

# Hands-on Lab: Develop an Application for Healthcare Census



Estimated time needed: 40 minutes

## What you will learn

In this healthcare data analysis interface, you will learn to build interactive forms using HTML and employ JavaScript to collect and manage patient data.

This project will help you understand DOM manipulation and searching techniques based on a health condition. Moreover, you will acquire skills in dynamically generating reports within a webpage, showcasing statistical insights derived from the data. This practical exercise also emphasizes the application of data-driven decision making in healthcare analytics.

The insights and skills acquired through working on this healthcare data analysis interface project will form a solid basis for your final project.

## Learning objectives

After completing this lab, you will be able to:

- **Interactive data input interface:** Develop an understanding of front-end web development by creating an interactive interface for collecting patient data. You will learn to use HTML forms and input elements, validate user input, and handle various types of data entry (text, radio buttons, number inputs, and dropdowns).
- **Data processing and analysis:** Learn data management and searching techniques using JavaScript. You will explore data manipulation concepts such as storing data in arrays of objects and filtering data based on conditions.
- **Dynamic report generation:** You will dynamically generate and display reports within a webpage. This action involves updating the HTML content based on the processed data, presenting statistical information in an organized and understandable manner, and manipulating the DOM to reflect changes instantly.
- **User interaction and event handling:** Practice event-driven programming and user interaction. You will learn how to handle user-triggered events (button clicks and navigation links) and respond with appropriate search queries.

## Prerequisites

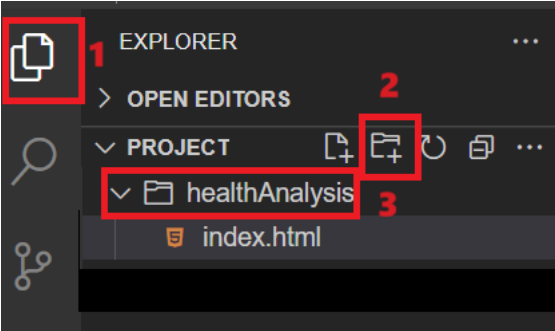
- Basic knowledge of HTML and GitHub.
- Basic understanding of arrays, array methods, strings, objects, and functions.
- Web browser with a console (Chrome DevTools, Firefox Console, and so on).

## Setting up the environment

1. You need to create one blank online GitHub repository and name it `heath_census`.

**Important!** Make sure your repository is public.

2. Create files and folder structure as given in the instructions.
  - On the window to the right, click **Explorer**, as shown at number 1 in the screenshot below.
  - Then click the project folder and click the folder icon highlighted in red color at number 2 in the screenshot.
  - Enter the folder name as **healthAnalysis**. It will create a folder for you.
  - Then select the **healthAnalysis** folder shown at number 3, right click, and select **New File**.
  - Enter the file named **index.html** and click **OK**. It will create your HTML file.



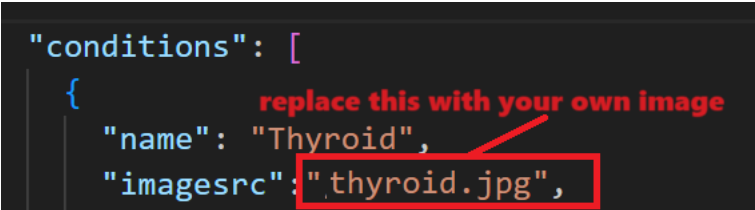
3. Create the JavaScript file as given in the instructions:

- Select the **healthAnalysis** folder again, right click and select **New File**.
- Enter the file named **health\_analysis.js** and click **OK**. It will create your JavaScript file.

4. For this project, you need to create a JSON file as well. Follow the given instructions:

- Again, right click the **healthAnalysis** folder and select **New File**.
- Enter the file name **health\_analysis.json** and click **OK**. It will create your JSON file.

5. Click this link [health\\_analysis.json](#) and copy the data, then paste it into the **health\_analysis.json** file, and then save it. You can retrieve this JSON data to find details for symptoms, prevention, and treatment related to a particular health condition.
  - You need to choose an appropriate online image related to the topic of “Thyroid, diabetes and blood\_pressure”. Please upload the image to the folder where your HTML and JS files are stored. Replace “thyroid.jpg” with the name of uploaded image as shown in the screenshot in the given below:



- For eg, If you have uplaoded an image named “thyroid-my-img.png”, change the line shown in above screenshot to:

```
"imagesrc": "thyroid-my-img.png",
```

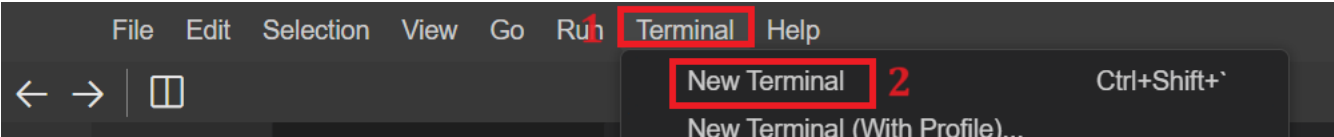
6. Create contact HTML file as given in the instructions:

- Select the **healthAnalysis** folder again, right click and select **New File**.
- Enter the file named **health\_contact.html** and click **OK**. It will create your contact HTML file.

## Perform Git commands

1. Now perform git commands as per given instructions:

- Click on the terminal shown at number 1 and then select **New Terminal** as shown as number 2 in the given screenshot.



- Then, you need to go inside the **healthAnalysis** folder. For this write the given command in the terminal and hit **Enter**.

```
cd healthAnalysis
```

```
theia@theia-richaar: /home/project$ cd healthAnalysis
```

- Then initialize this folder as a git repository by wrting given command in terminal.

git init
- Perform git add to add latest files and folder by writing given command in terminal in the git environment.

git add --a
- Then perform git commit in the terminal. While performing git commit, terminal can show message to set up your git config --global for user.name and user.email. If yes, then you need to perform git config command as well for user.name and user.email as given.

git config --global user.email "you@example.com"

git config --global user.name "Your Name"

**Note:** Replace data within qoutes with your own details.

Then perform commit command as given:

git commit -m "message"
- Next, perform git push just by writing given command in terminal one after another.

git remote add origin2 <git-repo-url>

**Note:** Replace entire <git-repo-url> with your GitHub repository url such as git remote add origin2 https://github.com//youraccountname//yourrepositoryname
- Then perform given command in terminal to push the content of your file in GitHub repository and click **Enter**.

git push origin2
- While pushing the files using git push command, it will ask you to enter username for your GitHub account in the terminal. Enter your username and then press enter. Now, it will ask for your password as well, you need to paste your **Personal Access Token** here that you generated in the first lab.

**Note:** Upon pasting this into the terminal, it will not show for security reasons, but it's already there. Simply hit enter, and it will push your files and folders to the GitHub repository.
- It will push all the files directly into your GitHub repository.

**Note:** After pasting the code, save your file.

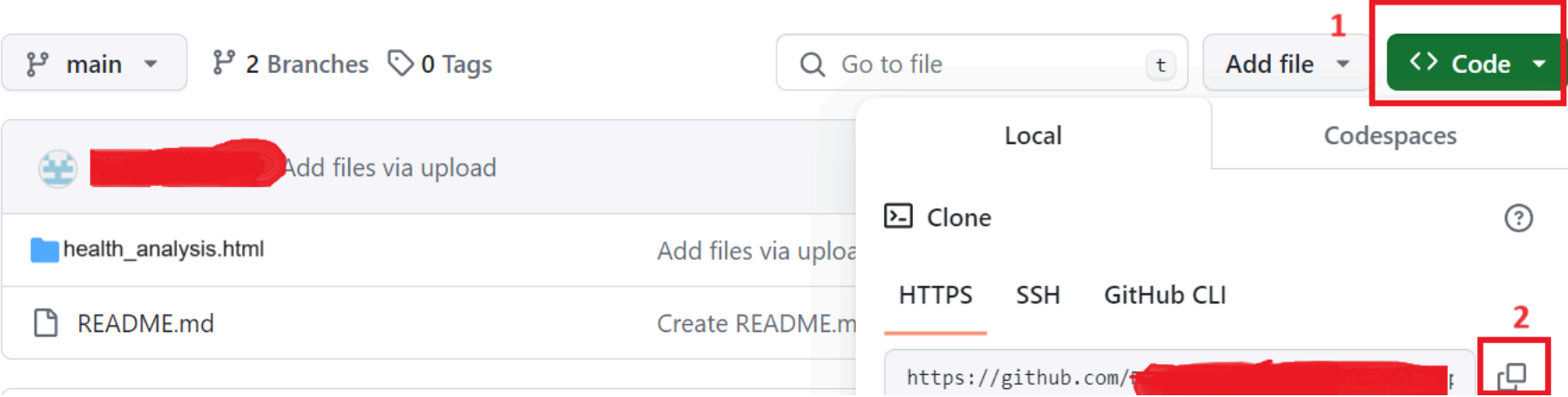
To remember

**IMPORTANT!** If you log out of the session, you need to clone the GitHub repository of this project after opening the **Skill Network Environment**. For this, you need to follow the steps:

- Open a terminal within the **Skill Network Environment** and write this command:

git clone <git-repo-link>

**Note:** Replace <git-repo-link> with your actual GitHub repository link.
- You can get your repo link by clicking the **code**, as shown in the screenshot below at number 1.
- Then copy the URL link shown at number 2.



- Perform git add, git commit and git push commands to push all of your latest files and folders in GitHub repository.
- Note:**
- After cloning you just need to use git push origin command to push all your latest work.
  - If you are working continously and you have not performed git clone yet, then use git push origin2 command to push.

Task 1: Webpage creation for *index.html*

In this task, you will create a basic HTML template structure in a file by adding the code below.

- This HTML structure sets up a webpage titled **Health Analysis Data** with a header and an empty <div> element intended to display health-related content dynamically using JavaScript.
- Include the given code below in your **index.html** file. Explanations of this code follow:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Health Analysis Data</title>
  <link rel="stylesheet" href="./health_analysis.css">
</head>
<body>
  <nav><h1>Health Analysis Census</h1>
  <ul>
    <li><a href="./index.html" id="home">Home </a></li>
    <li><a href="./health_contact.html" id="contact">Contact Us </a></li>
    <li><input type="text" id="conditionInput" placeholder="Enter a health condition"> </li>
    <li><button id='btnSearch'>Search</button>
  </li>
</ul>
</nav>
<div class="container">
  <div class="analysisForm">
    <h1>Healthcare Data Analysis</h1>
    <div>
      <label for="name">Name:</label>
      <input type="text" id="name">
    </div>
    <div>
      <label>Gender:</label>
      <label for="male">Male</label>
      <input type="radio" name="gender" id="male" value="Male">
      <label for="female">Female</label>
      <input type="radio" name="gender" id="female" value="Female">
    </div>
    <div>
      <label for="age">Age:</label>
      <input type="number" id="age">
    </div>
    <div>
      <label for="condition">Condition:</label>
      <select id="condition">
        <option value="">Select condition</option>
        <option value="Diabetes">Diabetes</option>
        <option value="Thyroid">Thyroid</option>
        <option value="High Blood Pressure">High Blood Pressure</option>
      </select>
    </div>
  </div>
</div>
```

```
<button id="addPatient">Add Patient</button>
<h2>Analysis Report</h2>
<div id="report"></div>
</div>
<div class="searchCondition">
  <div id="result"></div>
</div>
</div>
<script src="./health_analysis.js"></script>
</body>
</html>
```

- The <nav> tag signifies a navigation section within the HTML document.
- The <head> tag presents the title "Health Analysis Data" within the navigation section.
- An unordered list <ul> contains a list of the items <li>:
  - one for the home link using an anchor tag
  - another is an input box for a search bar
  - a third is for a search button to search data related to a particular health condition
- The input form constitutes of a form for entering healthcare-related data. Fields include:
  - A text input field to capture the patient's name
  - Radio buttons for selecting the patient's gender (male/female)
  - An input field specifically for entering the patient's age
  - A drop down <select> menu allowing the selection of a patient's medical condition
  - A button labeled **Add Patient** to submit the data for processing and analysis
- An <h2> tag presents the header "Analysis Report."
- The ID report identifies an empty <div> element. This action will dynamically display the generated analysis reports based on the user-entered data.
- One more empty <div> element identified by the ID result. This action will dynamically showcase the generated details for a particular health condition to showcase its symptoms and prevention methods.
- You have given line of code available at line number 12. The hyperlink has an href attribute set to "/health\_contact.html", indicating that when clicked, it will redirect the user to the "health\_contact.html" page located in the current directory.  
<li><a href="/health\_contact.html" id="contact">Contact Us </a></li>

- Apply css according to your design and color theme. For this create one css file and named as **health\_analysis.css** by selecting **healthAnalysis** folder and right click select **New File**.

You can also use the CSS provided on this link [health\\_analysis.css](#)


## Task 2: Check the output

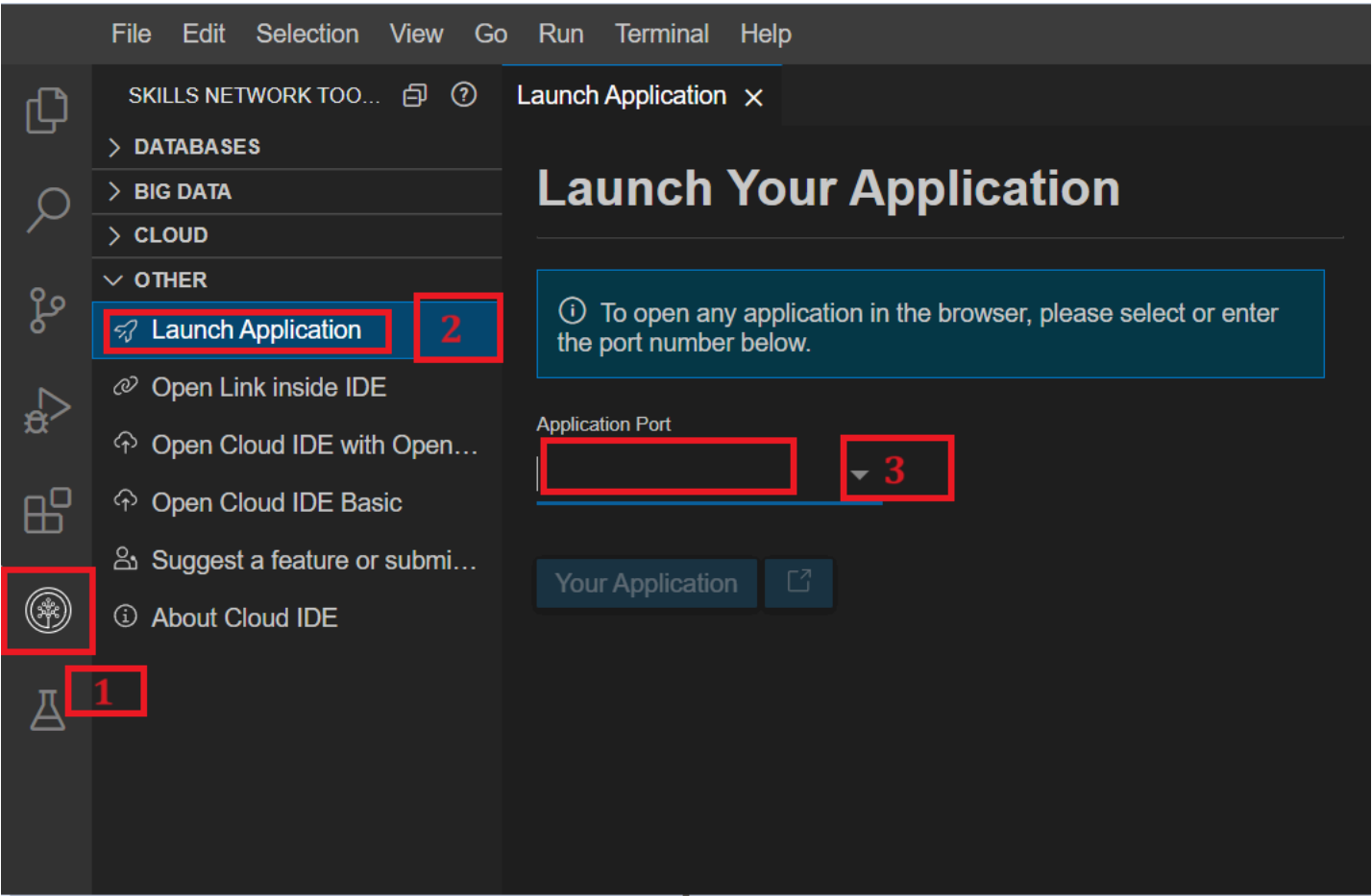
1. To view your HTML page in a browser, use the built-in Live Server extension. Select file **index.html** within the project folder and right click that file. Choose 'Open with Live Server'.
2. A notification will appear at the bottom right, indicating that the server started on port 5500.



3. Click the **Skills Network** icon on the left hand side of the screen shown at number 1 in the screenshot below. It will open the **Skills Network Toolbox**.

- Then click the **Launch Application** shown at number 2.
- Enter the port no. as 5500 at number 3.

- Click this button .



4. Your default browser will open, and you will see a folder named **healthAnalysis**. Click that folder name and it will automatically open your front page.  
**Note:** The reason why the front page opened automatically is beacuse your main html file name is **index.html** which is default file name that gets pickedup automatically.
5. It will open the front page, and you will see the output shown in the screenshot below.

# Healthcare Data Analysis

Name:

Gender:Male☐Female☐

Age:

Condition:

Select condition

Add Patient

## Analysis Report

6. Then perform the `git init`, `git add`, `git commit`, and `git push` commands to push your latest code into your GitHub repository.

**Note:** Make sure that you save your files and perform git commands, so your work is up to date in your GitHub repository.

### Task 3: Defining variables

- Now, we will initialize the variables. Include the given code in the **health\_analysis.js** file.

```
const addPatientButton = document.getElementById("addPatient");
const report = document.getElementById("report");
const btnSearch = document.getElementById('btnSearch');
const patients = [];

// addPatientButton: The button used to add patient data
// report: The HTML element where you will see analysis reports displayed
// btnSearch: The variable name of the button which displays the search results when clicked
// An empty array named patients is also created to store the collected patient data.
```

### Task 4: Creating the function that adds the patient details

Include the given code below, which has the function `addPatient()`. This function captures user-entered data from the HTML form elements: name, gender, age, and medical condition. It ensures that all fields have valid inputs.

```
function addPatient() {
  const name = document.getElementById("name").value;
  const gender = document.querySelector('input[name="gender"]:checked');
  const age = document.getElementById("age").value;
  const condition = document.getElementById("condition").value;
  if (name && gender && age && condition) {
    patients.push({ name, gender: gender.value, age, condition });
    resetForm();
    generateReport();
  }
}
```

This function retrieves the patient's details in the form such as name, gender, age, and condition. For example, the variable `name` is defined by `const name = document.getElementById("name").value;`

Additionally, it:

- appends the patient's details to the `patients[]` array, which stores all entered patient data using the `push()` method
- resets the form fields using the `resetForm()` method to clear the input fields for the next entry
- triggers the `generateReport()` method to update and display the analysis report based on the newly added patient data

### Task 5: Create a function to reset form values

Create a function named `resetForm()`. This function clears the values of the name, gender, age, and condition fields in the HTML form by setting them to empty strings or unchecked for radio buttons, effectively resetting the form to its initial state. Hence it is ready for new data entry.

Include this code after the `addPatient()` function.

```
function resetForm() {
  document.getElementById("name").value = "";
  document.querySelector('input[name="gender"]:checked').checked = false;
  document.getElementById("age").value = "";
  document.getElementById("condition").value = "";
}
```

The above code assigns an empty value to all the fields to clear previously entered details.

### Task 6: Create the function that generates the report

If you previously logged out of the session after entering the **Skill Network Environment**, you must clone your GitHub repository for this project. Otherwise, continue with the environment you are already working in.

- Create a function named `generateReport()` to generate reports. Include the code following the `resetForm()` form function.

```
function generateReport() {
  const numPatients = patients.length;
  const conditionsCount = {
    Diabetes: 0,
    Thyroid: 0,
    "High Blood Pressure": 0,
  };
  const genderConditionsCount = {
    Male: {
      Diabetes: 0,
      Thyroid: 0,
      "High Blood Pressure": 0,
    },
    Female: {
      Diabetes: 0,
      Thyroid: 0,
      "High Blood Pressure": 0,
    },
  };
  for (const patient of patients) {
    conditionsCount[patient.condition]++;
    genderConditionsCount[patient.gender][patient.condition]++;
  }
  report.innerHTML = `Number of patients: ${numPatients}<br><br>`;
  report.innerHTML += `Conditions Breakdown:<br>`;
  for (const condition in conditionsCount) {
    report.innerHTML += `${condition}: ${conditionsCount[condition]}<br>`;
  }
  report.innerHTML += `<br>Gender-Based Conditions:<br>`;
  for (const gender in genderConditionsCount) {
    report.innerHTML += `${gender}<br>`;
    for (const condition in genderConditionsCount[gender]) {
```

```
        report.innerHTML += `&nbsp;&nbsp;&nbsp;${condition}: ${genderConditionsCount[gender][condition]}<br>`;
      }
    }
  }
  addPatientButton.addEventListener("click", addPatient);
```

- This generateReport() function calculates and constructs an analysis report based on the collected patient data stored in the patients[] array. Here's a breakdown:
  - Initialization:
    - numPatients Represents the total number of patients stored in the patients[] array
    - conditionsCount A data structure (object) initializing counters for specific medical conditions (Diabetes, Thyroid, High Blood Pressure), initially set to zero.
    - genderConditionsCount A nested object with gender-specific condition counters ( male and female) for each medical condition, also initialized to zero for each condition
  - Data processing loop:
    - Iterates through the patients[] array: Utilizes a for...of loop to iterate through each patient's data within the patients[] array
    - Increment condition counts: Increments the count for each patient's specific medical condition in the conditionsCount object.
    - Updating gender-based condition counts: Increases the count of each medical condition within the respective gender category in the genderConditionsCount object based on the patient's gender and condition
  - HTML update:
    - Update report element: Dynamically updates the HTML content within the designated report element
    - Total patients display: Displays the total number of patients
    - Conditions breakdown: Lists the counts for each medical condition in the conditionsCount object
    - Gender-based conditions display: Illustrates counts of each condition categorized by gender in the genderConditionsCount object, showing the distribution of conditions among males and females separately.
  - Event Listener
    - Now, you need to set up event listener using addPatientButton.addEventListener("click", addPatient) to add patient details when the user clicks the **Add Patient** button.

Go to your browser where your code runs, enter details, and click the **Add Patient** button. It should generate data, as shown in the screenshot below.

## Analysis Report

Number of patients: 1

Conditions Breakdown:

Diabetes: 0

Thyroid: 1

High Blood Pressure: 0

Gender-Based Conditions:

Male:

Diabetes: 0

Thyroid: 1

High Blood Pressure: 0

Female:

Diabetes: 0

Thyroid: 0

High Blood Pressure: 0

You will also see the functionality of **ResetForm** function which resets the entire form after user clicks on the **Add Patient** button.

## Task 7: Create a function for search request

This JavaScript function searchCondition() is designed to work within a web page to retrieve health condition information based on user input. Include the code below after the previous task's resetForm() function in your JavaScript file.

```
function searchCondition() {
  const input = document.getElementById('conditionInput').value.toLowerCase();
  const resultDiv = document.getElementById('result');
  resultDiv.innerHTML = '';
  fetch('health_analysis.json')
    .then(response => response.json())
    .then(data => {
      const condition = data.conditions.find(item => item.name.toLowerCase() === input);
      if (condition) {
        const symptoms = condition.symptoms.join(', ');
        const prevention = condition.prevention.join(', ');
        const treatment = condition.treatment;
        resultDiv.innerHTML += `<h2>${condition.name}</h2>`;
        resultDiv.innerHTML += ``;
        resultDiv.innerHTML += `<p><strong>Symptoms:</strong> ${symptoms}</p>`;
        resultDiv.innerHTML += `<p><strong>Prevention:</strong> ${prevention}</p>`;
        resultDiv.innerHTML += `<p><strong>Treatment:</strong> ${treatment}</p>`;
      } else {
        resultDiv.innerHTML = 'Condition not found.';
      }
    })
    .catch(error => {
      console.error('Error:', error);
      resultDiv.innerHTML = 'An error occurred while fetching data.';
    });
}
btnSearch.addEventListener('click', searchCondition);
```

This function fetches the health condition data from the **health.json** file and searches for a matching condition based on user input. Then, it displays the condition details or an error message in a designated HTML element (resultDiv).

The above code includes:

- const input = document.getElementById('conditionInput').value.toLowerCase(); This retrieves the value entered into the input field with the ID conditionInput. It converts the entered text to lowercase to ensure case-insensitive comparison.
- const resultDiv = document.getElementById('result'); resultDiv.innerHTML = ''; This retrieves the HTML element with the ID 'result'. It clears any previous content within this HTML element.
- fetch('health.json') This API method initiates a fetch request to the file named 'health.json'. It assumes a JSON file named 'health.json' is in the same directory as the HTML file.
- .then(response => response.json()) Converts the fetched response into JSON format.
- .then(data => { /\* ... \*/ }) This handles the retrieved JSON data. It searches for a health condition that matches the user input.
- const condition = data.conditions.find(item => item.name.toLowerCase() === input); This searches within the JSON data for a health condition whose name matches the entered input.
- if (condition) { /\* ... \*/ } else { /\* ... \*/ } This code checks for a matching condition. If found, it constructs HTML content to display details about the condition (name, symptoms, prevention, treatment) within the resultDiv. If the system cannot find a matching condition, it displays a 'Condition not found' message within the resultDiv.
- .catch(error => { /\* ... \*/ }) This handles any errors that might occur during the fetch request or data processing. If an error occurs, it logs it to the console and displays an error message within the resultDiv.
- Suppose you have entered **Thyroid** in search bar. After clicking on Search button it will display given information from **health\_analysis.json**.

# Thyroid



**Symptoms:** Fatigue, Weight gain or loss, Dry skin, Muscle weakness, Irregular menstrual periods

**Prevention:** Eat a balanced diet, Exercise regularly, Get regular check-ups

**Treatment:** Medication like levothyroxine may be prescribed by a doctor.

## Task 8: Webpage creation for *health\_contact.html*

- Now you need to add content in **health\_contact.html**. Copy given code in contact file html.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Contact Us</title>
  <link rel="stylesheet" href="./health_analysis.css">
  <style>
    body{
      text-align: center;
    }
    /* Some basic styling for demonstration purposes */
    #contactForm {
      max-width: 400px;
      margin: 0 auto;
    }
    #contactForm input,
    #contactForm textarea {
      width: 100%;
      margin-bottom: 15px;
      padding: 8px;
    }
    #contactForm button {
      padding: 10px 20px;
      background-color: #007bff;
      color: white;
      border: none;
      cursor: pointer;
    }
    #contactForm button:hover {
      background-color: #0056b3;
    }
  </style>
</head>
<body>
  <nav><h1>Health Analysis Census</h1>
  <ul>
    <li><a href="./index.html" id="home">Home </a></li>
    <li><a href="./health_contact.html" id="contact">Contact Us </a></li>
  </ul>
</nav>
<h1>Contact Us</h1>
<p>Feel free to contact us to know more about your conditions and for treatment methods.</p>
<form id="contactForm">
  <div>
    <label for="name">Name</label>
    <input type="text" id="name" name="name" required>
  </div>
  <div>
    <label for="email">Email</label>
    <input type="email" id="email" name="email" required>
  </div>
  <div>
    <label for="condition">Condition</label>
    <input type="text" id="condition" name="condition" required>
  </div>
  <div>
    <label for="message">Message</label>
    <textarea id="message" name="message" rows="5" required></textarea>
  </div>
  <button type="submit" onclick="thankyou()">Submit</button>
</form>
<script>
  function thankyou(){
    alert('Thank you for contacting us!')
  }
</script>
</body>
</html>
```

- Check the output of above code which will be shown as below:



# Contact Us

Feel free to contact us to know more about your conditions and for treatment methods.

Name

Email

Condition

Message

Submit

- Fill the form and click on **Submit** button. A pop up box will appear displaying **Thank you for contacting us!** message.

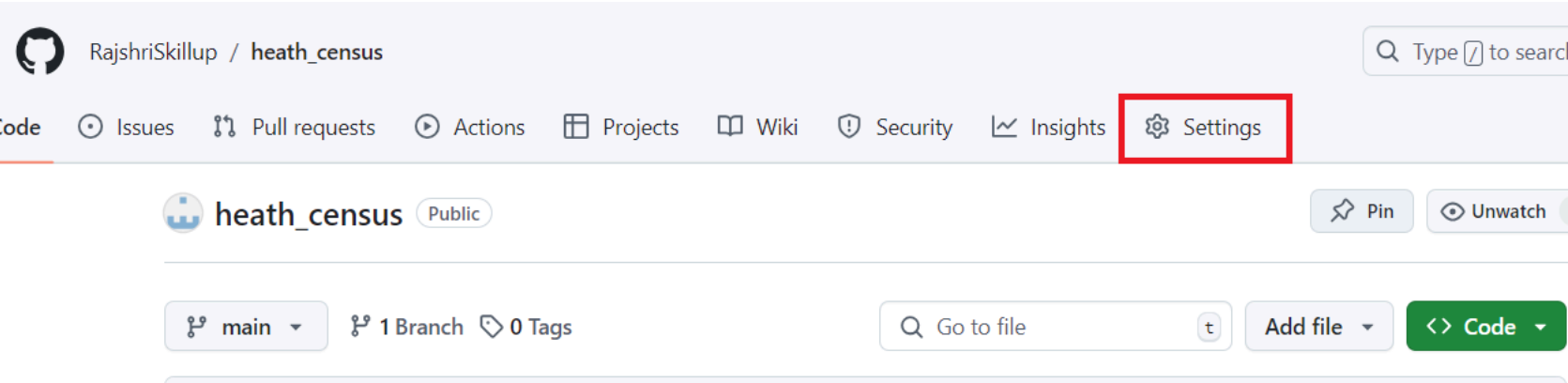
127.0.0.1:5500 says

Thank you for contacting us!

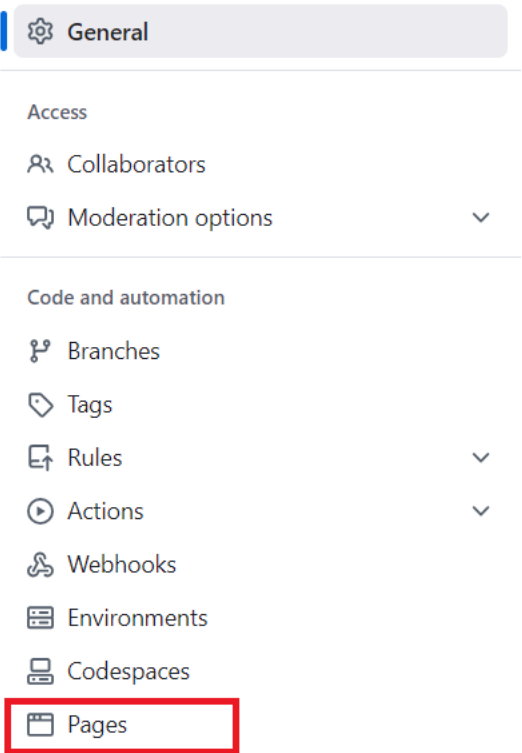
OK

## Task 9: Perform git commands and generate pages

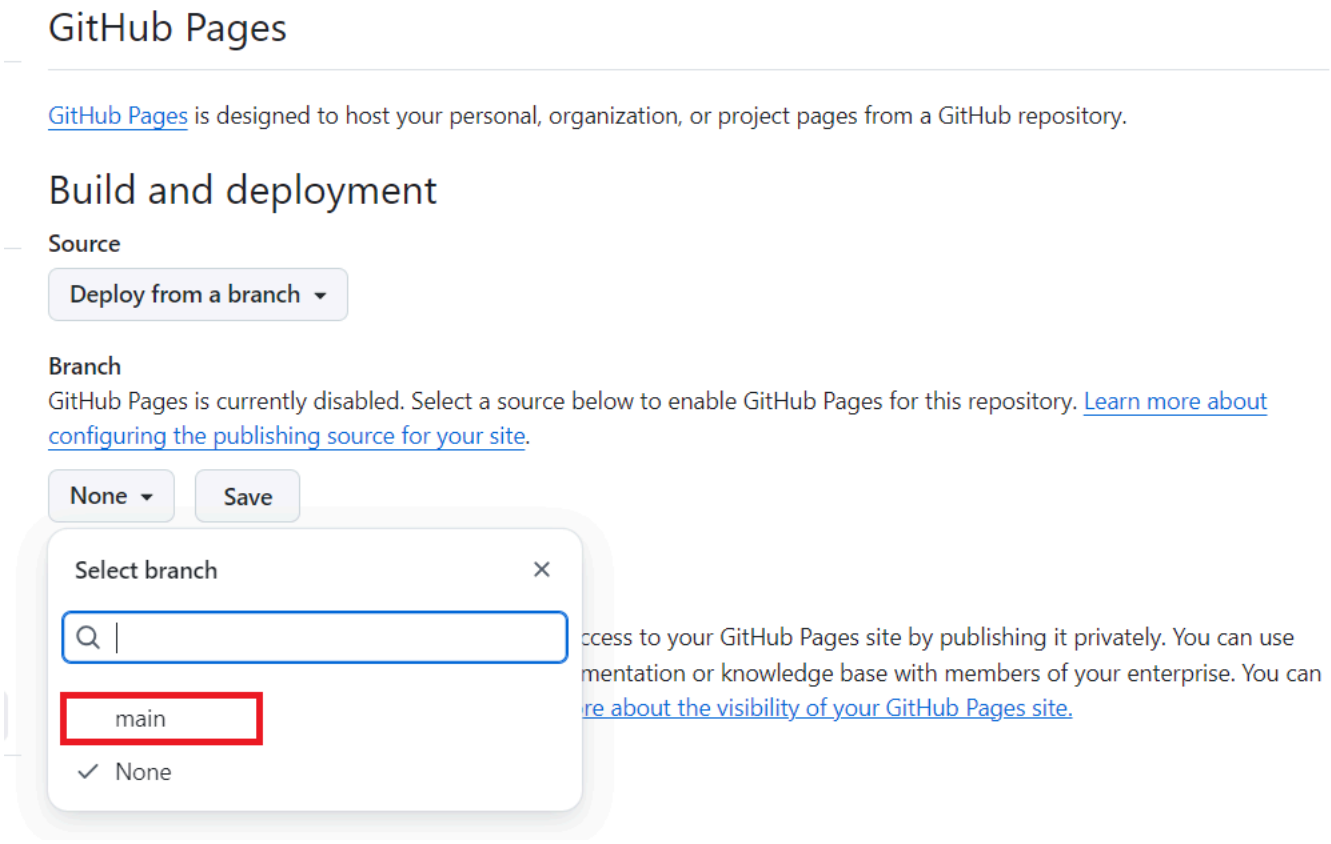
- You need to save all your files and perform git commands for them.
- Perform `git add`, `git commit`, and `git push` commands to update changes inside the **healthAnalysis** folder. Use your GitHub repository for proper code management.
- Go to your GitHub repository. Then, navigate to your site's repository that you created at this project's start.
- Under your repository name, click **Settings**. If you cannot see the **Settings** tab, select the dropdown menu, then click **Settings**.



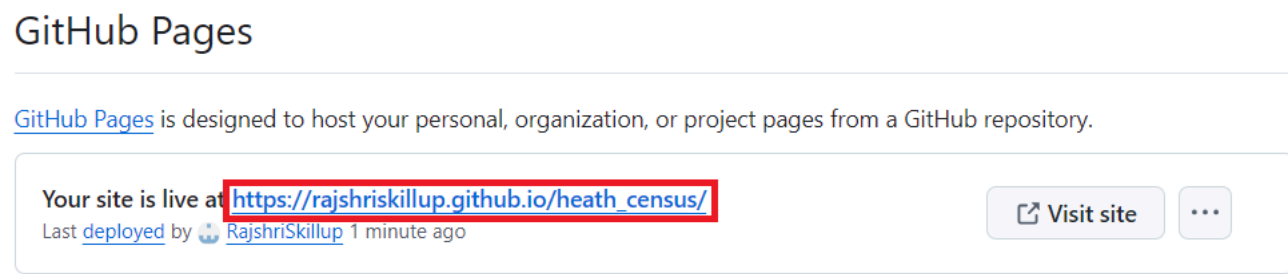
- Navigate to the left hand side navigation bar. In the **Code and Automation** section of the sidebar, click **Pages**.



- You will see the page shown below. Click the drop down menu where you see **None**, then click **main**, and then click the **Save** button.



7. Refresh your page again, and you will see the link, highlighted below.



**Note:** If you are not able to see the link, please wait for (1-2) minutes and refresh the page again.

8. Click above generated link to see your live website. Depending upon the size of the images that you must have used the images may take time to open.

**Note:** You can refer to the first lab of this course to review all the git commands.

**Congratulations! You have finished the practice lab.**

## Summary

- HTML Structure:** The HTML file establishes a structure comprising navigation links for analysis options and form elements to input patient data, including name, gender, age, and medical condition.
- JavaScript Logic:** The embedded JavaScript code manages patient data storage, providing functions to add patients, reset the form inputs, and generate an analysis report based on the entered information.
- Data Analysis Capabilities:** The code includes functionalities to filter patients based on specific conditions and age groups triggered by the navigation links. These functions prompt user input via prompts and display the corresponding analysis results within the webpage.
- Event Handling:** Event listeners are set up to respond to user interactions with the navigation links and the "Add Patient" button, facilitating dynamic updates and analysis generation based on the healthcare data provided.

© IBM Corporation. All rights reserved.