Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

Task 1:

1. **Screen shots of input, MD5 and SHA-256 output, both in hex and base 64**

Web input:

MD5 Output:

Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

SHA-256 Output:

2. Code snippets of computation of each hash

doPost() in file computeHashes:

```java
@WebServlet("/ComputeHashes")
public class ComputeHashes extends HttpServlet {
    @Override  no usages
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        //Received the text from user
        String text = request.getParameter( s: "text");
        //Received the type of compute from users
        String hashComputeRequest = request.getParameter( s: "hashComputeRequest");

        String compute_in_hex = "";
        String compute_in_base64 = "";

        try {
            //Use MessageDigest.getInstance to digest the compute request
            MessageDigest digest_request = MessageDigest.getInstance(hashComputeRequest);
            byte[] request_in_byte =  digest_request.digest(text.getBytes());
            //Compute the input text and compute
            //Code snippets of computation of each hash
            compute_in_hex = DatatypeConverter.printHexBinary(request_in_byte);
            compute_in_base64 = DatatypeConverter.printBase64Binary(request_in_byte);
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        }
```

Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

```java
        try {
            //Use MessageDigest.getInstance to digest the compute request
            MessageDigest digest_request = MessageDigest.getInstance(hashComputeRequest);
            byte[] request_in_byte =  digest_request.digest(text.getBytes());
            //Compute the input text and compute
            //Code snippets of computation of each hash
            compute_in_hex = DatatypeConverter.printHexBinary(request_in_byte);
            compute_in_base64 = DatatypeConverter.printBase64Binary(request_in_byte);
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        }

        // Show out the details on website
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        out.println("<html><body>");
        out.println("<h1>Generation Results for Hash Computation</h1>");
        out.println("<p>Typed Text: " + text + "</p>");
        out.println("<p>Hash Compute Request Name: " + hashComputeRequest + "</p>");
        out.println("<p>Hexadecimal Hash: " + compute_in_hex + "</p>");
        out.println("<p>Base64 Hash: " + compute_in_base64 + "</p>");
        out.println("</body></html>");
    }

}
```

Code snippets of computation of each hash

doPost() in hashCompute.java

```
protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    //Received the text from user
    String text = request.getParameter("text");
    //Received the type of compute from users
    String hashComputeRequest = request.getParameter("hashComputeRequest");

    String compute_in_hex = "";
    String compute_in_base64 = "";

    try {
        //Use MessageDigest.getInstance to digest the compute request
        MessageDigest digest_request =
MessageDigest.getInstance(hashComputeRequest);
        byte[] request_in_byte =    digest_request.digest(text.getBytes());
        //Compute the input text and compute
```

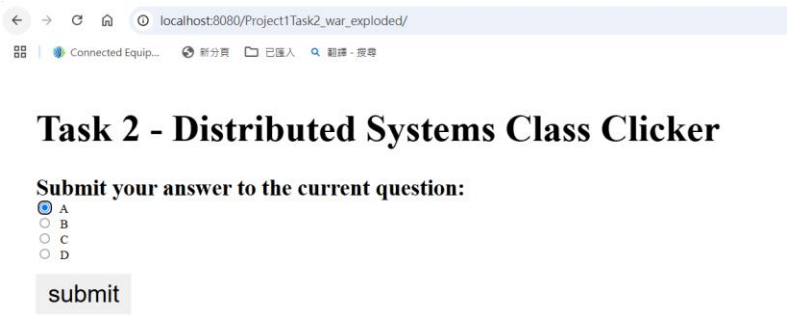
```
            //Code snippets of computation of each hash
            compute_in_hex = DatatypeConverter.printHexBinary(request_in_byte);
            compute_in_base64 =
DatatypeConverter.printBase64Binary(request_in_byte);
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        }

        // Show out the details on website
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        out.println("<html><body>");
        out.println("<h1>Generation Results for Hash Computation</h1>");
        out.println("<p>Typed Text: " + text + "</p>");
        out.println("<p>Hash Compute Request Name: " + hashComputeRequest +
"</p>");
        out.println("<p>Hexadecimal Hash: " + compute_in_hex + "</p>");
        out.println("<p>Base64 Hash: " + compute_in_base64 + "</p>");
        out.println("</body></html>");
}
```

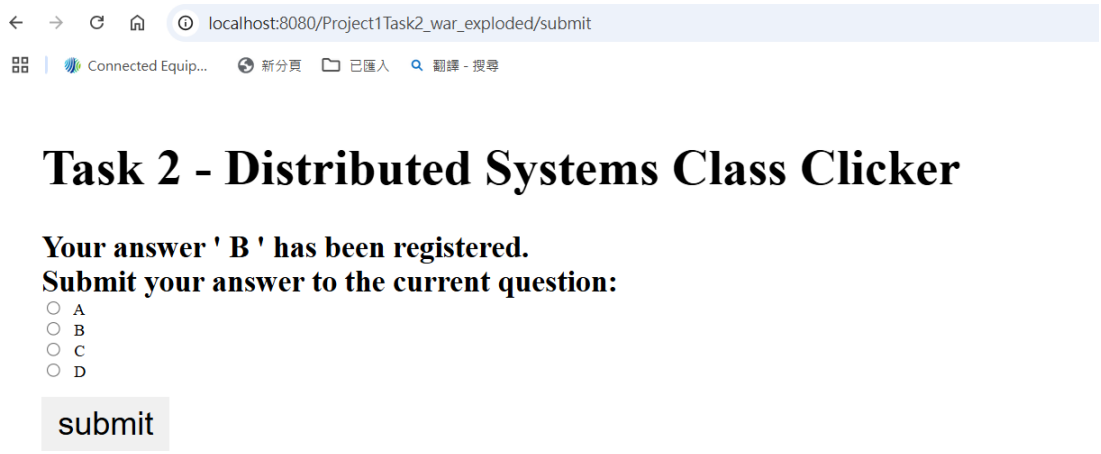
Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

Task 2:

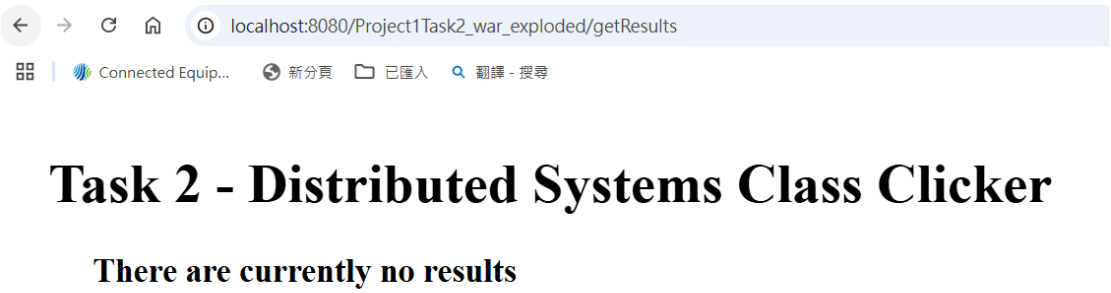
1. Screen shots of input page(s) and output page(s).

Input page in desktop:

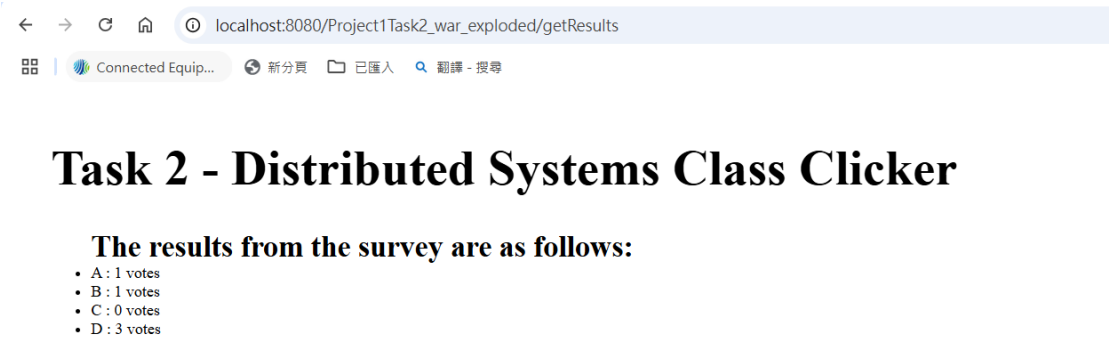




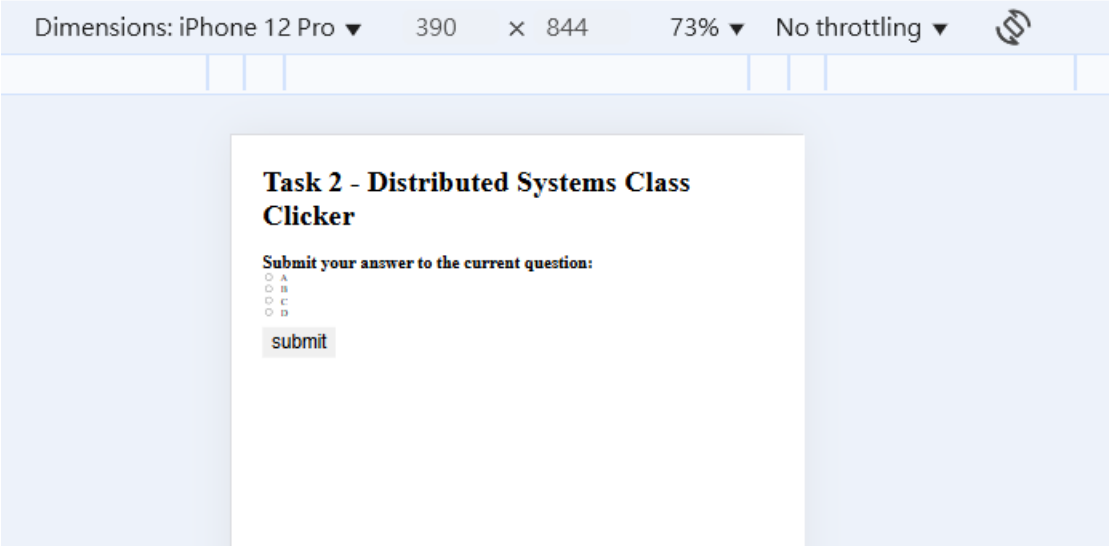
Onput page in desktop:



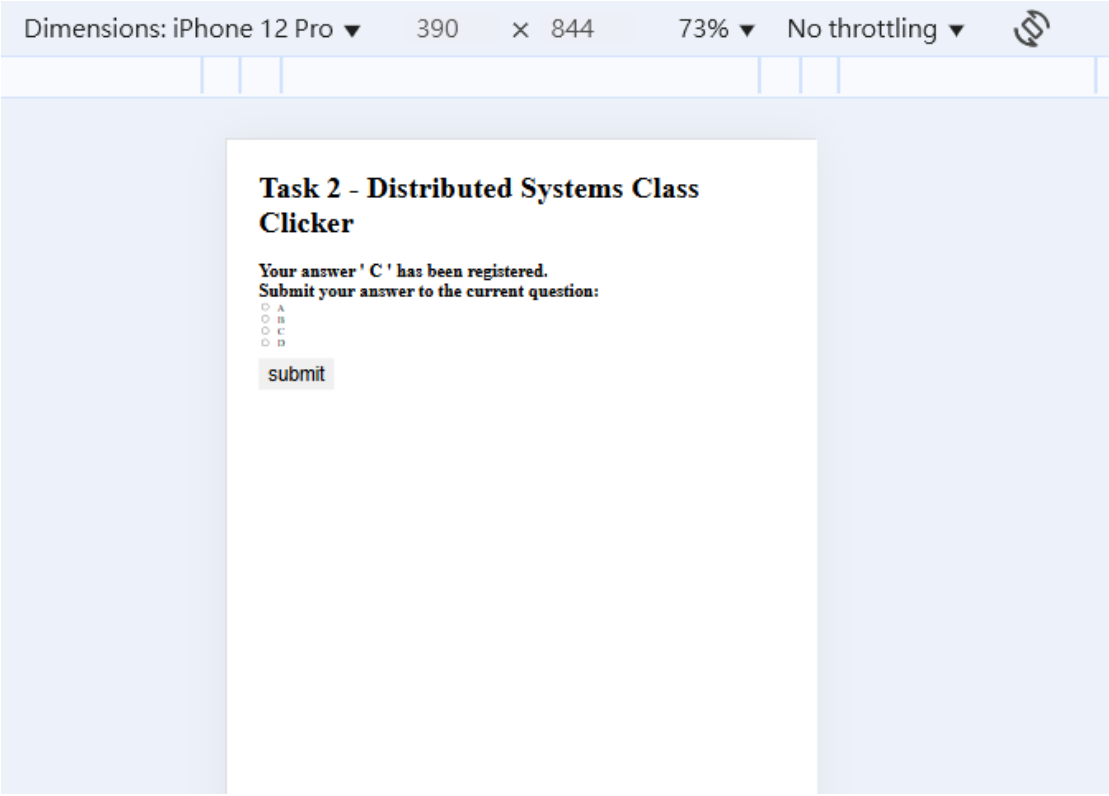
Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh



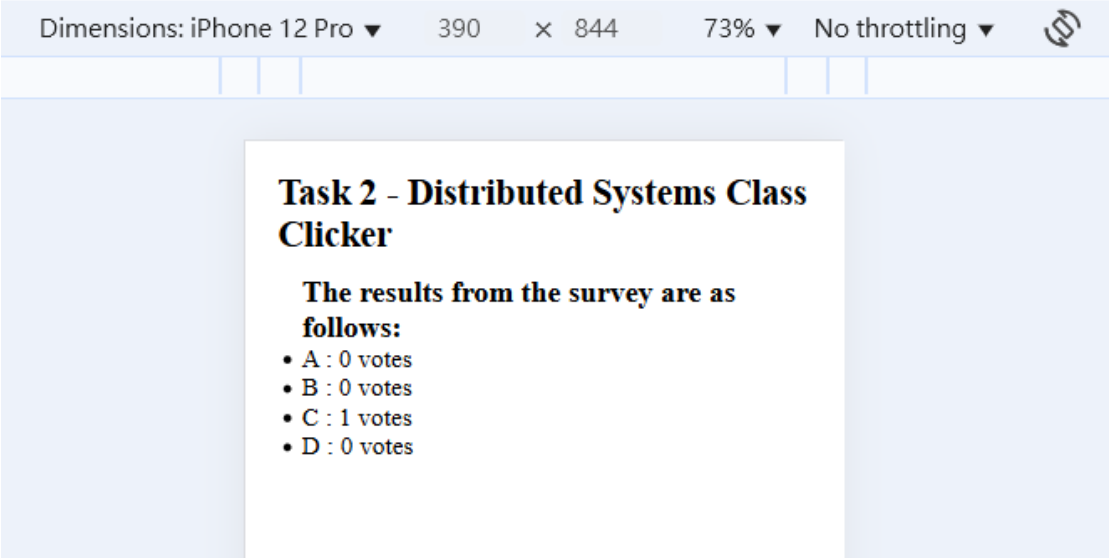
Input page in Mobile phone:



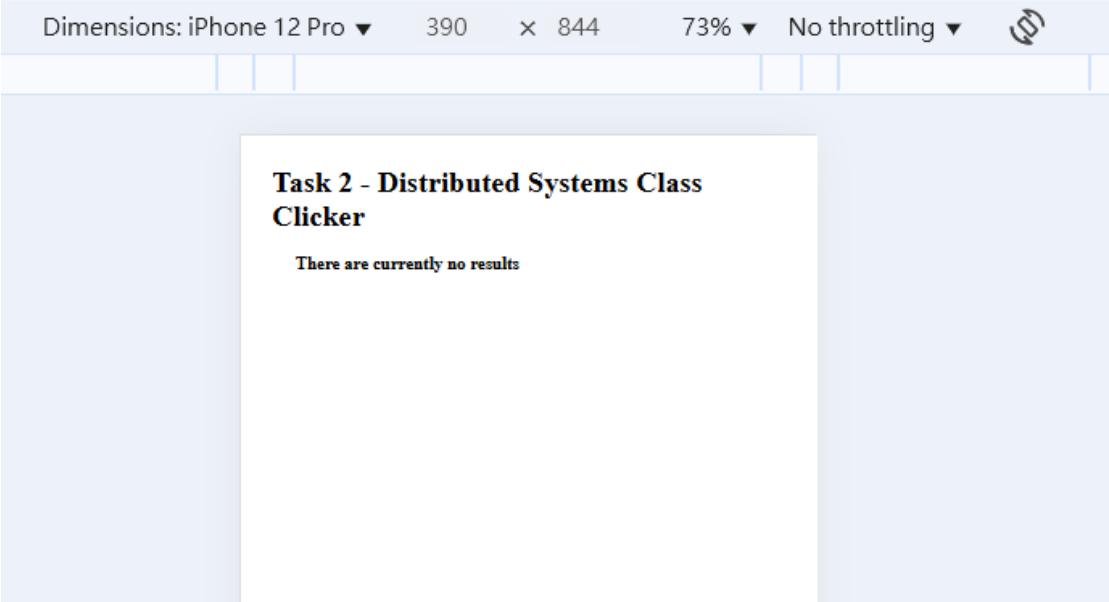
Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh



Output page in Mobile phone:



Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh



2. Code snippets for producing clicker output.

```java
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    String path = request.getServletPath();
    //for returning to /index.jsp page
    if ("/submit".equals(path)) {
        // Redirect to index.jsp (front page)
        response.sendRedirect( location: "index.jsp");
        return;
    }
    //check whether user had clicked the ABCD and submit or not
    if ("/getResults".equals(path)) {
        Map<String, Integer> answer_result_responses = responses.getResponses();
        boolean isClicked = false;
        for (Integer count : answer_result_responses.values()) {
            if (count > 0) {
                isClicked = true;
                break;
            }
        }
        if (isClicked) {
            //if the result is clicked, set up the records of resultMap
            request.setAttribute( s: "resultsMap", answer_result_responses);
        }
        // Clear results after display
        responses.clearResponses();
        //send out the request to /result.jsp
        request.getRequestDispatcher( s: "/result.jsp").forward(request, response);
    }
}
```

doGet() in Task2Servlet.java: Used to receive the click records.

Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

```
//to post the submission of clicker
@Override  no usages
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    String answer = request.getParameter( s: "answer");

    if (answer != null) {
        //set into reponses model
        responses.addResponse(answer);
        //show out feedback shows the registered answer
        request.setAttribute( s: "feedback",  o: "Your answer ' " + answer + " ' has been registered.");
    }
    //send out the request
    request.getRequestDispatcher( s: "/index.jsp").forward(request, response);

}
```

doPost() in Task2Servlet.java: Used to post the click submission from user

Code snippets for producing clicker output:

```
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    String answer = request.getParameter("answer");

    if (answer != null) {
        //set into reponses model
        responses.addResponse(answer);
        //show out feedback shows the registered answer
        request.setAttribute("feedback", "Your answer ' " + answer + " ' has been
registered.");
    }
    //send out the request
    request.getRequestDispatcher("/index.jsp").forward(request, response);

}
//to get the clicker result Map
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    String path = request.getServletPath();
```

```
        //for returning to /index.jsp page
        if ("/submit".equals(path)) {
            // Redirect to index.jsp (front page)
            response.sendRedirect("index.jsp");
            return;
        }
        //check whether user had clicked the ABCD and submit or not
        if ("/getResults".equals(path)) {
            Map<String, Integer> answer_result_responses =
responses.getResponses();
            boolean isClicked = false;
            for (Integer count : answer_result_responses.values()) {
                if (count > 0) {
                    isClicked = true;
                    break;
                }
            }
            if (isClicked) {
                //if the result is clicked, set up the records of resultMap
                request.setAttribute("resultsMap", answer_result_responses);
            }
            // Clear results after display
            responses.clearResponses();
            //send out the request to /result.jsp
            request.getRequestDispatcher("/result.jsp").forward(request, response);
        }
}
```

result.jsp:

```
<body>
<h1>Task 2 - Distributed Systems Class Clicker</h1>
<ul>
```

```
<%if (request.getAttribute("resultsMap")!= null){%>
<label>The results from the survey are as follows: </label>
<%
   Map<String, Integer> resultsMap = (Map<String, Integer>)
request.getAttribute("resultsMap");
   for(Map.Entry<String, Integer> entry : resultsMap.entrySet()){
%>
<li><%=entry.getKey()%> : <%=entry.getValue()%> votes</li>
<%}%>
</ul>
<%} else{ %>
<label><%= request.getAttribute("message") != null ?
request.getAttribute("message") : "There are currently no results" %></label>
<% } %>

</body>
```

<u>Task3:</u>

1. **Screen shots of two uses of the input page (two different national parks) and the corresponding output pages.**

2 Input pages for national parks on desktop:

Acadia NP:

Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

Great Smoky Mountain NP:

2 Output pages for national parks on desktop:

Acadia National Park on desktop:

Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

localhost:8080/Project1Task3_war_exploded/result?park=Acadia+NP

Connected Equip... 新分頁 已匯入 翻譯 - 搜尋

Acadia National Park

Credit: www.nps.gov

Current Conditions:
Temperature: 11°F
Humidity: 88%
Wind Speed: Calm

Credit: forecast.weather.gov

Acadia Activities

Arts and Culture

Astronomy

Biking

Boating

Camping

Climbing

Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

Connected Equip... 新分頁 已匯入 翻譯 - 搜尋

Acadia Activities

Arts and Culture

Astronomy

Biking

Boating

Camping

Climbing

Compass and GPS

Fishing

Food

Guided Tours

Hands-On

Hiking

Horse Trekking

Ice Skating

Junior Ranger Program

Paddling

Park Film

Shopping

Skiing

Snow Play

Snowmobiling

Snowshoeing

Swimming

Wildlife Watching

Credit: https://developer.nps.gov

Great Smoky Mountains National Park on desktop:

Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

Great Smoky Mountains National Park

Credit: www.nps.gov

Current Conditions:
Temperature: 43°F
Humidity: 100%
Wind Speed: W 6 mph

Credit: forecast.weather.gov

Great Smoky Mountains Activities

Arts and Culture

Astronomy

Auto and ATV

Biking

Camping

Fishing

Food

Great Smoky Mountains Activities

Arts and Culture

Astronomy

Auto and ATV

Biking

Camping

Fishing

Food

Guided Tours

Hands-On

Hiking

Horse Trekking

Junior Ranger Program

Museum Exhibits

Park Film

Shopping

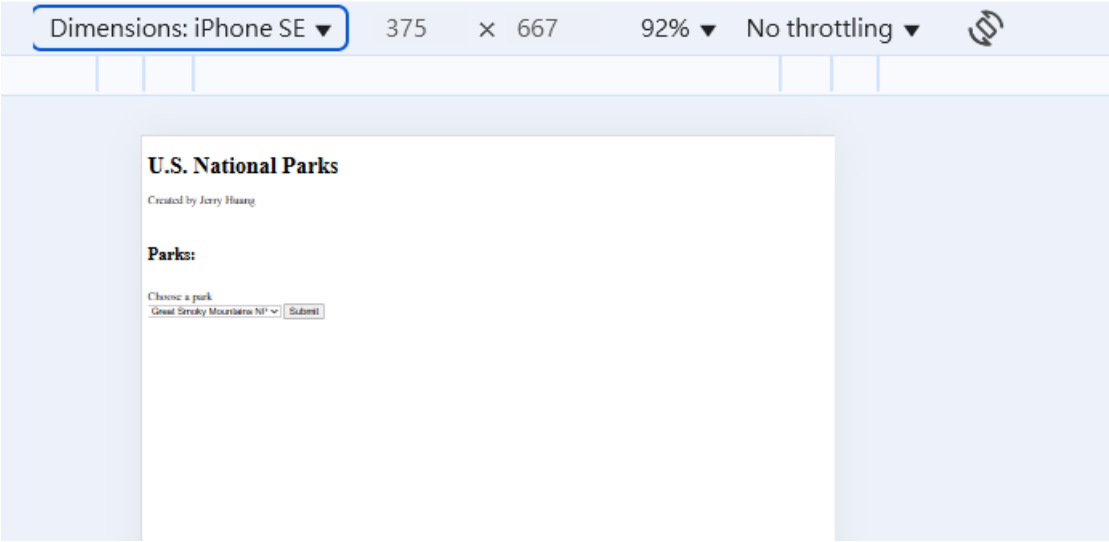
Wildlife Watching

Credit: https://developer.nps.gov

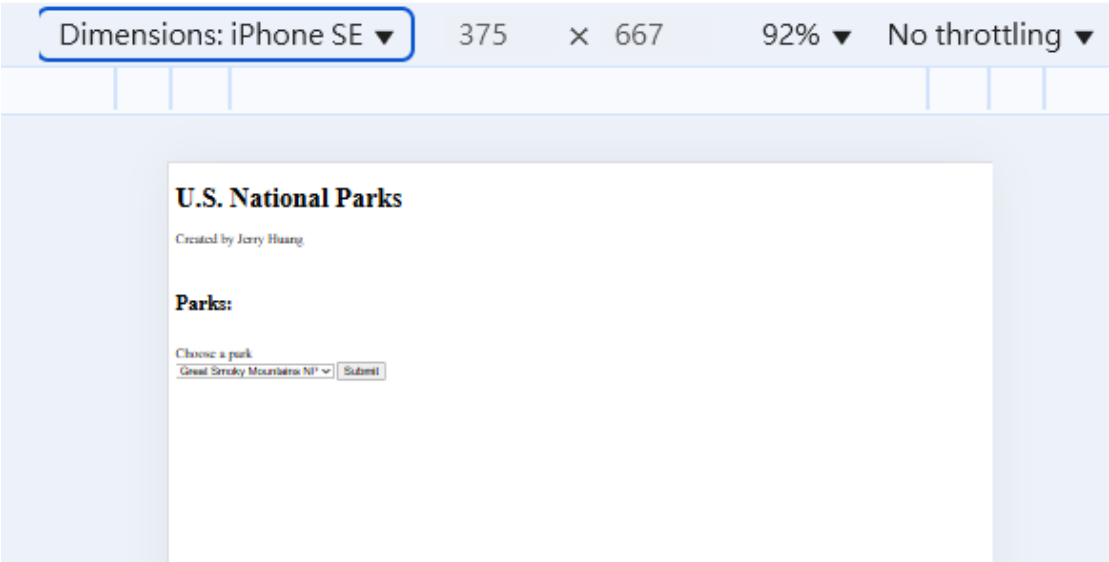
Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

2 input pages for 2 national park on mobile:

Input page for Acadia:



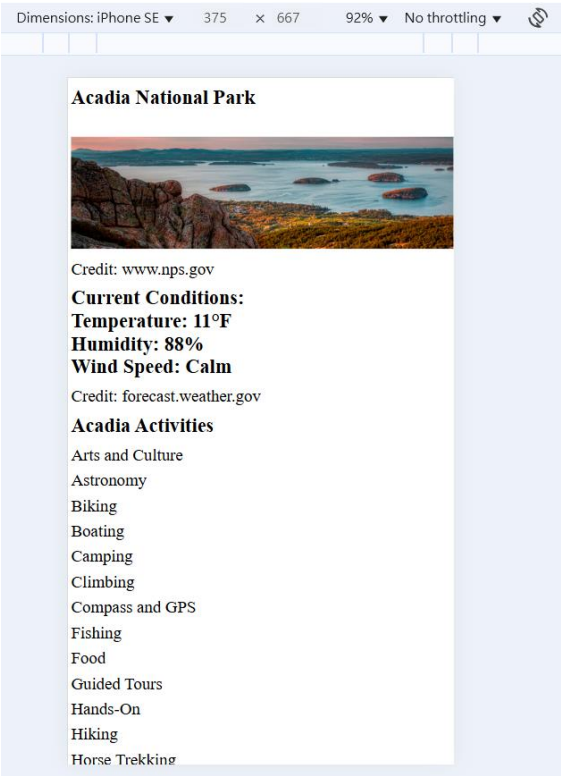
Input page for Great Smoky :

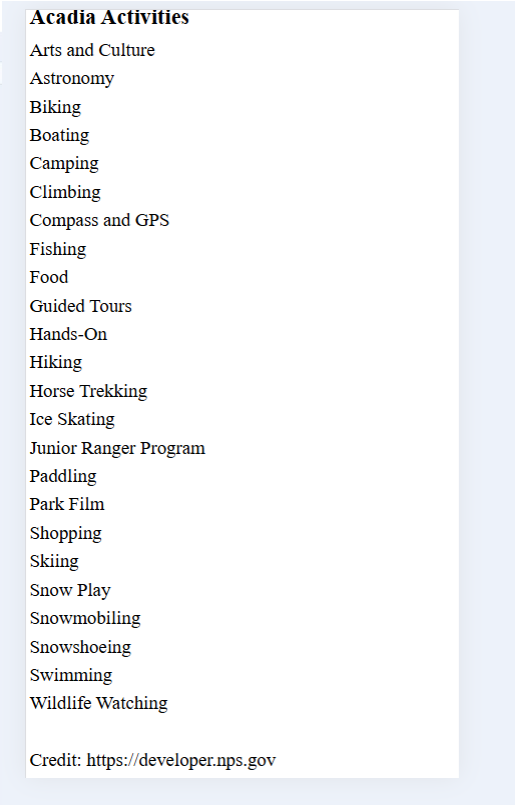


Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

2 output pages for 2 national park on mobile:

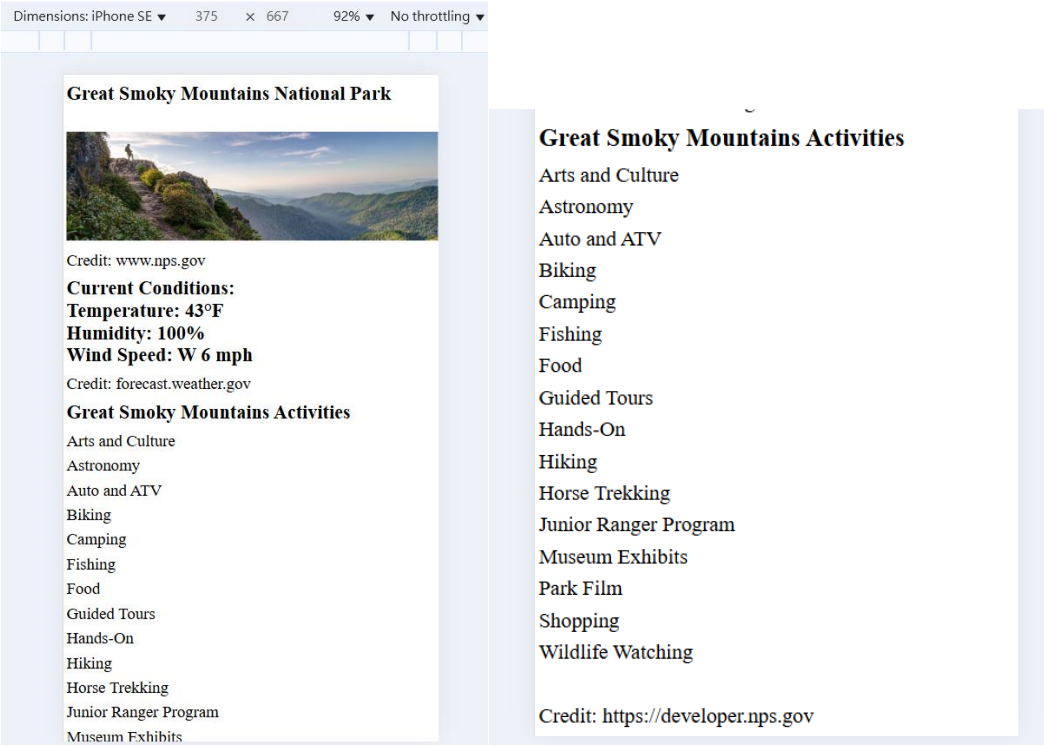
Output page for Acadia:





Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

Output page for Great Smoky:



2. **Code snippets from the Java code that screen scrapes, queries the API, and produces output.**

Note: seeing that the latest jsoup version 1.18.3 does not have the method validateTLSCertificates(false). Thus, instead of using validateTLSCertificates(false), I used the provided createTrustManager(String certType) to do the SSLHandshakeException. By using it, it can create the trust manager that does not validate certificate chains.

Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

doGet() from USNationParkServlet.java: to get the fetched image, park weather, and activities:

```java
@Override  no usages
protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

    String parkName = request.getParameter( s: "park");
    //API key
    String ApiKey = "5A0FeqBXwbswl435MQcTg5CKgnEIl659SysLgYkd";
    //read ParksData.json I draft
    List<ParkData> parks = ParkDataJSON.fetchParkDataFromJson();
    ParkData selectedPark = null;

    // Find the park by name
    for (ParkData park : parks) {
        if (park.getParkName().equalsIgnoreCase(parkName)) {
            selectedPark = park;
            break;
        }
    }
    //Use controller to handle the value
    // Make the parkName become full name
    controller.setModelParkName(selectedPark.getParkFullName());
    String parkFullName = controller.getModelParkName();
    //take off the NP from the park selection:
    String park_name_without_NP = controller.getModelParkName().substring(0, selectedPark.getParkName().length() - 2);
    // Fetch park image by parkCode
    controller.showParkImage(selectedPark.getParkCode());
    String parkImage = controller.getModelImgURL();
```

```java
    String parkImage = controller.getModelImgURL();
    // Fetch park info by park's latitude and longitude
    controller.showParkWeatherCond(selectedPark.getLatitude(), selectedPark.getLongitude());
    String weatherData = controller.getModelWeatherCondition();
    // Fetch activities by park code and ApiKey
    try {
        controller.showParkActivitiesList(ApiKey, selectedPark.getParkCode());
    } catch (NoSuchAlgorithmException e) {
        throw new RuntimeException(e);
    } catch (KeyManagementException e) {
        throw new RuntimeException(e);
    }
    List<String> activities = controller.getModelActivitiesList();

    // Set attributes to pass to result.jsp
    request.setAttribute( s: "parkFullName", parkFullName);
    request.setAttribute( s: "parkImage", parkImage);
    request.setAttribute( s: "imageCredit",  o: "www.nps.gov");
    request.setAttribute( s: "weather", weatherData);
    request.setAttribute( s: "weatherCredit",  o: "forecast.weather.gov");
    request.setAttribute( s: "parkName", park_name_without_NP);
    request.setAttribute( s: "activities", activities);
    request.setAttribute( s: "activitiesCredit",  o: "https://developer.nps.gov");

    // Forward to result.jsp
    RequestDispatcher dispatcher = request.getRequestDispatcher( s: "result.jsp");
    dispatcher.forward(request, response);
```

```java
    RequestDispatcher dispatcher = request.getRequestDispatcher( s: "result.jsp");
    dispatcher.forward(request, response);
}
}
```

Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

Functions used in USNationParkModel:

fetchParkImage:

Note: to fetch the image, I inspect to check the element for picture that asked to provided on github. To screen scrap the image to my web page, I try to use LLM model to extract the form of the image. Finally extract the image from style and set into the webpage.

```java
    public String fetchParkImage(String parkCode) {  1 usage
        String searchImgURL = "https://www.nps.gov/" + parkCode + "/index.htm";
        try {
            //SSLHandshakeException
            createTrustManager( certType: "TLSV1.3");

            // Connect to the park's page on www.nps.gov
            Document doc = Jsoup.connect(searchImgURL).timeout( t: 5000).get();

            // By checking the html code from website, select the div id='HeroBanner' image
            Element heroBannerDiv = doc.select( cssQuery: "div#HeroBanner.HeroBanner").first();

            // If the div is found, extract the background image URL
            if (heroBannerDiv != null) {
                // Find the div with the class "picturefill-background"
                Element backgroundDiv_element = heroBannerDiv.select( cssQuery: "div.picturefill-background").first();
                // if the element has value then fetch the image
                if (backgroundDiv_element != null) {
                    // Extract the background image URL from the style attribute
                    String style = backgroundDiv_element.attr( attributeKey: "style");

                    // Use regex to fetch the URL inside the background-image
                    String imageURL = style.replaceAll( regex: ".*url\\(['\"]?(.*?)['\"]?\\).*",  replacement: "$1");

                    // Return image URL
                    return "https://www.nps.gov" + imageURL;
```

```java
                    // Use regex to fetch the URL inside the background-image
                    String imageURL = style.replaceAll( regex: ".*url\\(['\"]?(.*?)['\"]?\\).*",  replacement: "$1");

                    // Return image URL
                    return "https://www.nps.gov" + imageURL;
                }
            }
            return "Image not found";
        } catch (IOException e) {
            e.printStackTrace();
            return "Fetching image failed";
        } catch (NoSuchAlgorithmException e) {
            throw new RuntimeException(e);
        } catch (KeyManagementException e) {
            throw new RuntimeException(e);
        }
    }
```

fetchParkWeather:

Note: to fetch the image, I inspect to check the element for temperature, humidity and windspeed that asked to provide on github. To get values to my web page, I try to use LLM model to extract the value of the element. Finally extract the value from

Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

css element and put it into my website.

```java
    // To Fetch park weather using latitude and longitude
    public String fetchParkWeather(double latitude, double longitude) {  1 usage
        //API URL
        String api_URL = "https://forecast.weather.gov/MapClick.php?lat=" + latitude + "&lon=" + longitude;

        try {
            //SSLHandshakeException
            createTrustManager( certType: "TLSV1.3");
            // Using Jsoup to fetch the webpage
            Document doc = Jsoup.connect(api_URL).timeout( i: 5000).get();

            // Extract the temperature value from the webpage
            Element temperature_element = doc.selectFirst( cssQuery: "p.myforecast-current-lrg");
            // Extract the humidity value from the webpage
            Element humidity_element = doc.selectFirst( cssQuery: "td:contains(Humidity) + td");
            // Extract the wind speed value from the webpage
            Element windSpeed_element = doc.selectFirst( cssQuery: "td:contains(Wind) + td");

            //make it into text
            String temperature = temperature_element.text();
            String humidity = humidity_element.text();
            String windSpeed = windSpeed_element.text();
            // Return to String
            return String.format("Temperature: %s<br>Humidity: %s<br>Wind Speed: %s", temperature, humidity, windSpeed);

        } catch (IOException e) {
            e.printStackTrace();
            return "Data Loading failed";
```

```java
            String windSpeed = windSpeed_element.text();
            // Return to String
            return String.format("Temperature: %s<br>Humidity: %s<br>Wind Speed: %s", temperature, humidity, windSpeed);

        } catch (IOException e) {
            e.printStackTrace();
            return "Data Loading failed";
        } catch (NoSuchAlgorithmException e) {
            throw new RuntimeException(e);
        } catch (KeyManagementException e) {
            throw new RuntimeException(e);
        }
    }
```

fetchParkActivities:

```java
    public List<String> fetchParkActivities(String api_key, String parkCode) throws NoSuchAlgorithmException, KeyManagementException {

        // Use API to fetch the activities
        String api_URL = "https://developer.nps.gov/api/v1/activities?parkCode=" + parkCode + "&api_key=" + api_key;

        // list for activities
        List<String> listOfActivities = new ArrayList<>();

        try {
            //SSLHandshakeException
            createTrustManager( certType: "TLSV1.3");
            // Fetch the data from the API
            String json = Jsoup.connect(api_URL).ignoreContentType( b: true).execute().body();

            // Use Gson to parse Json data
            Gson gson = new Gson();
            ActivitiesList activitiesResponse = gson.fromJson(json, ActivitiesList.class);
            // Add the activities into activity list
            for (Activity activity : activitiesResponse.getData()) {
                listOfActivities.add(activity.getName());
            }

        } catch (IOException e) {
            e.printStackTrace();
        }
```

Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

```java
        } catch (IOException e) {
            e.printStackTrace();
        }

        // Return the list of activities
        return listOfActivities;
    }
```

ParkDataJSON.java:

```java
//Used to read JSON file
public class ParkDataJSON {  1 usage

    private static final String JSON_FILE_PATH = "C:\\Users\\USER\\IdeaProjects\\Project1Task3\\ParksData.json";  1 usage

    public static List<ParkData> fetchParkDataFromJson() throws IOException {  1 usage
        Gson gson = new Gson();
        FileReader reader = new FileReader(JSON_FILE_PATH);
        // Deserialize the JSON data into a list of Park objects
        ParkDataList parksDataList = gson.fromJson(reader, ParkDataList.class);
        return parksDataList.getParks();
    }
}
```

USNationParkController.java: Use Controller to fetch data

```java
// Fetch and print out imgURL
public void showParkImage(String parkCode) {  1 usage
    String imageUrl = model.fetchParkImage(parkCode);
    model.setImageURL(imageUrl);
    view.displayImageURL(model.getImageURL());
}

// Fetch and print out the String of weatherCondition
public void showParkWeatherCond(double latitude, double longitude) {  1 usage
    String weatherCondition = model.fetchParkWeather(latitude, longitude);
    model.setWeatherCondition(weatherCondition);
    view.displayWeatherDetails(model.getWeatherCondition());
}

// Fetch and print out activitiesList
public void showParkActivitiesList(String apiKey, String parkCode) throws NoSuchAlgorithmException, KeyManagementException {  1
    List<String> activities = model.fetchParkActivities(apiKey, parkCode);
    model.setActivitiesList(activities);
    view.displayActivities(model.getActivitiesList());
}
```

Code snippets from the Java code that screen scrapes, queries the API, and produces output.

doGet() from USNationParkServlet.java:

Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

```
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

    String parkName = request.getParameter("park");
    //API key
    String ApiKey = "5A0FeqBXwbswl435MQcTg5CKgnEIl659SysLgYkd";
    //read ParksData.json I draft
    List<ParkData> parks = ParkDataJSON.fetchParkDataFromJson();
    ParkData selectedPark = null;

    // Find the park by name
    for (ParkData park : parks) {
        if (park.getParkName().equalsIgnoreCase(parkName)) {
            selectedPark = park;
            break;
        }
    }
    //Use controller to handle the value
    // Make the parkName become full name
    controller.setModelParkName(selectedPark.getParkFullName());
    String parkFullName = controller.getModelParkName();
    //take off the NP from the park selection:
    String park_name_without_NP = controller.getModelParkName().substring(0,
selectedPark.getParkName().length() - 2);
    // Fetch park image by parkCode
    controller.showParkImage(selectedPark.getParkCode());
    String parkImage = controller.getModelImgURL();
    // Fetch park info by park's latitude and longitude
    controller.showParkWeatherCond(selectedPark.getLatitude(),
selectedPark.getLongitude());
```

```
        String weatherData = controller.getModelWeatherCondition();
        // Fetch activities by park code and ApiKey
        try {
            controller.showParkActivitiesList(ApiKey, selectedPark.getParkCode());
        } catch (NoSuchAlgorithmException e) {
            throw new RuntimeException(e);
        } catch (KeyManagementException e) {
            throw new RuntimeException(e);
        }
        List<String> activities = controller.getModelActivitiesList();

        // Set attributes to pass to result.jsp
        request.setAttribute("parkFullName", parkFullName);
        request.setAttribute("parkImage", parkImage);
        request.setAttribute("imageCredit", "www.nps.gov");
        request.setAttribute("weather", weatherData);
        request.setAttribute("weatherCredit", "forecast.weather.gov");
        request.setAttribute("parkName", park_name_without_NP);
        request.setAttribute("activities", activities);
        request.setAttribute("activitiesCredit", "https://developer.nps.gov");

        // Forward to result.jsp
        RequestDispatcher dispatcher = request.getRequestDispatcher("result.jsp");
        dispatcher.forward(request, response);
    }
```

USNationParkModel.java: Code Used for producing output

```
    public String fetchParkImage(String parkCode) {
        String searchImgURL = "https://www.nps.gov/" + parkCode + "/index.htm";
        try {
            //SSLHandshakeException
            createTrustManager("TLSV1.3");
```

```
            // Connect to the park's page on www.nps.gov
            Document doc = Jsoup.connect(searchImgURL).timeout(5000).get();

            // By checking the html code from website, select the div id='HeroBanner'
image
            Element heroBannerDiv = doc.select("div#HeroBanner.HeroBanner").first();

            // If the div is found, extract the background image URL
            if (heroBannerDiv != null) {
                // Find the div with the class "picturefill-background"
                Element backgroundDiv_element =
heroBannerDiv.select("div.picturefill-background").first();
                // if the element has value then fetch the image
                if (backgroundDiv_element != null) {
                    // Extract the background image URL from the style attribute
                    String style = backgroundDiv_element.attr("style");

                    // Use regex to fetch the URL inside the background-image
                    String imageURL = style.replaceAll(".*url\\(['\"]?(.*?)['\"]?\\).*",
"$1");

                    // Return image URL
                    return "https://www.nps.gov" + imageURL;
                }
            }
            return "Image not found";
        } catch (IOException e) {
            e.printStackTrace();
            return "Fetching image failed";
        } catch (NoSuchAlgorithmException e) {
```

```
            throw new RuntimeException(e);
        } catch (KeyManagementException e) {
            throw new RuntimeException(e);
        }
}



// To Fetch park weather using latitude and longitude
public String fetchParkWeather(double latitude, double longitude) {
        //API URL
        String api_URL = "https://forecast.weather.gov/MapClick.php?lat=" + latitude +
"&lon=" + longitude;

        try {
            //SSLHandshakeException
            createTrustManager("TLSV1.3");
            // Using Jsoup to fetch the webpage
            Document doc = Jsoup.connect(api_URL).timeout(5000).get();

            // Extract the temperature value from the webpage
            Element temperature_element = doc.selectFirst("p.myforecast-current-
lrg");
            // Extract the humidity value from the webpage
            Element humidity_element = doc.selectFirst("td:contains(Humidity) + td");
            // Extract the wind speed value from the webpage
            Element windSpeed_element = doc.selectFirst("td:contains(Wind) + td");

            //make it into text
            String temperature = temperature_element.text();
            String humidity = humidity_element.text();
            String windSpeed = windSpeed_element.text();
```

```
            // Return to String
            return String.format("Temperature: %s<br>Humidity: %s<br>Wind Speed:
%s", temperature, humidity, windSpeed);

        } catch (IOException e) {
            e.printStackTrace();
            return "Data Loading failed";
        } catch (NoSuchAlgorithmException e) {
            throw new RuntimeException(e);
        } catch (KeyManagementException e) {
            throw new RuntimeException(e);
        }
}

// To fetch activities for the park from NPS API
public List<String> fetchParkActivities(String api_key, String parkCode) throws
NoSuchAlgorithmException, KeyManagementException {

        // Use API to fetch the activities
        String api_URL = "https://developer.nps.gov/api/v1/activities?parkCode=" +
parkCode + "&api_key=" + api_key;

        // list for activities
        List<String> listOfActivities = new ArrayList<>();

        try {
            //SSLHandshakeException
            createTrustManager("TLSV1.3");
            // Fetch the data from the API
            String json =
Jsoup.connect(api_URL).ignoreContentType(true).execute().body();
```

```
            // Use Gson to parse Json data
            Gson gson = new Gson();
            ActivitiesList activitiesResponse = gson.fromJson(json, ActivitiesList.class);
            // Add the activities into activity list
            for (Activity activity : activitiesResponse.getData()) {
                listOfActivities.add(activity.getName());
            }

        } catch (IOException e) {
            e.printStackTrace();
        }

        // Return the list of activities
        return listOfActivities;
}
```

ParkDataJSON.java:

```
public class ParkDataJSON {

    private static final String JSON_FILE_PATH =
"C:\\Users\\USER\\IdeaProjects\\Project1Task3\\ParksData.json";

    public static List<ParkData> fetchParkDataFromJson() throws IOException {
        Gson gson = new Gson();
        FileReader reader = new FileReader(JSON_FILE_PATH);
        // Deserialize the JSON data into a list of Park objects
        ParkDataList parksDataList = gson.fromJson(reader, ParkDataList.class);
        return parksDataList.getParks();
    }
```

```
}
```

USNationParkController.java : Used for producing output:

```
// Fetch and print out imgURL
public void showParkImage(String parkCode) {
    String imageUrl = model.fetchParkImage(parkCode);
    model.setImageURL(imageUrl);
    view.displayImageURL(model.getImageURL());
}

// Fetch and print out the String of weatherCondition
public void showParkWeatherCond(double latitude, double longitude) {
    String weatherCondition = model.fetchParkWeather(latitude, longitude);
    model.setWeatherCondition(weatherCondition);
    view.displayWeatherDetails(model.getWeatherCondition());
}

// Fetch and print out activitiesList
public void showParkActivitiesList(String apiKey, String parkCode) throws
NoSuchAlgorithmException, KeyManagementException {
    List<String> activities = model.fetchParkActivities(apiKey, parkCode);
    model.setActivitiesList(activities);
    view.displayActivities(model.getActivitiesList());
}
```