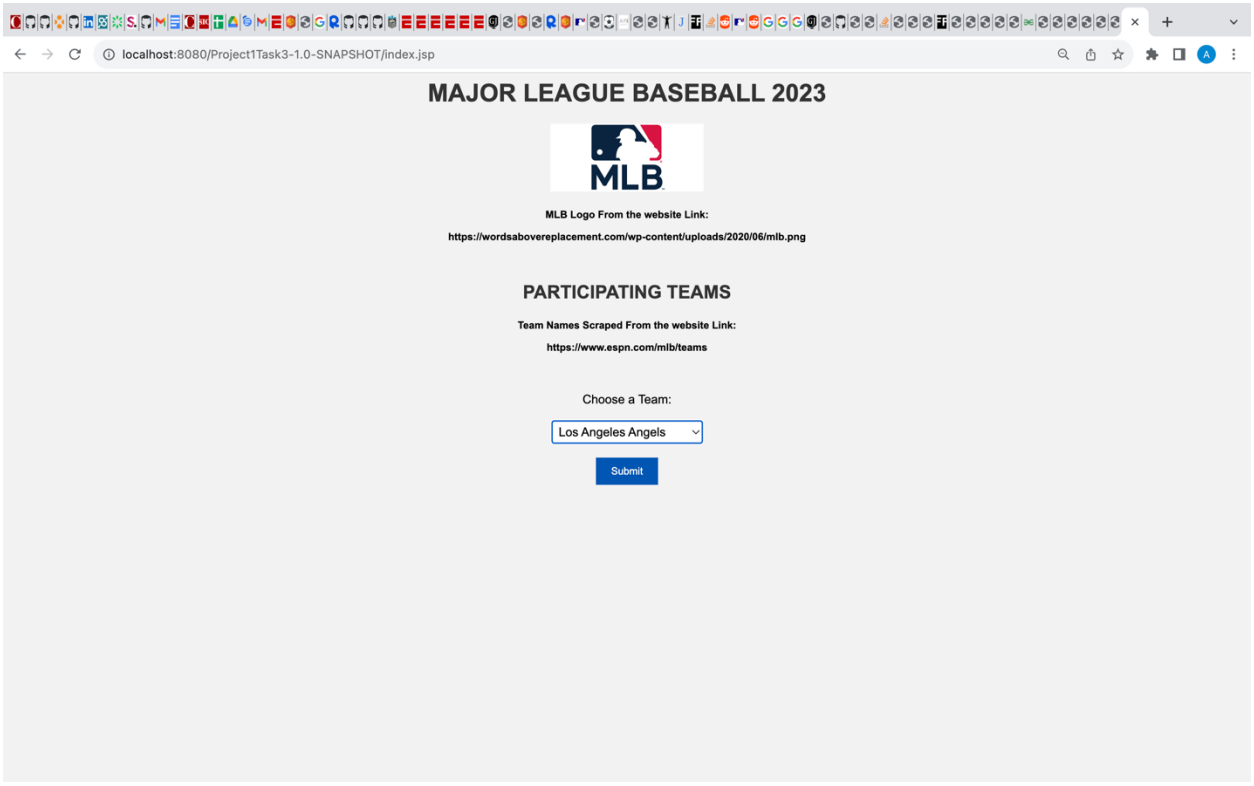
2. Home Page with Dropdown menu to select Team



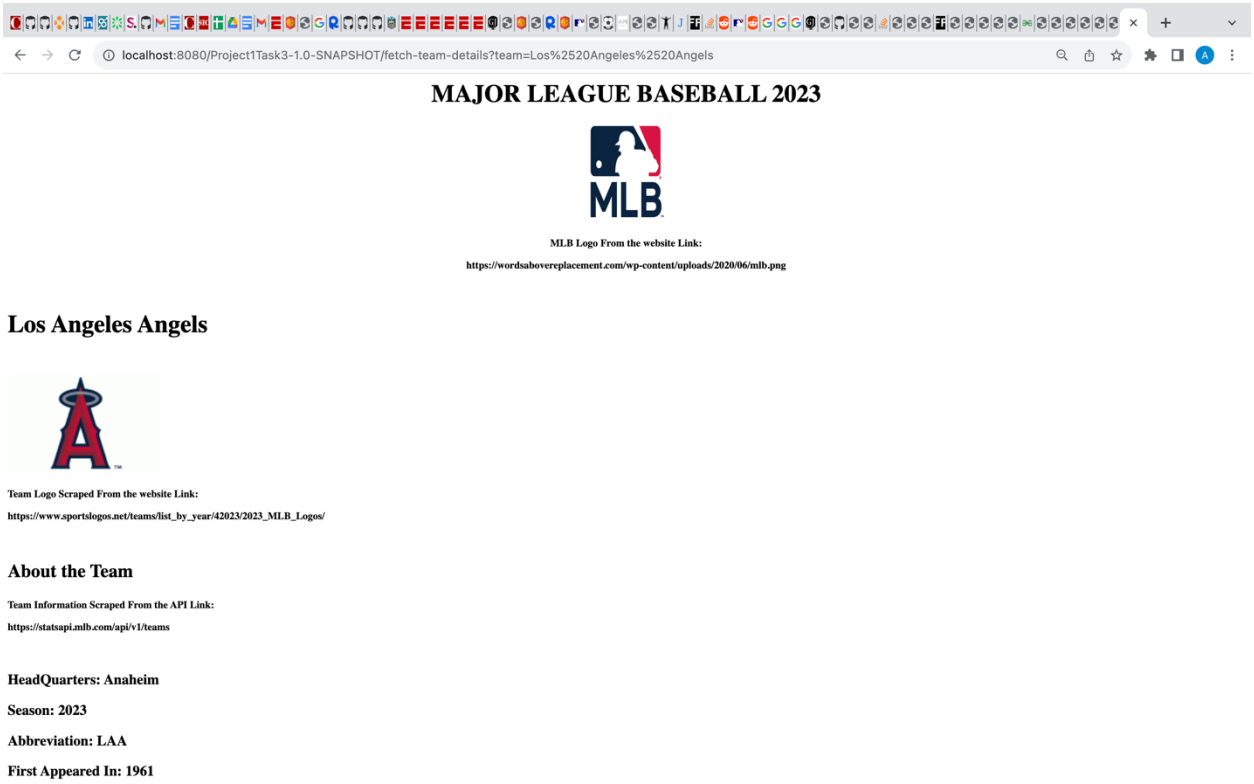
3. We Select Team Los Angeles Angels from the dropdown



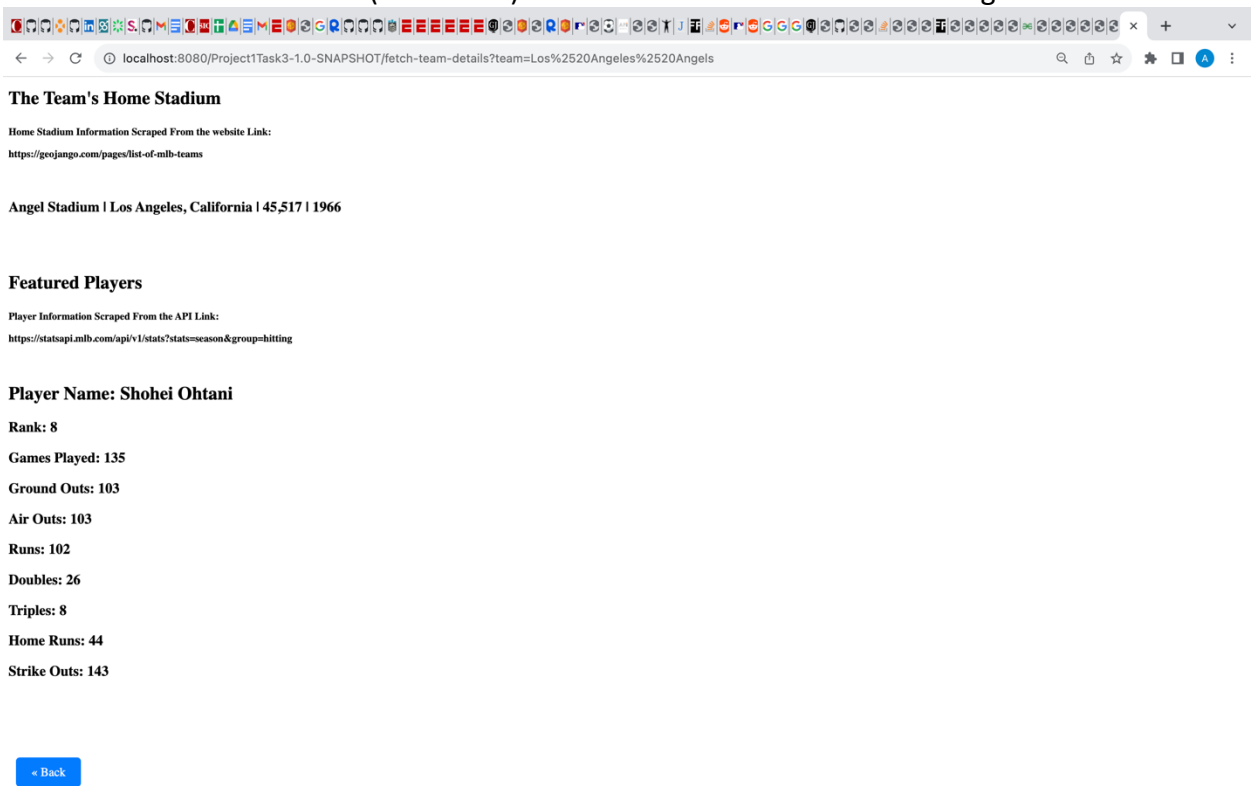
4. We hit Submit



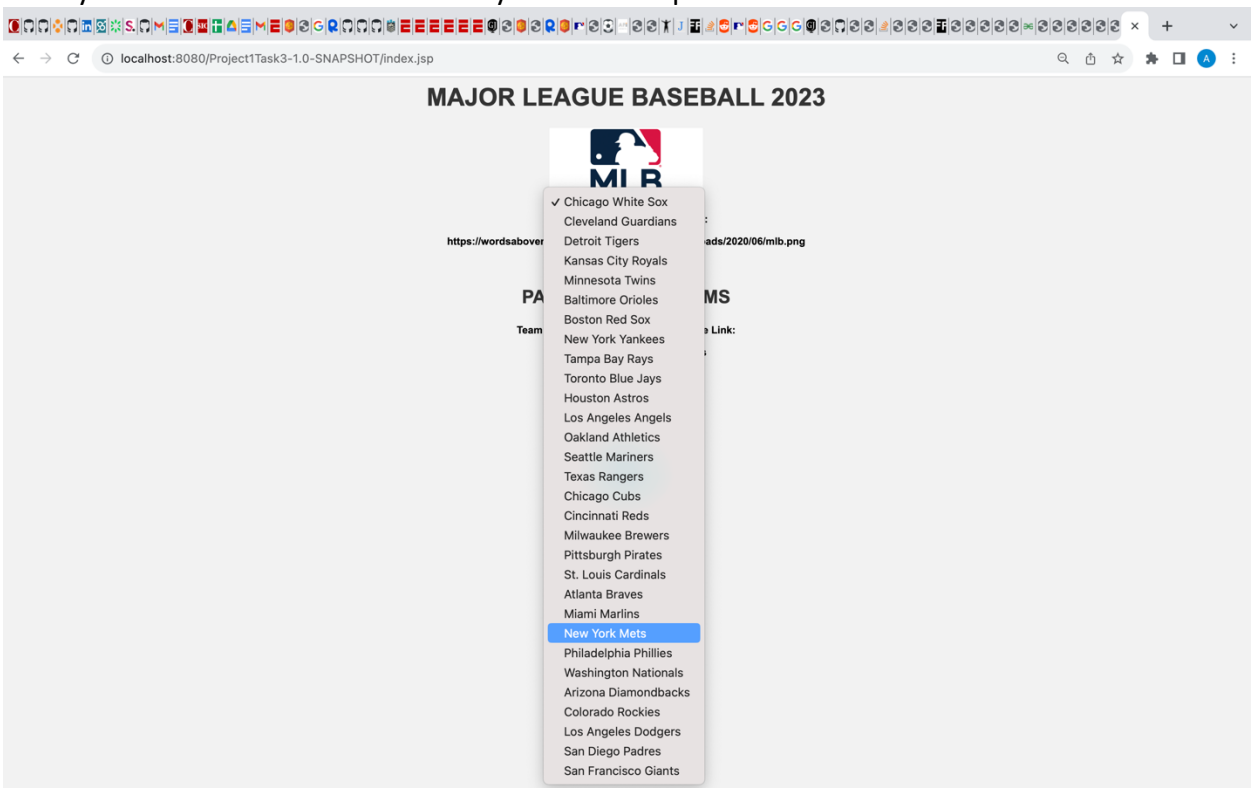
5. Results – Team Information



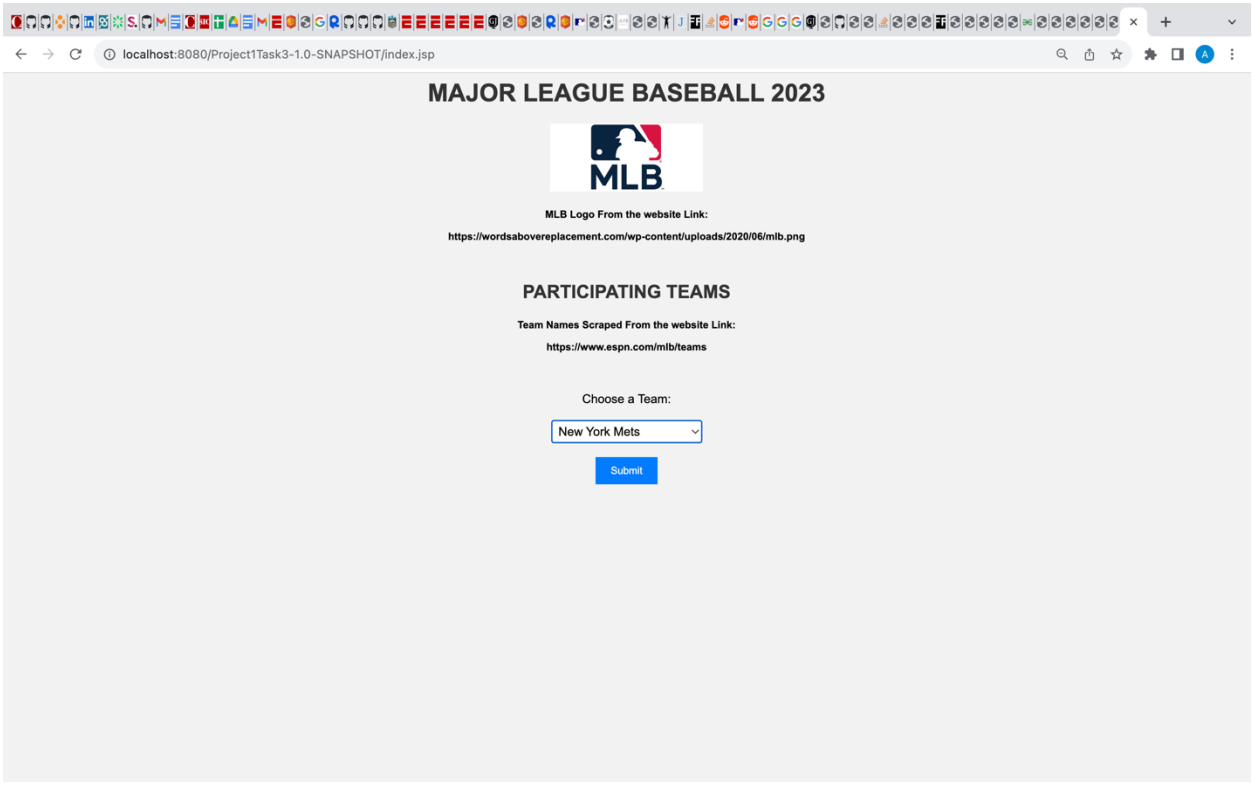
6. Results – Team Information (continued) with a back button to the Home Page



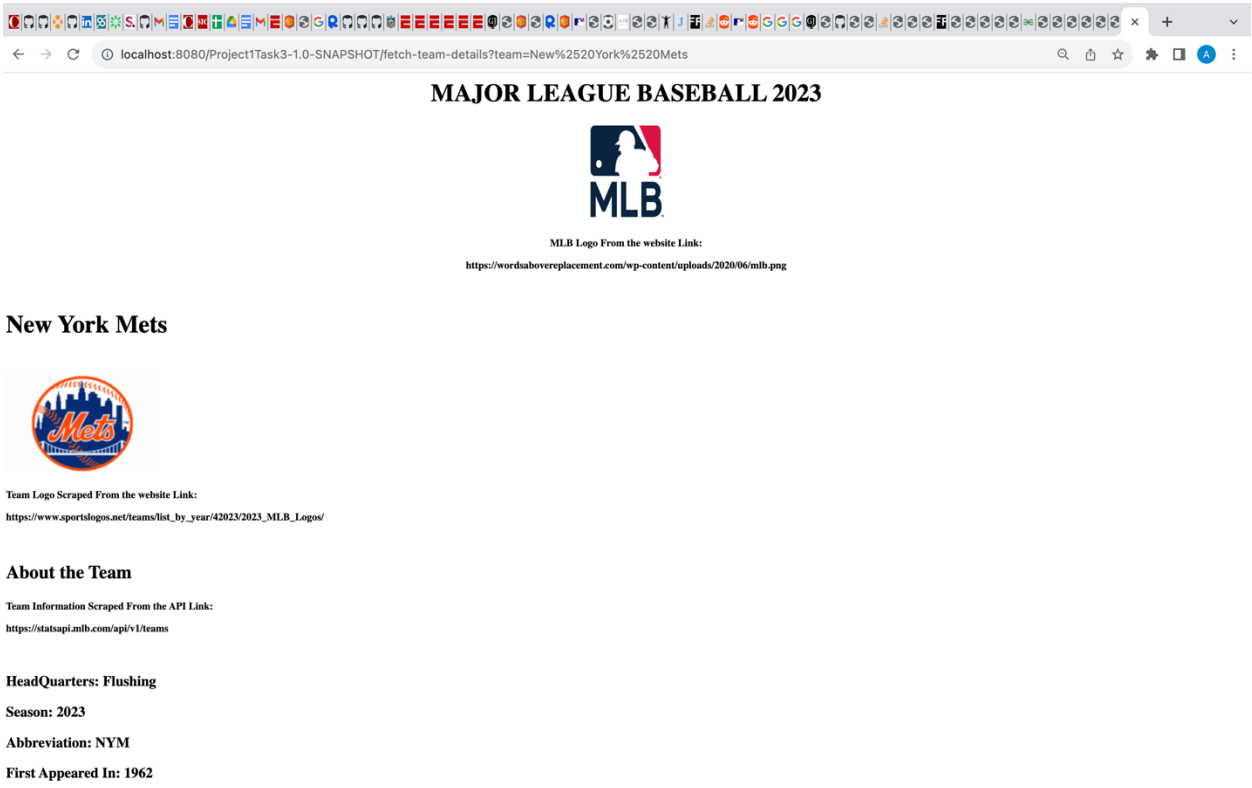
7. We try for another team so as to verify results are specific and accurate



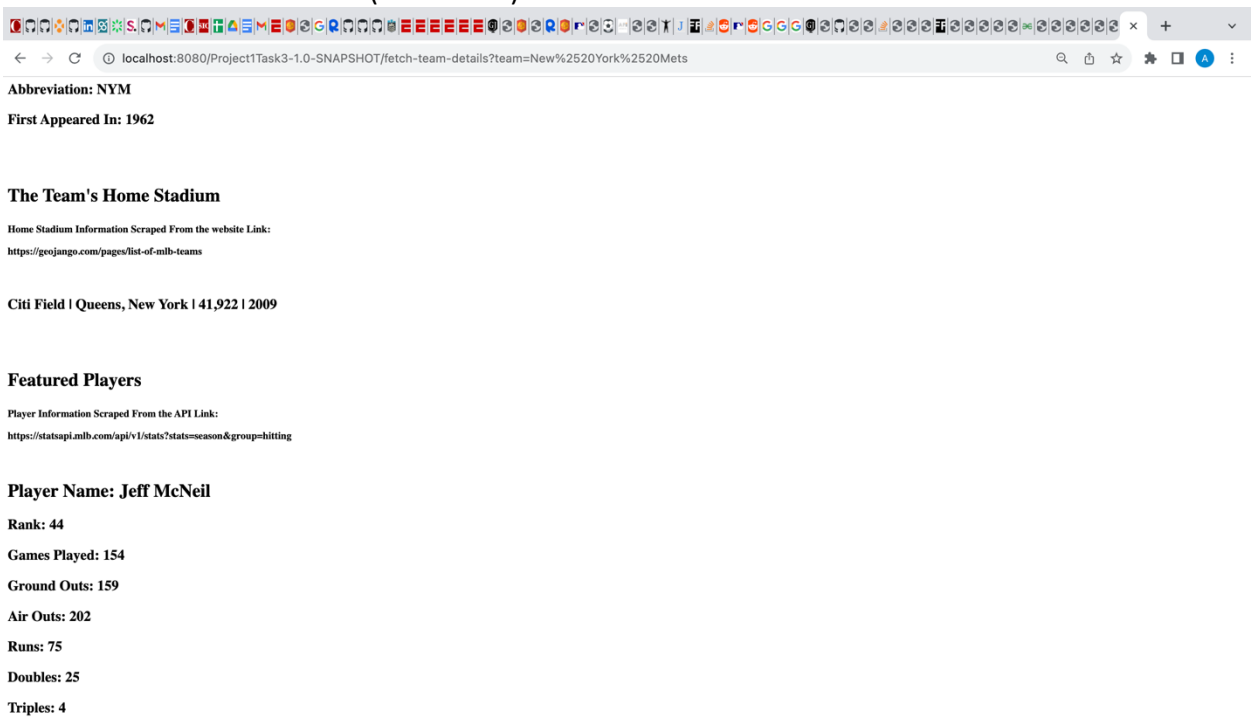
8. We Hit Submit



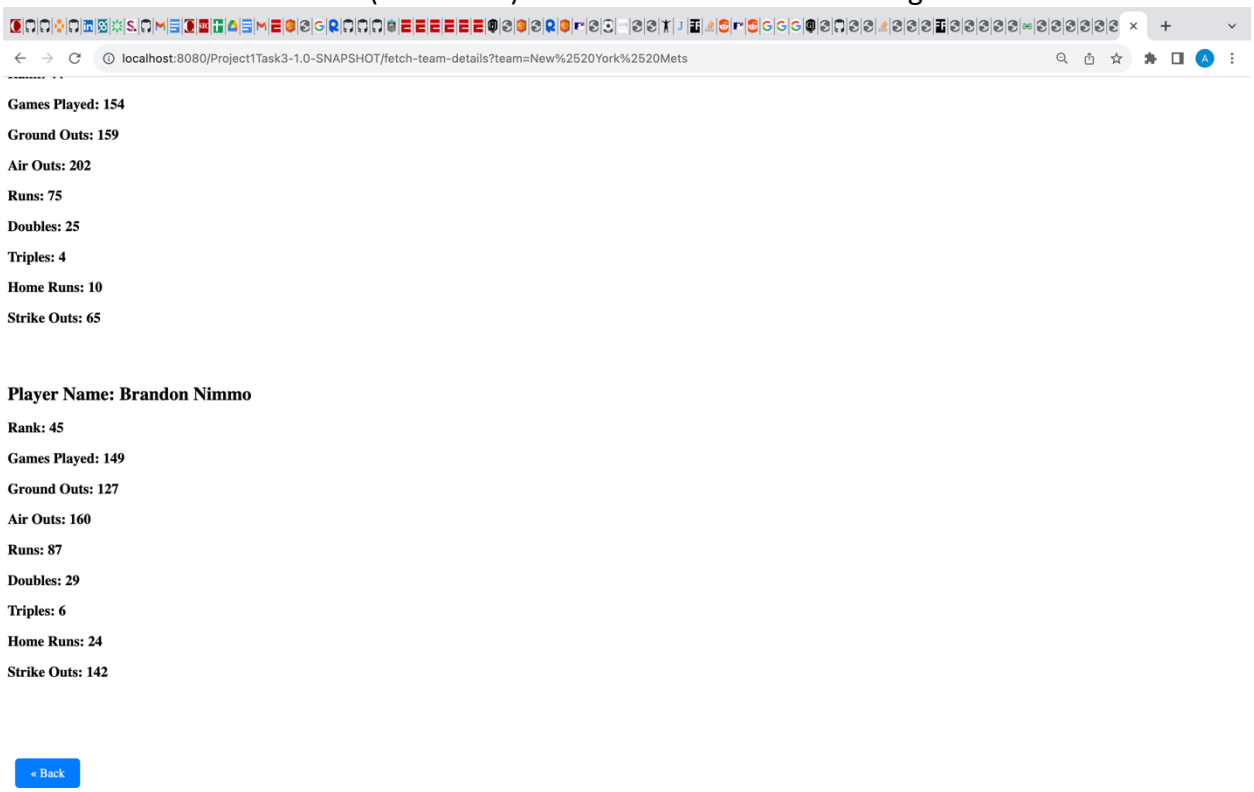
9. Results – Team Information



10. Results – Team Information (continued)



11. Results – Team Information (continued) with Back button to Home Page



Code Snippets:

1. This method scrapes team names from the URL: https://www.espn.com/mlb/teams and adds the values extracted to a text file called teams.txt which is used for the dropdown and also for further scraping purposes such as comparing other extracted values with team name values that we have to ensure we only gather data for MLB teams.

```
41      void ScrapeTeamName() throws IOException {
42          // Use Jsoup to fetch HTML data into a document
43          Document doc = Jsoup.connect(teamNameScrapperUrl).get();
44
45          // Select the HTML elements that contain team names
46          Elements elem = doc.select( cssQuery: ".TeamsWrapper");
47          Elements table = elem.select( query: ".ContentList__Item");
48
49          // Define the path for the output text file
50          String outputPath = "teams.txt";
51          Path path = Paths.get(outputPath);
52
53          try (FileWriter writer = new FileWriter(path.toFile())) {
54              // Loop through the HTML elements and extract team names
55              for (Element e : table) {
56                  Element h2 = e.select( cssQuery: "h2").first();
57                  if (h2 != null) {
58                      String teamName = h2.text();
59                      // Write the team name to the output text file
60                      writer.write( str: teamName + "\n");
61                  }
62              }
63              // Notify the user when team names have been successfully scraped and saved to teams.txt
64              System.out.println("Scraped data has been saved to teams.txt");
65          } catch (IOException e) {
66              // Handle any IOException that may occur during file writing
67              System.err.println("Error writing to the file: " + e.getMessage());
68          }
69      }
```

2. This method extracts team information such as location, season, abbreviation and first year of play from an API linked: https://statsapi.mlb.com/api/v1/teams
and stores it in a list

```java
public List<TeamInfo> extractTeamInfo(String jsonData, List<String> fileData) {
    List<TeamInfo> teamInfoList = new ArrayList<>();

    // Create a JSON parser to parse the JSON data
    JsonParser jsonParser = new JsonParser();
    JsonObject rootObject = jsonParser.parse(jsonData).getAsJsonObject();

    // Extract team data
    for (JsonElement teamElement : rootObject.get("teams").getAsJsonArray()) {
        TeamInfo teamInfo = new TeamInfo();
        JsonObject teamObject = teamElement.getAsJsonObject();

        // Check if the team's name is present in the provided list
        if (fileData.contains(teamObject.get("name").getAsString())) {
            // Extract and set team information
            teamInfo.setName(teamObject.get("name").getAsString());
            teamInfo.setLocationName(teamObject.get("locationName").getAsString());
            teamInfo.setSeason(teamObject.get("season").getAsInt());
            teamInfo.setAbbreviation(teamObject.get("abbreviation").getAsString());
            teamInfo.setFirstYearOfPlay(teamObject.get("firstYearOfPlay").getAsString());

            // Add the TeamInfo object to the list
            teamInfoList.add(teamInfo);

            // Print the extracted TeamInfo object
            System.out.println(teamInfo);
        }
    }

```

3. This method extracts team stadium information from https://geojango.com/pages/list-of-mlb-teams

```java
    public Map<String,String>  extractStadiumInfo() throws IOException {
        Map<String, String> combinedMap = new HashMap<>();
        // Jsoup to fetch HTML data into a document
        Document doc = null;
        try {
            doc = Jsoup.connect(stadiumUrl).get();
        } catch (IOException e) {
            e.printStackTrace();
        }
        if (doc == null) {
            System.out.println("Failed to fetch the HTML document.");
            return Collections.emptyMap();
        }
        Element table = doc.select( cssQuery: "table").first();
        if (table == null) {
            System.out.println("Table element not found in the HTML document.");
            return Collections.emptyMap();
        }
        Elements rows = table.select( cssQuery: "tr");
        // Create a new map to combine the values
        Map<String, String> stadiumMap = new HashMap<>();

        for (int i = 1; i < rows.size(); i++) {
            Element row = rows.get(i);
            Elements columns = row.select( cssQuery: "td");

            // Extract data from the columns
            String teamName = columns.get(0).text();
            String value1 = columns.get(1).text();
            String value2 = columns.get(2).text();
            String value3 = columns.get(3).text();
            String value4 = columns.get(4).text();

            // Combine the values into a single string and put it into the combined map
            String combinedValue = value1 + " | " + value2 + " | " + value3 + " | " + value4;
            stadiumMap.put(teamName, combinedValue);
        }
        return stadiumMap;
    }

    1 usage
    public String fetchStadiumInfo(String teamName) throws IOException {

        Map<String, String> stadiumInfo = extractStadiumInfo();
        String stadiumData = null;

        for (Map.Entry<String,String> entry : stadiumInfo.entrySet()){
            if(entry.getKey().equals(teamName)){
                System.out.println(entry.getKey() +"~~~" + entry.getValue());
                stadiumData = entry.getValue();
                break;
            }
    }
        return  stadiumData;
    }
```

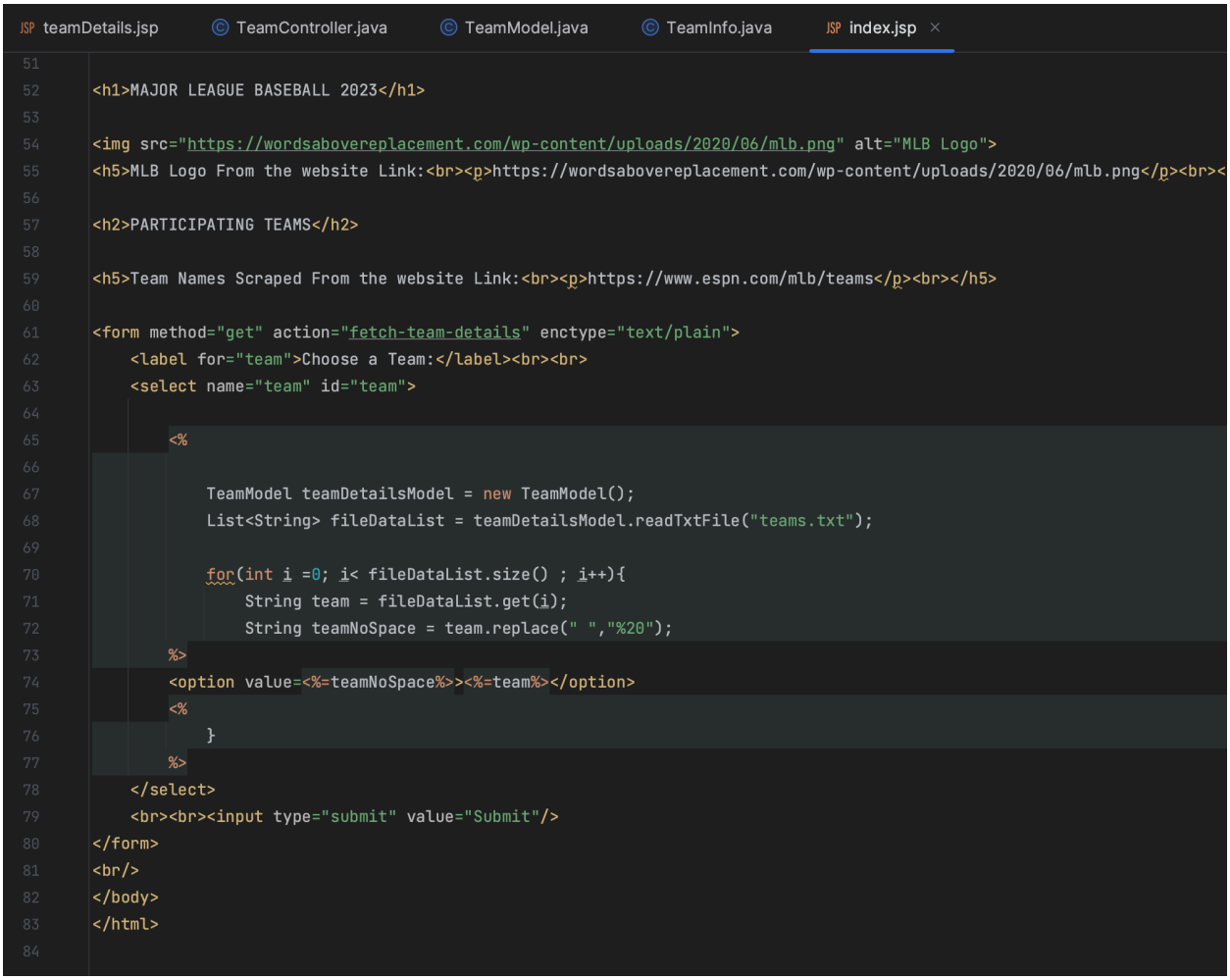
4. This method extracts individual team logo URLS from the website: https://www.sportslogos.net/teams/list_by_year/42023/2023_MLB_Logos/ and stores it in a Map. We retrieve these URLS to display logos on our web page.

```java
public Map<String, String> fetchLogoMap() throws IOException {
    // Jsoup to fetch HTML data into a document
    Document doc = Jsoup.connect(logoUrl).get();

    // Select the HTML element containing the logos
    Element logoWall = doc.select( cssQuery: "ul[class=logoWall]").first();

    if (logoWall == null) {
        System.out.println("Table element not found in the HTML document.");
        return Collections.emptyMap();
    }

    // Create a map to store team name as the key and img src as the value
    Map<String, String> teamLogoMap = new HashMap<>();

    // Select all <a> elements within the logoWall
    Elements anchorElements = logoWall.select( cssQuery: "a");

    for (Element anchorElement : anchorElements) {
        // Extract team name and img src
        String teamName = anchorElement.text();
        String imgSrc = anchorElement.select( cssQuery: "img").attr( attributeKey: "src");

        // Put the data into the map
        teamLogoMap.put(teamName, imgSrc);
    }

    return teamLogoMap;
}
```

5. This method extracts stats for Featured Players within the respective teams from the API linked: https://statsapi.mlb.com/api/v1/stats?stats=season&group=hitting

```
278     public List<TeamPlayerStats> extractTeamPlayerStats(String jsonData) {
279         List<TeamPlayerStats> teamPlayerStatsList = new ArrayList<>();
280         JsonParser jsonParser = new JsonParser();
281         JsonArray jsonArray = jsonParser.parse(jsonData).getAsJsonObject() JsonObject
282                 .get("stats").getAsJsonArray().get(0) JsonElement
283                 .getAsJsonObject().get("splits").getAsJsonArray();
284         for (JsonElement jsonElement : jsonArray) {
285             JsonObject jsonObject = jsonElement.getAsJsonObject();
286             TeamPlayerStats teamPlayerStats = new TeamPlayerStats();
287 
288             // Extract rank
289             teamPlayerStats.setRank(!jsonObject.get("rank").isJsonNull() ? jsonObject.get("rank").getAsInt() : 0);
290 
291             // Extract team[name]
292             JsonObject teamObject = jsonObject.getAsJsonObject( memberName: "team");
293             teamPlayerStats.setTeamName(teamObject.get("name").getAsString());
294 
295             // Extract player[fullName]
296             JsonObject playerObject = jsonObject.getAsJsonObject( memberName: "player");
297             teamPlayerStats.setPlayerName(playerObject.get("fullName").getAsString());
298 
299             // Extract player statistics
300             JsonObject statObject = jsonObject.getAsJsonObject( memberName: "stat");
301             teamPlayerStats.setGamesPlayed(statObject.get("gamesPlayed").getAsInt());
302             teamPlayerStats.setGroundOuts(statObject.get("groundOuts").getAsInt());
303             teamPlayerStats.setAirOuts(statObject.get("airOuts").getAsInt());
304             teamPlayerStats.setRuns(statObject.get("runs").getAsInt());
305             teamPlayerStats.setDoubles(statObject.get("doubles").getAsInt());
306             teamPlayerStats.setTriples(statObject.get("triples").getAsInt());
307             teamPlayerStats.setHomeRuns(statObject.get("homeRuns").getAsInt());
308             teamPlayerStats.setStrikeOuts(statObject.get("strikeOuts").getAsInt());
309             teamPlayerStats.setAverage(statObject.get("avg").getAsString());
310 
311             teamPlayerStatsList.add(teamPlayerStats);
312         }
313         return teamPlayerStatsList;
314     }
```

6. Home Page jsp Output




```jsp
<h1>MAJOR LEAGUE BASEBALL 2023</h1>

<img src="https://wordsabovereplacement.com/wp-content/uploads/2020/06/mlb.png" alt="MLB Logo">
<h5>MLB Logo From the website Link:<br><p>https://wordsabovereplacement.com/wp-content/uploads/2020/06/mlb.png</p><br><

<h2>PARTICIPATING TEAMS</h2>

<h5>Team Names Scraped From the website Link:<br><p>https://www.espn.com/mlb/teams</p><br></h5>

<form method="get" action="fetch-team-details" enctype="text/plain">
    <label for="team">Choose a Team:</label><br><br>
    <select name="team" id="team">

        <%

            TeamModel teamDetailsModel = new TeamModel();
            List<String> fileDataList = teamDetailsModel.readTxtFile("teams.txt");

            for(int i =0; i< fileDataList.size() ; i++){
                String team = fileDataList.get(i);
                String teamNoSpace = team.replace(" ","%20");
        %>
        <option value=<%=teamNoSpace%>><%=team%></option>
        <%
            }
        %>
    </select>
    <br><br><input type="submit" value="Submit"/>
</form>
<br/>
</body>
</html>
```

7. Result Page jsp Output

```
<body>
<div class="header-container">
    <h1>MAJOR LEAGUE BASEBALL 2023</h1>
    <img class="mlb-logo" src="https://wordsabovereplacement.com/wp-content/uploads/2020/06/mlb.png" alt="MLB Logo">
    <h5>MLB Logo From the website Link:<br><p>https://wordsabovereplacement.com/wp-content/uploads/2020/06/mlb.png</p><br></h5>

</div>

<h1>${teamName}</h1>

<h2><br><img src =${teamlogoUrl} width="200" height="125"/><br></h2>
<h5>Team Logo Scraped From the website Link:<br><p>https://www.sportslogos.net/teams/list_by_year/42023/2023_MLB_Logos/</p><br></h5>


<h2>About the Team</h2>
<h5>Team Information Scraped From the API Link:<br><p>https://statsapi.mlb.com/api/v1/teams</p><br></h5>

<h3>HeadQuarters:  ${locationName}</h3>
<h3>Season:  ${season}</h3>
<h3>Abbreviation:  ${abbreviation}</h3>
<h3>First Appeared In:  ${firstYearOfPlay}</h3>

<br><br>

<h2>The Team's Home Stadium</h2>
<h5>Home Stadium Information Scraped From the website Link:<br><p>https://geojango.com/pages/list-of-mlb-teams</p><br></h5>

<h3>${arenaName}</h3>

<br><br>

html › body › h3
```

8. Result Page Output jsp Continued

```jsp
<br><br>

<h2>Featured Players</h2>
<h5>Player Information Scraped From the API Link:<br><p>https://statsapi.mlb.com/api/v1/stats?stats=season&group=hitting</p><br></h5>

    <c:forEach items="${playerDetails}" var="player">

            <h2>Player Name: <c:out value="${player.playerName}"/></h2>
            <h3>Rank: <c:out value="${player.rank}"/></h3>
            <h3>Games Played: <c:out value="${player.gamesPlayed}"/></h3>
            <h3>Ground Outs: <c:out value="${player.groundOuts}"/></h3>
            <h3>Air Outs: <c:out value="${player.airOuts}"/></h3>
            <h3>Runs: <c:out value="${player.runs}"/></h3>
            <h3>Doubles: <c:out value="${player.doubles}"/></h3>
            <h3>Triples: <c:out value="${player.triples}"/></h3>
            <h3>Home Runs:  <c:out value="${player.homeRuns}"/></h3>
            <h3>Strike Outs: <c:out value="${player.strikeOuts}"/></h3>
            <br><br>
    </c:forEach>


<br><br>
<div class="button-container">
    <a href="index.jsp" class="button">&laquo; Back</a>
</div>

</body>
</html>
```

9. This code invokes the Servlet Method

```
61 @    public void doGet(HttpServletRequest request, HttpServletResponse response) throws IOException, ServletException {
62
63          request.setAttribute( s: "teamNameScrapperUrl",teamNameScrapperUrl);
64          // select path for file which will store the scraped mlb team names
65
66
67          // Fetch JSON data from the API
68          response.setContentType("text/html");
69          String teamName = request.getParameter( s: "team").replace( target: "%20", replacement: " ");
70
71
72          List<TeamPlayerStats> playerDetails = teamDetailsModel.fetchPlayersFromStadium(teamName);
73          System.out.println(playerDetails.size());
74          TeamInfo teamInfo = teamDetailsModel.fetchTeamInfo(teamName);
75
76          String stadiumInfo = teamDetailsModel.fetchStadiumInfo(teamName);
77          System.out.println("qwerthjkhgfds" + stadiumInfo);
78
79          request.setAttribute( s: "teamName", teamName);
80          request.setAttribute( s: "teamlogoUrl",teamDetailsView.getTeamLogoMap().get(teamName));
81          request.setAttribute( s: "playerDetails",playerDetails);
82
83
84          request.setAttribute( s: "locationName", teamInfo.getLocationName());
85          request.setAttribute( s: "season", teamInfo.getSeason());
86          request.setAttribute( s: "abbreviation", teamInfo.getAbbreviation());
87          request.setAttribute( s: "firstYearOfPlay", teamInfo.getFirstYearOfPlay());
88
89          request.setAttribute( s: "arenaName",stadiumInfo);
90
91
92          // Transfer control over the the correct "view"
93      RequestDispatcher view = request.getRequestDispatcher( s: "teamDetails.jsp");
94      view.forward(request, response);
95
96      }
```