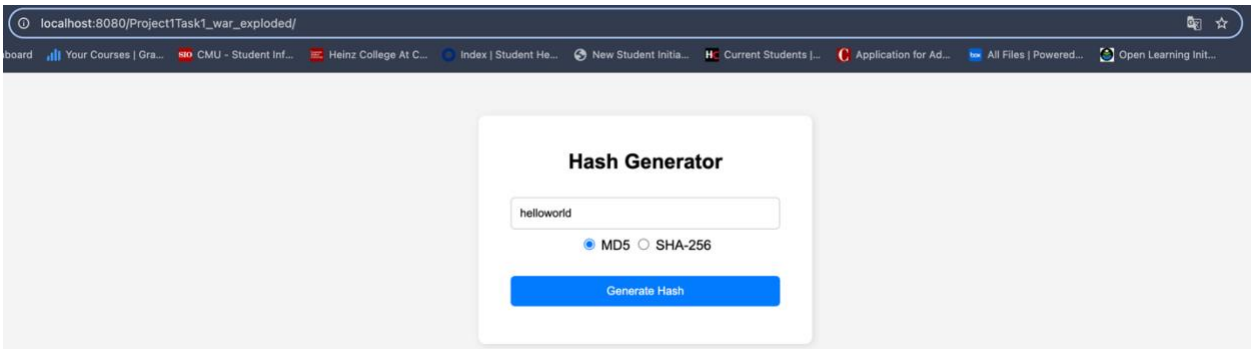
Project 1 - Task 1

Input – Hash Generator



Output – Hash Generator

- **Hash Computation Output for MD5 in Hex and Base 64**

Hash Computation Result

Original Text: helloworld

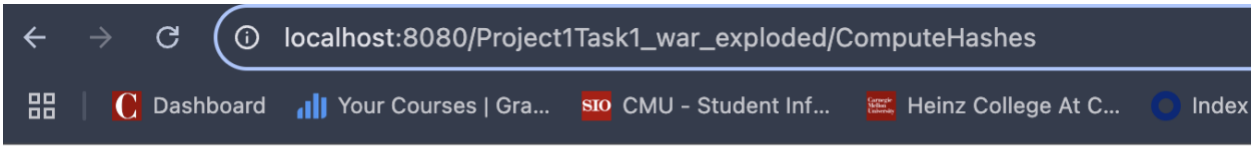
Hash Type: MD5

Hexadecimal: FC5E038D38A57032085441E7FE7010B0

Base64: /F4DjTilcDIIVEHn/nAQsA==

Back

- **Hash Computation Output for SHA-256 in Hex and Base 64**



Hash Computation Result

Original Text: helloworld

Hash Type: SHA-256

Hexadecimal: 936A185CAAA266BB9CBE981E9E05CB78CD732B0B3280EB944412BB6F8F8F07AF

Base64: k2oYXKqiZrucvpgengXLeM1zKwsygOuURBK7b4+PB68=

Back

- **Code Snippet of computation of each Hash**

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
    // Get user input / Hash type selection
    String inputText = request.getParameter("inputText");
    String hashType = request.getParameter("hashType");

    // Validate input
    if (inputText == null || inputText.trim().isEmpty()) {
        response.sendError(HttpServletResponse.SC_BAD_REQUEST, "Input text cannot be
empty.");
        return;
    }

    // Compute into hash
    String hexHash = "";
    String base64Hash = "";

    try {
        MessageDigest digest = MessageDigest.getInstance(hashType);
        byte[] hashBytes = digest.digest(inputText.getBytes());

        // Convert hash to Hex & Base64 formats
        hexHash = DatatypeConverter.printHexBinary(hashBytes);
        base64Hash = DatatypeConverter.printBase64Binary(hashBytes);
```

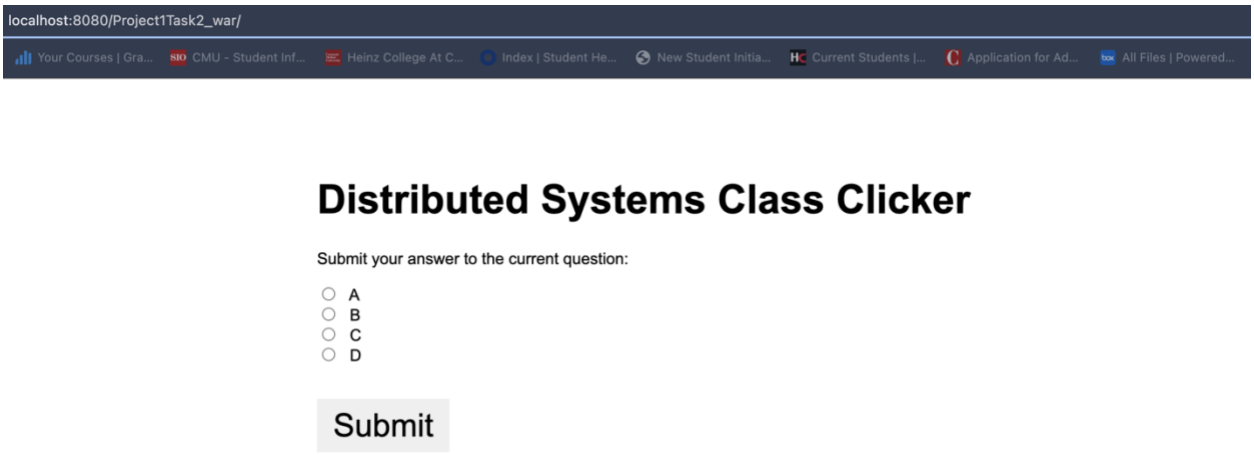
```
    } catch (NoSuchAlgorithmException e) {
        response.sendError(HttpServletResponse.SC_INTERNAL_SERVER_ERROR, "Error
computing hash.");
        return;
    }
```

```java
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    // Get user input / Hash type selection
    String inputText = request.getParameter( s: "inputText");
    String hashType = request.getParameter( s: "hashType");

    // Validate input
    if (inputText == null || inputText.trim().isEmpty()) {
        response.sendError(HttpServletResponse.SC_BAD_REQUEST, s: "Input text cannot be empty.");
        return;
    }

    // Compute into hash
    String hexHash = "";
    String base64Hash = "";

    try {
        MessageDigest digest = MessageDigest.getInstance(hashType);
        byte[] hashBytes = digest.digest(inputText.getBytes());

        // Convert hash to Hex & Base64 formats
        hexHash = DatatypeConverter.printHexBinary(hashBytes);
        base64Hash = DatatypeConverter.printBase64Binary(hashBytes);
```

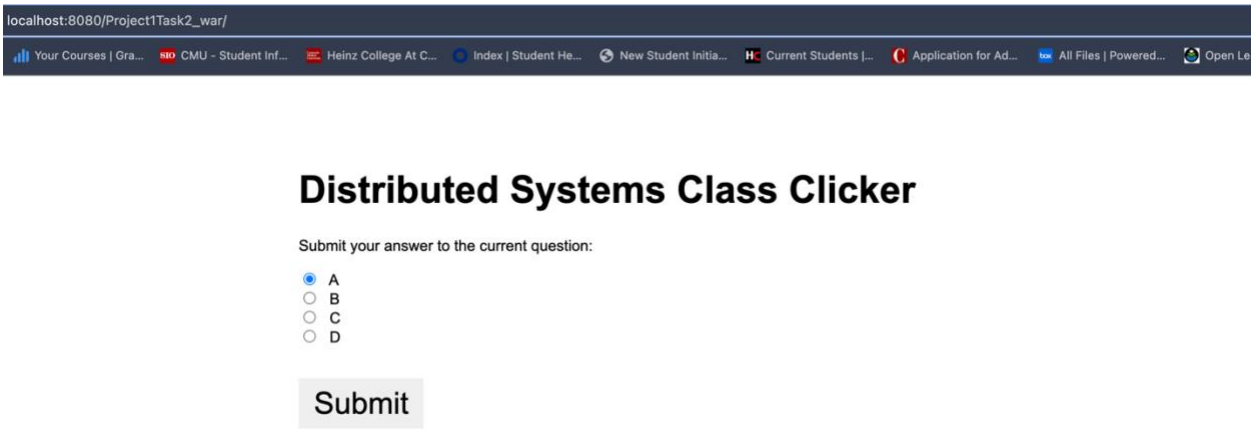
Project 1 - Task 2

Input Pages

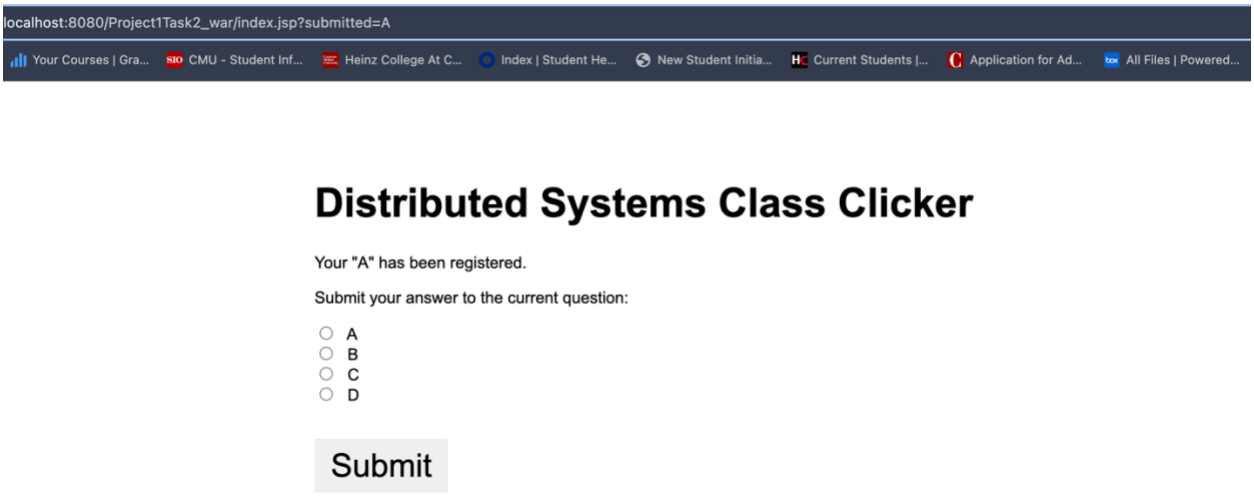
- **Home of Distributed Systems Class Clicker**



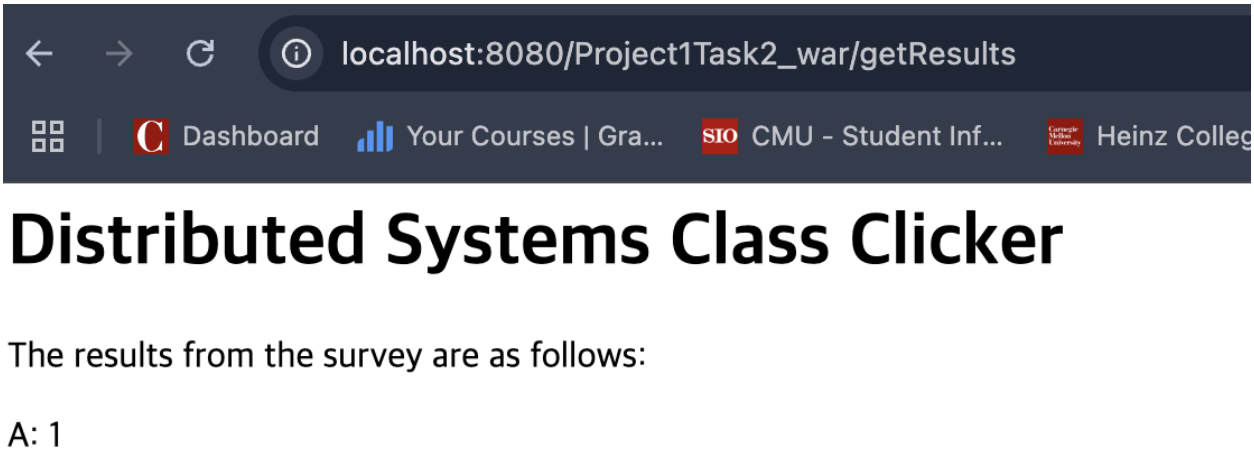
- **Selecting the radio buttons (EX. A) on the picture**



- **“A” is registered into the storage**

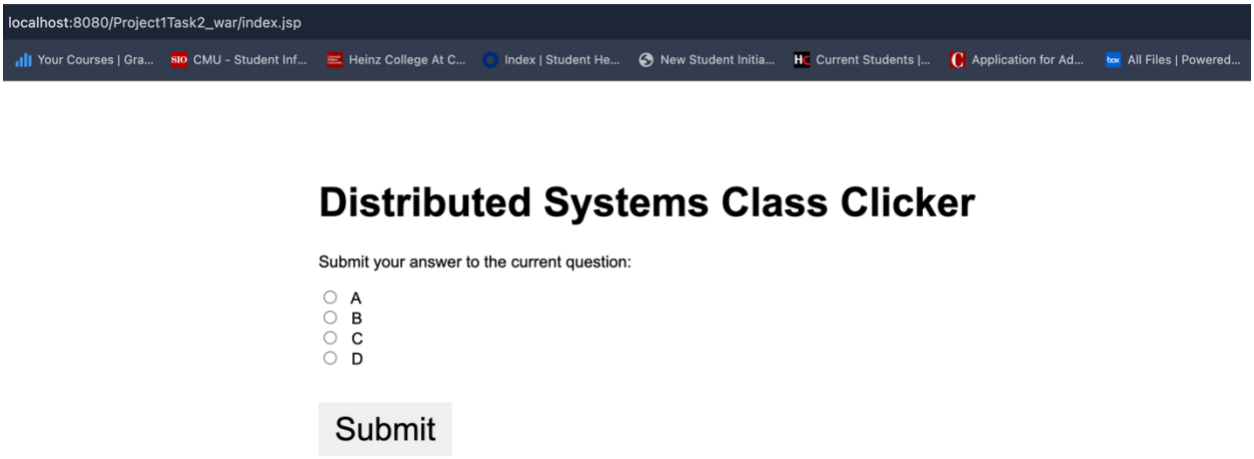


- **Output on the “Localhost URL /getResults”**

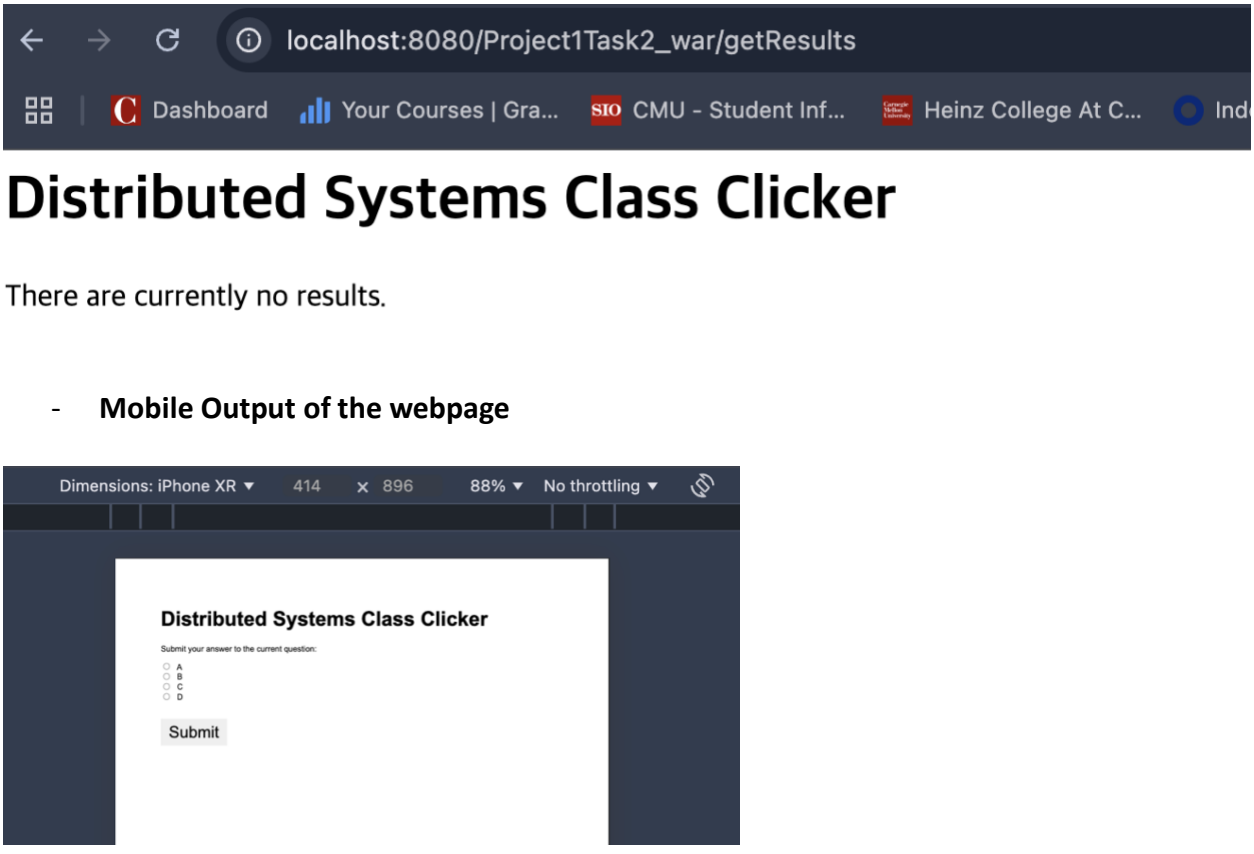


- **Changing the URL with “Local Host URL /submit”**

- **Redirecting to Home (DSM Class Clicker)**



- **Output on the “localhost URL /getResults” without choosing any options.**



- **Mobile Output of the webpage**

- **Code Snippet for producing clicker output.**

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    String path = request.getServletPath();

    if ("/submit".equals(path)) {
        // Redirect to index.jsp (front page)
        response.sendRedirect( location: "index.jsp");
        return;
    }

    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.println("<html><body>");
    out.println("<h1>Distributed Systems Class Clicker</h1>");

    if (results.isEmpty()) {
        out.println("<p>There are currently no results.</p>");
    } else {
        out.println("<p>The results from the survey are as follows:</p>");

        // Sort the results in alphabetical order. The unselected one doesn't display.
        results.entrySet().stream()
                .sorted(Map.Entry.comparingByKey())
                .forEach(entry -> out.println("<p>" + entry.getKey() + ": " + entry.getValue() + "</p>"));

        // Clear the Results out
        results.clear();
```

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    String path = request.getServletPath();

    if ("/submit".equals(path)) {
        // Redirect to index.jsp (front page)
        response.sendRedirect("index.jsp");
        return;
    }

    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.println("<html><body>");
    out.println("<h1>Distributed Systems Class Clicker</h1>");

    if (results.isEmpty()) {
        out.println(" There are currently no results. ");
    } else {
        out.println(" The results from the survey are as follows: ");

        // Sort the results in alphabetical order. The unselected one doesn't display.

```
        results.entrySet().stream()
                .sorted(Map.Entry.comparingByKey())
                .forEach(entry -> out.println("<p>" + entry.getKey() + ": " + entry.getValue() +
"</p>"));

        // Clear the Results out
        results.clear();
    }
```

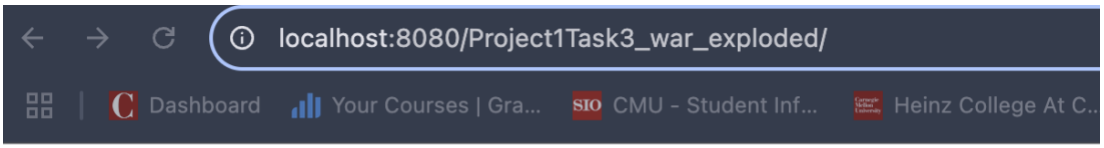
- **Code Snippet for processing user submission for the output (optional)**

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    String answer = request.getParameter( s: "answer");
    if (answer != null && !answer.isEmpty()) {
        results.put(answer, results.getOrDefault(answer, defaultValue: 0) + 1);
    }

    // Redirect to index.jsp with chosen alphabets
    response.sendRedirect( location: "index.jsp?submitted=" + answer);
}
```

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
    String answer = request.getParameter("answer");
    if (answer != null && !answer.isEmpty()) {
        results.put(answer, results.getOrDefault(answer, 0) + 1);
    }

    // Redirect to index.jsp with chosen alphabets
    response.sendRedirect("index.jsp?submitted=" + answer);
  }
```

Project 1 – Task 3

- **Input page with Great Smoky Mountains NP**



U.S. National Parks

Created by Hojoon Lee

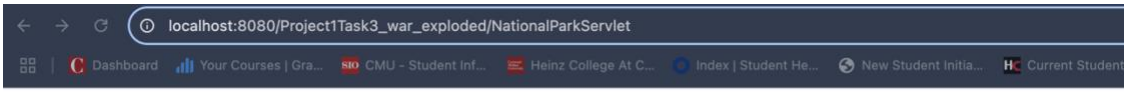
Parks:

Choose a Park:

Great Smoky Mountains NP

Submit

- **Output pages for Great Smoky Mountains NP**



Great Smoky Mountains National Park

Credit: www.nps.gov

Current conditions:

Temperature: 43°F

Humidity: 100%

Wind speed: 6 mph

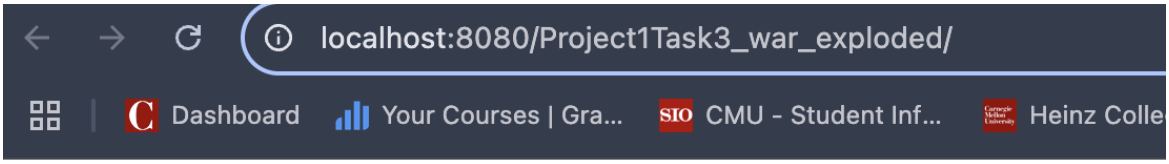
Credit: forecast.weather.gov (National Weather Service)

Great Smoky Mountains National Park Activities:

- Arts and Culture
- Live Music
- Auto and ATV
- Scenic Driving
- Astronomy
- Stargazing
- Biking
- Road Biking
- Camping
- Backcountry Camping
- Car or Front Country Camping
- Horse Camping (see also Horse/Stock Use)
- Group Camping
- RV Camping
- Fishing
- Freshwater Fishing
- Fly Fishing
- Food
- Picnicking
- Guided Tours
- Self-Guided Tours - Auto
- Hands-On
- Citizen Science
- Hiking
- Backcountry Hiking
- Front-Country Hiking
- Horse Trekking
- Horse Camping (see also camping)
- Horseback Riding
- Junior Ranger Program
- Wildlife Watching
- Birdwatching
- Park Film
- Museum Exhibits
- Shopping
- Bookstore and Park Store

Credit: https://developer.nps.gov

- **Input page with Mammoth Cave NP**



U.S. National Parks

Created by Hojoon Lee

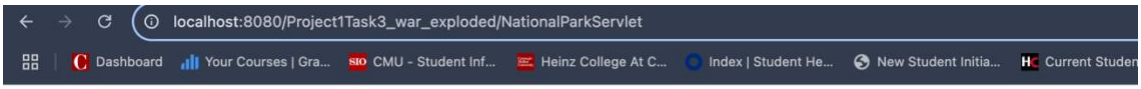
Parks:

Choose a Park:

Mammoth Cave NP

Submit

- **Output page for Mammoth Cave NP**



Mammoth Cave National Park

Credit: www.nps.gov

Current conditions:

Temperature: 37°F

Humidity: 74%

Wind speed: 6 mph

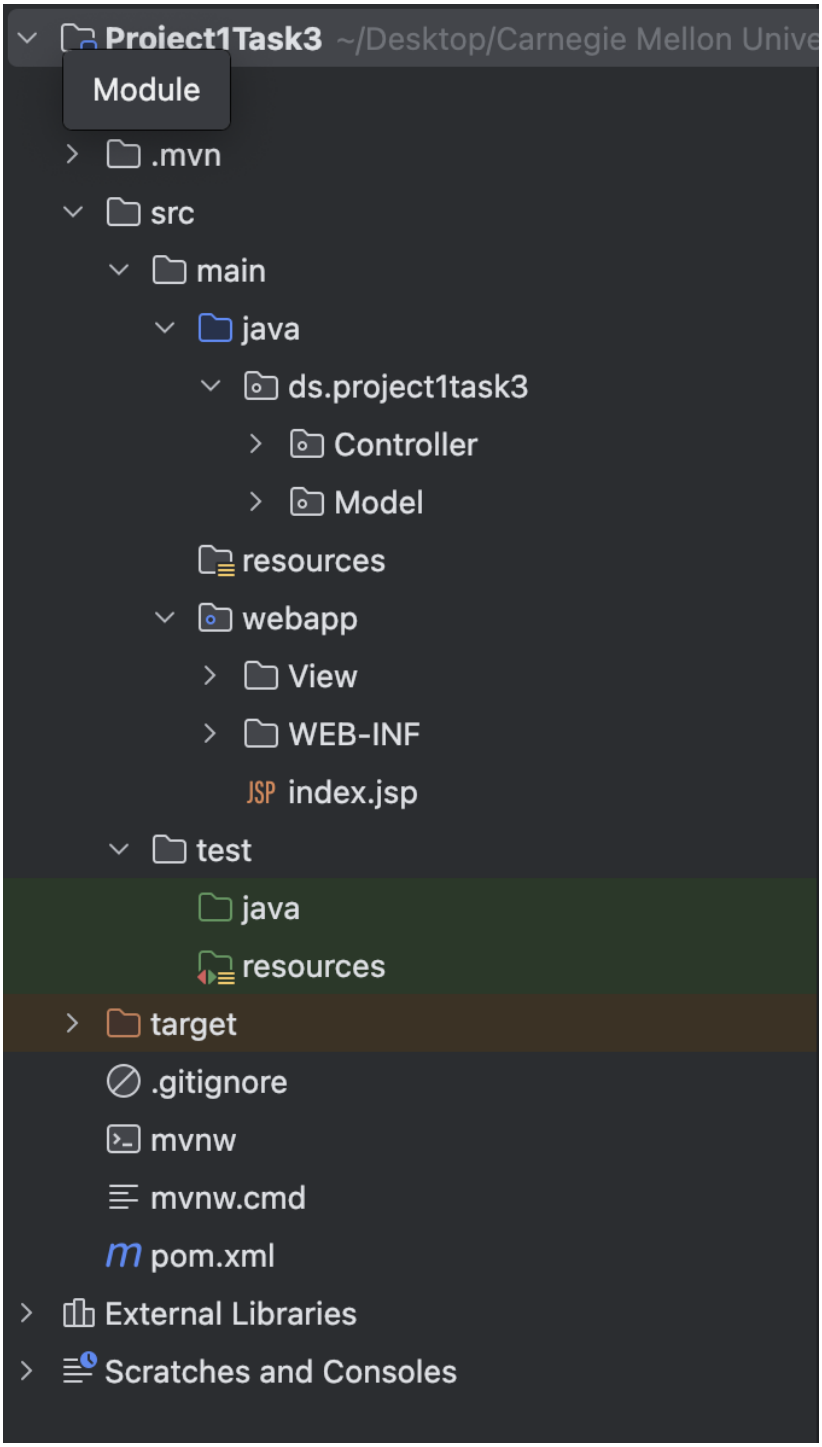
Credit: forecast.weather.gov (National Weather Service)

Mammoth Cave National Park Activities:

- Astronomy
- Stargazing
- Biking
- Boating
- Camping
- Backcountry Camping
- Canoe or Kayak Camping
- Car or Front Country Camping
- Horse Camping (see also Horse/Stock Use)
- Group Camping
- RV Camping
- Caving
- Fishing
- Freshwater Fishing
- Food
- Dining
- Picnicking
- Guided Tours
- Self-Guided Tours - Walking
- Hands-On
- Hiking
- Backcountry Hiking
- Front-Country Hiking
- Horse Trekking
- Horse Camping (see also camping)
- Horseback Riding
- Paddling
- Canoeing
- Canoe or Kayak Camping
- Kayaking
- Junior Ranger Program
- Wildlife Watching
- Park Film
- Shopping
- Bookstore and Park Store

Credit: https://developer.nps.gov

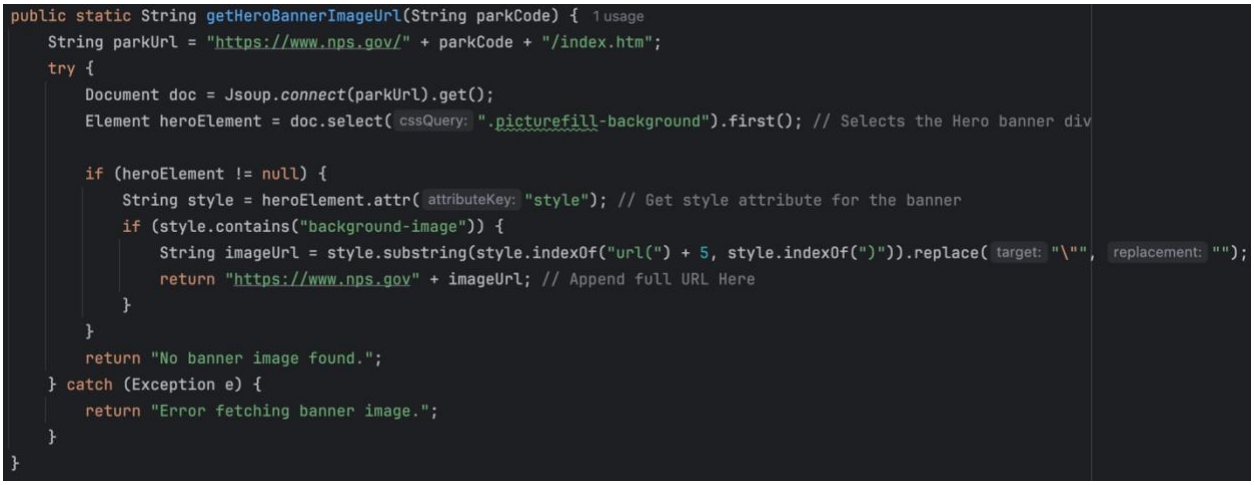
MVC for the Code



- **Controller Folder**
- **Model Folder**
- **View Folder**

Code Snippets for Screen Scarping

For screen scraping, the target was not a direct image URL but rather the background banner image applied via CSS. Initially, I attempted to extract the image using standard <img> tag selection, but this consistently retrieved the first image on the website instead of the desired background banner. To resolve this, I used an LLM (GPT) to refine the Jsoup scraping logic, specifically for extracting the CSS background image URL. Additionally, I used the browser's Developer Tools to inspect the page structure, identify the correct HTML element referring to the banner, and implement it in my screen scraping method.



```
public static String getHeroBannerImageUrl(String parkCode) {
    String parkUrl = "https://www.nps.gov/" + parkCode + "/index.htm";
    try {
        Document doc = Jsoup.connect(parkUrl).get();
        Element heroElement = doc.select(".picturefill-background").first(); // Selects the Hero banner div

        if (heroElement != null) {
            String style = heroElement.attr("style"); // Get style attribute for the banner
            if (style.contains("background-image")) {
                String imageUrl = style.substring(style.indexOf("url(") + 5, style.indexOf(")")).replace("\"", "");
                return "https://www.nps.gov" + imageUrl; // Append full URL Here
            }
        }
        return "No banner image found.";
    } catch (Exception e) {
        return "Error fetching banner image.";
    }
}
```

```
  public static String getHeroBannerImageUrl(String parkCode) {
    String parkUrl = "https://www.nps.gov/" + parkCode + "/index.htm";
    try {
      Document doc = Jsoup.connect(parkUrl).get();
      Element heroElement = doc.select(".picturefill-background").first(); // Selects the Hero
banner div

      if (heroElement != null) {
        String style = heroElement.attr("style"); // Get style attribute for the banner
        if (style.contains("background-image")) {
          String imageUrl = style.substring(style.indexOf("url(") + 5,
style.indexOf(")")).replace("\"", "");
          return "https://www.nps.gov" + imageUrl; // Append full URL Here
        }
      }
      return "No banner image found.";
    } catch (Exception e) {
      return "Error fetching banner image.";
    }
  }
```

Code Snippets for queries the API

- Name of the park

```
public static String getFullParkName(String parkCode) {  1 usage
    String apiUrl = "https://developer.nps.gov/api/v1/parks?parkCode=" + parkCode + "&api_key=" + API_KEY;
    try {
        URL url = new URL(apiUrl);
        HttpURLConnection conn = (HttpURLConnection) url.openConnection();
        conn.setRequestMethod("GET");
        conn.setRequestProperty("User-Agent", "Mozilla/5.0");

        BufferedReader reader = new BufferedReader(new InputStreamReader(conn.getInputStream()));
        StringBuilder json = new StringBuilder();
        String line;
        while ((line = reader.readLine()) != null) {
            json.append(line);
        }
        reader.close();
```

```
public static String getFullParkName(String parkCode) {
      String apiUrl = "https://developer.nps.gov/api/v1/parks?parkCode=" + parkCode +
"&api_key=" + API_KEY;
      try {
         URL url = new URL(apiUrl);
         HttpURLConnection conn = (HttpURLConnection) url.openConnection();
         conn.setRequestMethod("GET");
         conn.setRequestProperty("User-Agent", "Mozilla/5.0");

         BufferedReader reader = new BufferedReader(new
InputStreamReader(conn.getInputStream()));
         StringBuilder json = new StringBuilder();
         String line;
         while ((line = reader.readLine()) != null) {
            json.append(line);
         }
         reader.close();

         JsonObject jsonObject = JsonParser.parseString(json.toString()).getAsJsonObject();
         JsonArray parksArray = jsonObject.getAsJsonArray("data");
```

- Weather

```
public static String getWeather(double latitude, double longitude) {  1 usage
    String apiUrl = "https://forecast.weather.gov/MapClick.php?lat=" + latitude + "&lon=" + longitude + "&FcstType=json";
    try {
        URL url = new URL(apiUrl);
        HttpURLConnection conn = (HttpURLConnection) url.openConnection();
        conn.setRequestMethod("GET");
        conn.setRequestProperty("User-Agent", "Mozilla/5.0");

        BufferedReader reader = new BufferedReader(new InputStreamReader(conn.getInputStream()));
        StringBuilder json = new StringBuilder();
        String line;
        while ((line = reader.readLine()) != null) {
            json.append(line);
        }
        reader.close();

        JsonObject jsonObject = JsonParser.parseString(json.toString()).getAsJsonObject();
        JsonObject observation = jsonObject.getAsJsonObject( memberName: "currentobservation");

        if (observation != null) {
            String temp = observation.get("Temp").getAsString();
            String humidity = observation.get("Relh").getAsString();
            String windSpeed = observation.get("Winds").getAsString();
            return "Temp: " + temp + "°F, Humidity: " + humidity + "%, Wind: " + windSpeed + " mph";
        }
```

```
public static String getWeather(double latitude, double longitude) {
    String apiUrl = "https://forecast.weather.gov/MapClick.php?lat=" + latitude + "&lon=" +
longitude + "&FcstType=json";
    try {
        URL url = new URL(apiUrl);
        HttpURLConnection conn = (HttpURLConnection) url.openConnection();
        conn.setRequestMethod("GET");
        conn.setRequestProperty("User-Agent", "Mozilla/5.0");

        BufferedReader reader = new BufferedReader(new
InputStreamReader(conn.getInputStream()));
        StringBuilder json = new StringBuilder();
        String line;
        while ((line = reader.readLine()) != null) {
            json.append(line);
        }
        reader.close();

        JsonObject jsonObject = JsonParser.parseString(json.toString()).getAsJsonObject();
        JsonObject observation = jsonObject.getAsJsonObject("currentobservation");
```

- Activities

```
public static String getActivities(String parkCode) {  1 usage
    String apiUrl = "https://developer.nps.gov/api/v1/parks?parkCode=" + parkCode + "&api_key=" + API_KEY;
    try {
        URL url = new URL(apiUrl);
        HttpURLConnection conn = (HttpURLConnection) url.openConnection();
        conn.setRequestMethod("GET");
        conn.setRequestProperty("User-Agent", "Mozilla/5.0");

        BufferedReader reader = new BufferedReader(new InputStreamReader(conn.getInputStream()));
        StringBuilder json = new StringBuilder();
        String line;
        while ((line = reader.readLine()) != null) {
            json.append(line);
        }
        reader.close();

        JsonObject jsonObject = JsonParser.parseString(json.toString()).getAsJsonObject();
        JsonArray parksArray = jsonObject.getAsJsonArray( memberName: "data");
```

```
public static String getActivities(String parkCode) {
    String apiUrl = "https://developer.nps.gov/api/v1/parks?parkCode=" + parkCode +
"&api_key=" + API_KEY;
    try {
        URL url = new URL(apiUrl);
        HttpURLConnection conn = (HttpURLConnection) url.openConnection();
        conn.setRequestMethod("GET");
        conn.setRequestProperty("User-Agent", "Mozilla/5.0");

        BufferedReader reader = new BufferedReader(new
InputStreamReader(conn.getInputStream()));
        StringBuilder json = new StringBuilder();
        String line;
        while ((line = reader.readLine()) != null) {
            json.append(line);
        }
        reader.close();

        JsonObject jsonObject = JsonParser.parseString(json.toString()).getAsJsonObject();
        JsonArray parksArray = jsonObject.getAsJsonArray("data");
```

Code Snippets for producing output (From Result.jsp)

- **Header / The name of National Park**

```
<%
    NationalPark park = (NationalPark) request.getAttribute("park");
%>
<!DOCTYPE html>
<html>
<head><title><%= park.getName() %></title></head>

<h1><%= park.getName() %></h1>
<img src="<%= park.getImageUrl() %>" alt="Park Image">
<p>Credit: www.nps.gov</p>
<style>
    body {
        font-family: Arial, sans-serif;}
    h1 {
        font-size: 28px;
        font-weight: bold;}
    h2 {
        font-size: 22px;
        font-weight: bold;}
    p {
        font-size: 18px;}
    .weather, .activities {
        margin-bottom: 20px;}
</style>
```

- **Weather / Activities**

```
<!-- Weather Conditions-->
<div class="weather">
    <h2>Current conditions:</h2>
    <p><strong>Temperature:</strong> <%= park.getWeather().split(",")[0].replace("Temp:", "").trim() %></p>
    <p><strong>Humidity:</strong> <%= park.getWeather().split(",")[1].replace("Humidity:", "").trim() %></p>
    <p><strong>Wind speed:</strong> <%= park.getWeather().split(",")[2].replace("Wind:", "").trim() %></p>
    <p>Credit: forecast.weather.gov (National Weather Service)</p>
</div>

<!-- Activities for the park -->
<div class="activities">
    <h2><%= park.getName() %> Activities:</h2>
    <ul>
        <%
            String[] activitiesList = park.getActivities().split(",");
            for (String activity : activitiesList) {
        %>
        <li><%= activity.trim() %></li>
        <%
            }
        %>
    </ul>
    <p>Credit: https://developer.nps.gov</p>
</div>
```

```
<%
   NationalPark park = (NationalPark) request.getAttribute("park");
%>
<!DOCTYPE html>
<html>
<head><title><%= park.getName() %></title></head>

<h1><%= park.getName() %></h1>
<img src="<%= park.getImageUrl() %>" alt="Park Image">
<p>Credit: www.nps.gov</p>
<style>
   body {
      font-family: Arial, sans-serif;}
   h1 {
      font-size: 28px;
      font-weight: bold;}
   h2 {
      font-size: 22px;
      font-weight: bold;}
   p {
      font-size: 18px;}
   .weather, .activities {
```

```
      margin-bottom: 20px;}
</style>

<body>
<!-- Weather Conditions-->
<div class="weather">
   <h2>Current conditions:</h2>
   <p><strong>Temperature:</strong> <%= park.getWeather().split(",")[0].replace("Temp:",
"").trim() %></p>
   <p><strong>Humidity:</strong> <%= park.getWeather().split(",")[1].replace("Humidity:",
"").trim() %></p>
   <p><strong>Wind speed:</strong> <%= park.getWeather().split(",")[2].replace("Wind:",
"").trim() %></p>
   <p>Credit: forecast.weather.gov (National Weather Service)</p>
</div>

<!-- Activities for the park -->
<div class="activities">
   <h2><%= park.getName() %> Activities:</h2>
   <ul>
      <%
         String[] activitiesList = park.getActivities().split(",");
         for (String activity : activitiesList) {
      %>
      <li><%= activity.trim() %></li>
      <%
         }
      %>
   </ul>
   <p>Credit: https://developer.nps.gov</p>
</div>
```