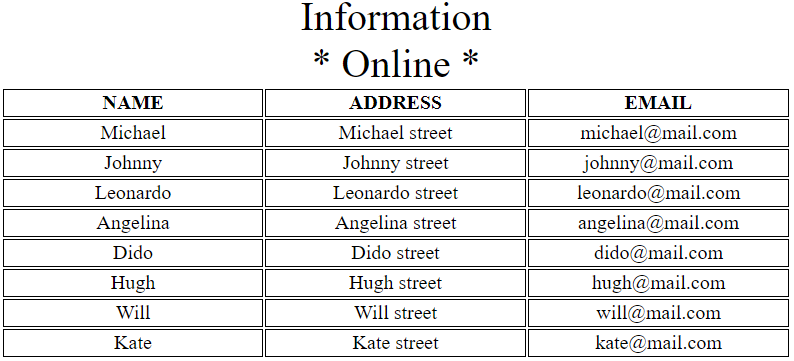
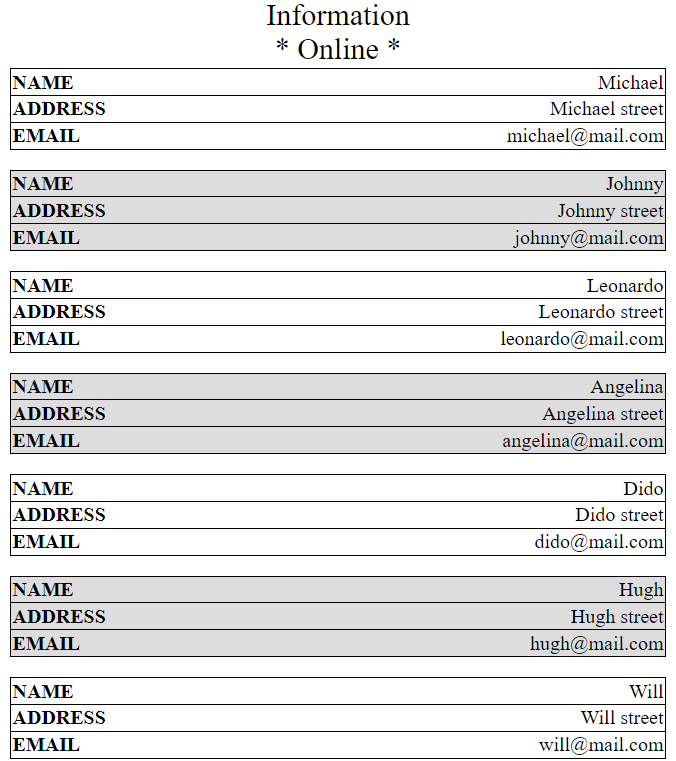
Tutorial 06 – Contact Info Mobile Web App

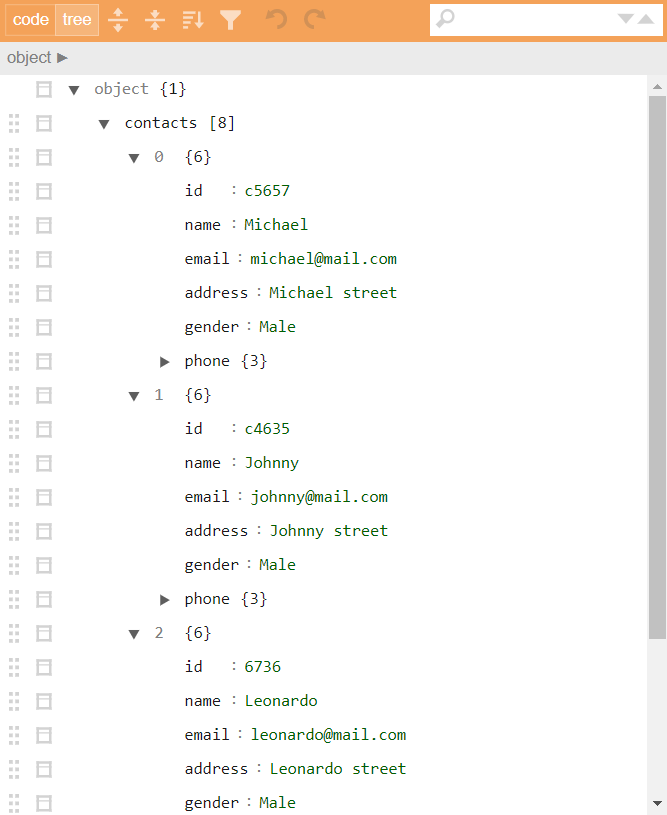
Preamble

* In the Contact Info Mobile Web application, which is a web application that can be run in browser of mobile device, data can be extracted from a JSON file, displayed with HTML table that support responsive table design and stored within the user’s browser.
* After completed this tutorial, you should be able to
  + Implement a web application to retrieve data from a JSON file
  + Store data locally within the browser with “localStorage”
  + Implement responsive table design
* To test the application, access the web application with browser of desktop computer or mobile device
* Note:
  + The page should be upload to a web server for testing
    - E.g., https://ouhklab.github.io/contact-info-mobile-web-app/

User Interface Design

* The following shows the screen captured from a browser which its width is greater than 600px.
  + 
* The following shows the screen captured from a browser which its width is small than or equal to 600px
  + 
* It can be observed if the width is small than or equal to 600px
  + Font size of the table caption becomes smaller
  + The content of the table is rearranged and the columns are collapsed
    - => Responsive table design

File Structure

* The project consists of three files
  + index.html
    - Represent the layout of the web page
  + style.css
    - Style sheet language used for describing the presentation of different element declared in the .html file
  + app.js
    - The .js file contains Javascript code
    - Include the logic implementation of the web application
  + contacts.json
    - JSON file that contains contacts of persons
      * Available at:
        + https://ouhklab.github.io/contact-info-mobile-web-app/contacts.json
      * Note:
        + Not all the contact information are included in this tutorial
        + Only name, email, address and mobile are required
    - Structure of the given JSON can be viewed by online JSON parser
      * E.g., https://jsoneditoronline.org/
    - 

Implementation procedures

* Create a new folder on the computer
  + E.g., contact-info
* Create and edit the three files (index.html, style.css and app.js) with favorite text editor
  + E.g., Notepad++, Visual Studio Code
* Upload the four files to server
* Test the application with browser of mobile device

Experiment 1. Edit the HTML file

* Create and edit the “index.html” file as follows

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8">  <meta name="viewport" content="width=device-width, initial-scale=1">  <!-- Name of the app -->  <title>Contact Info Mobile Web App</title>  <!-- Link to main style sheet -->  <link rel="stylesheet" href="style.css">  <body onload="initialize()">  <script src="app.js"></script>  <table>  <caption>Information<div id="status"></div></caption>  <thead>  <tr>  <th>Name</th>  <th>Address</th>  <th>Email</th>  </tr>  </thead>  <tbody id="tcontent">  </tbody>  </table>  </body>  </html> |

* <link rel="stylesheet" href="style.css">
  + Responsible to link to the specific .css file
* <script src="app.js"></script>
  + Refer to an external Javascript file with the “src” attribute in the <script> tag
* <body onload="initialize()">
  + Execute a JavaScript function “initialize” immediately after a page has been loaded
* <table>
  + A table is created with <table> tag for displaying contact information that extracted from a specific JSON file
* <caption>Information<div id="status"></div></caption>
  + Table caption is declared with <caption> tag
* <thead>
  + The <thead> tag is used to group header content in the table
* <tbody id="tcontent">
  + The <tbody> tag is used to group the body content in the table
  + The content will be generated by using Javascript function dynamically

Experiment 2. Edit the CSS file

* Create and edit the “style.css” file as follows

|  |
| --- |
| table {  width: 100%;  table-layout: fixed;  }  table caption {  font-size: 2em;  }  table td,  table th {  border: 1px solid #000000;  text-align: center;  }  table th {  text-transform: uppercase;  }  @media screen and (max-width: 600px) {  table caption {  font-size: 1.5em;  }  /\* Hide table header. \*/  table thead {  display:none  }  /\* Force table to not be like tables. \*/  table tr {  display: block;  margin-bottom: 1em;  }  /\* Behave like a "row" \*/  table td {  display: block;  text-align: right;  border-bottom: 0px;  }  /\* Behave like a table header \*/  table td::before {  content: attr(data-label);  float: left;  font-weight: bold;  text-transform: uppercase;  }  table td:last-child {  border-bottom: 0;  border: 1px solid #000000;  }  tr:nth-child(even) {  background-color: #dddddd;  }  } |

* Different styles are added to the page with this file
* @media screen and (max-width: 600px)
  + Define and apply styles when the screen width is less than or equal to 600px
  + Change the caption font size from 2em to 1.5em
  + Hide the table header
  + Change the format of table cell to block container
  + Before the data content, add data from table with attribute equals to “data-label” and aligns it to the left
  + Specify the background color to grey for every table row whose numeric position is even

Experiment 3. Edit the Javascript file

* Create and edit the “app.js” file as follows

|  |
| --- |
| function initialize() {  var status = "\* Offline \*";  if (navigator.onLine) {  status = "\* Online \*";  retrieveContacts();  } else {  const localStorage = window.localStorage;  if (localStorage) {  const contacts = localStorage.getItem("contacts");  if (contacts) {  displayContacts(JSON.parse(contacts));  }  }  }  document.getElementById("status").innerHTML = status;  document.body.addEventListener(  "online",  function () {  document.getElementById("status").innerHTML = "Online";  },  false  );  document.body.addEventListener(  "offline",  function () {  document.getElementById("status").innerHTML = "Offline";  },  false  );  }  function retrieveContacts() {  const xhr = new XMLHttpRequest();  const url = "contacts.json";  xhr.onreadystatechange = function () {  if (xhr.readyState === 4) {  var contacts = JSON.parse(xhr.response).contacts;  displayContacts(contacts);  // Store contact data to localstorage  const localStorage = window.localStorage;  if (localStorage) {  localStorage.setItem("contacts", JSON.stringify(contacts));  }  }  };  xhr.open("get", url);  xhr.send();  }  function displayContacts(contacts) {  contacts.forEach(addRow);  }  function addRow(contact) {  var tcontent = document.getElementById("tcontent");  var row = tcontent.insertRow();  var nameCell = row.insertCell();  nameCell.setAttribute('data-label', "Name");  nameCell.innerHTML = contact.name;  var addressCell = row.insertCell();  addressCell.setAttribute('data-label', "Address");  addressCell.innerHTML = contact.address;  var emailCell = row.insertCell();  emailCell.setAttribute('data-label', "Email");  emailCell.innerHTML = contact.email;  } |

* if (navigator.onLine) { ... }
  + Determine if the browser is in online or offline mode
  + If so
    - Get the JSON file and extract contact information
    - Display the contact in HTML table
    - Save contact to the “localstorage”
  + Otherwise
    - Obtain the contact information from “localstorage”
    - Display the contact in HTML table
* document.body.addEventListener( ... )
  + Define the event handler whenever the browser switches between online and offline mode
* if (xhr.readyState === 4) { ... }
  + Evaluate if the file retrieval has been completed
* displayContacts(JSON.parse(xhr.response).contacts);
  + Method “displayContacts” with parameter “JSON.parse(xhr.response).contacts“ is called if the JSON file is ready
* JSON.parse(xhr.response).contacts
  + The JSON.parse() method helps to parse a JSON string which is return from “xhr.response”
  + Then, the “contacts” element is obtained
* contacts.forEach(addRow);
  + for each contact record, call the method “addRow” to add a contact to a table row of the table
* var row = tcontent.insertRow();
  + Add a table row to element in HTML file with “id” attribute equal to “tcontent”
  + Three cells are insert to the table row with variable name “nameCell”, “addressCell” and “emailCell”
  + Specified the three cells’ attribute “data-label” with values “Name”, “Address” and “Email”
  + Assign values to table cells with the corresponding contact details that retrieved from the JSON file

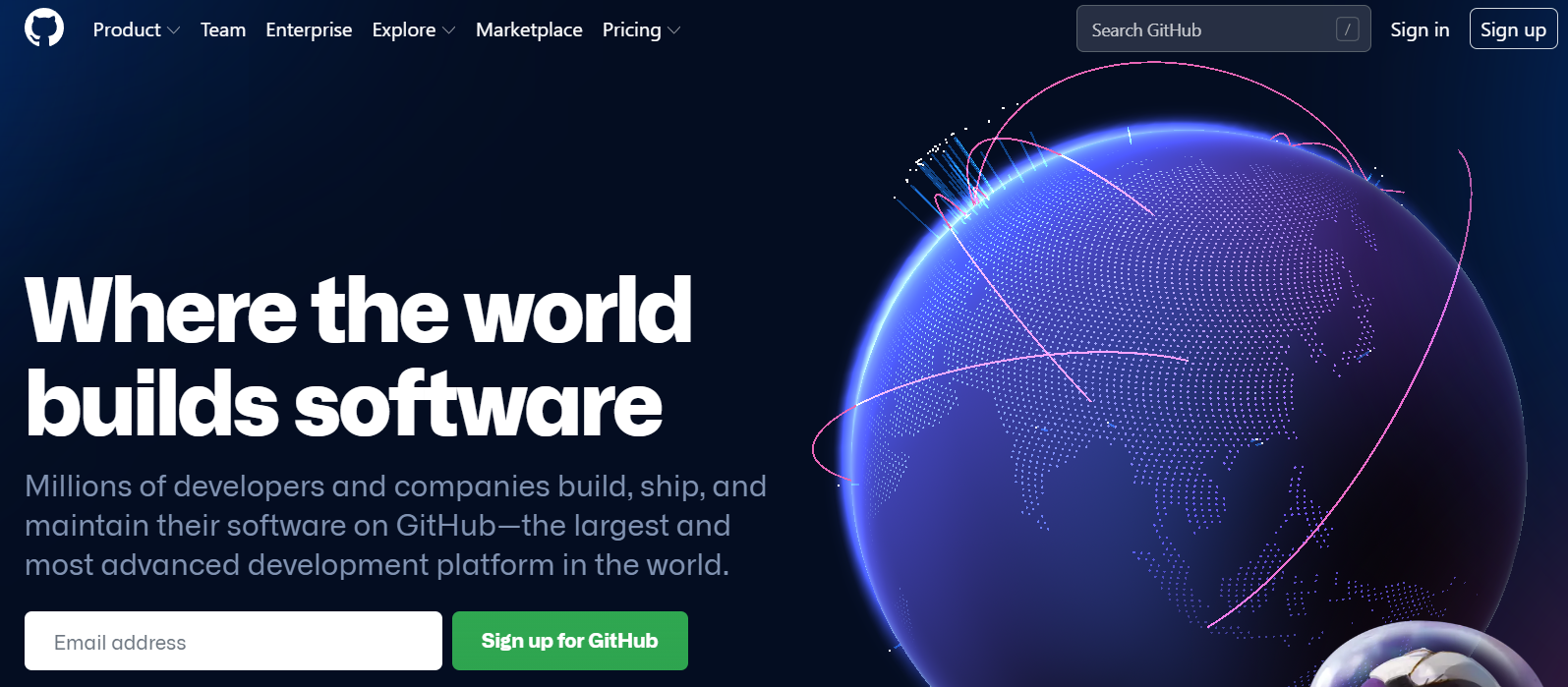
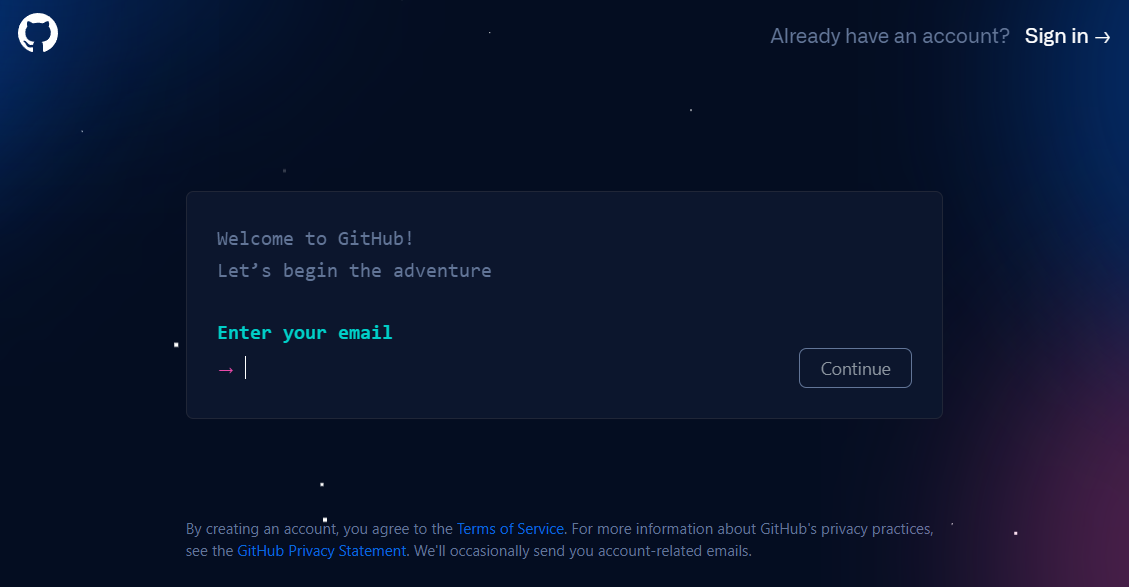
Upload the files to server

* Host the application on a secure HTTPS server
* 1. Locally on your machine
  + Not include in this tutorial
* 2. Provider of Internet hosting for software development and version control
  + E.g., GitHub

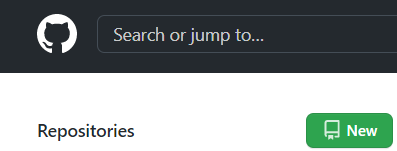
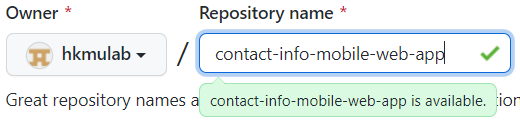
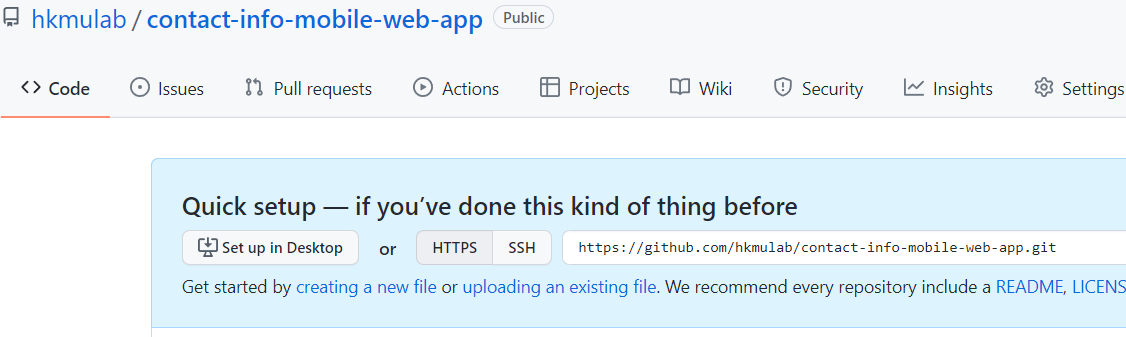
Using GitHub

* 1. Account registration
* 2. Create repository
* 3. Upload files
* 4. Setup GitHub pages

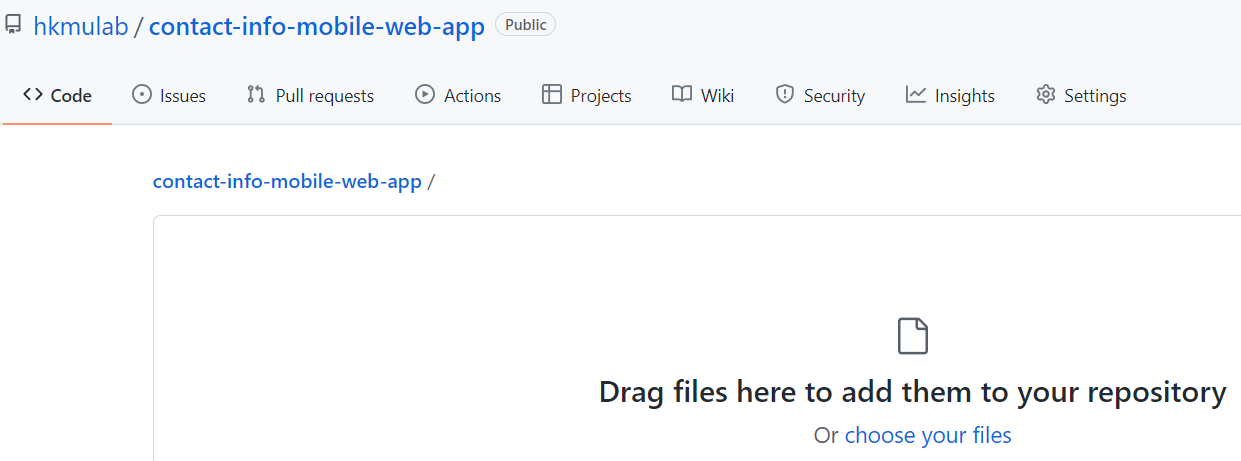
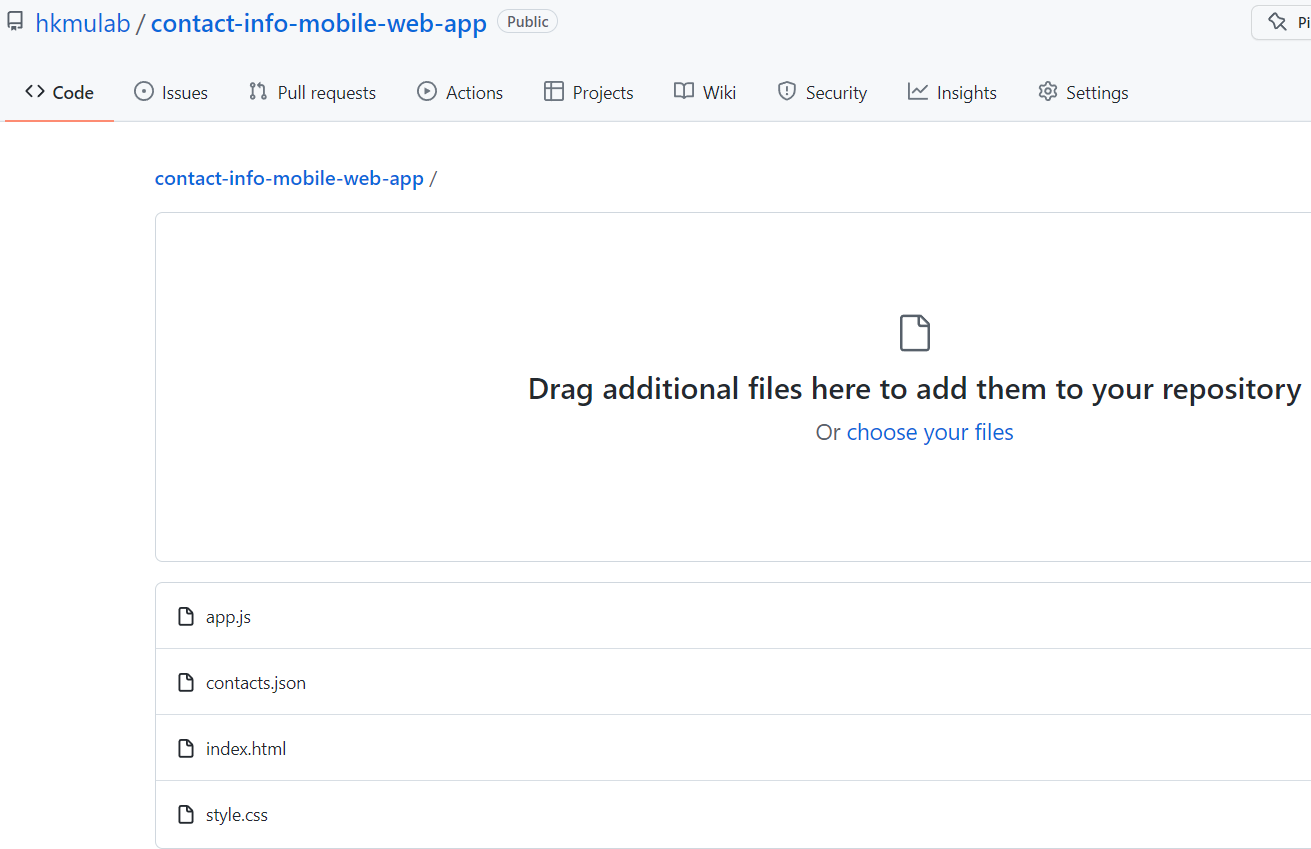
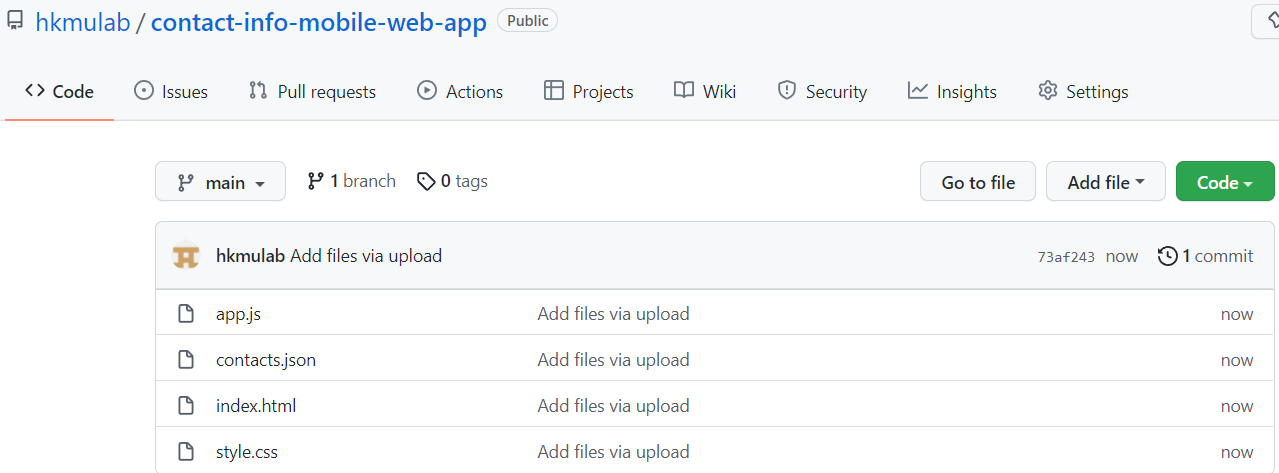
Experiment 1. Account registration

* Access the following URL:
  + https://github.com/
  + 
* Click “Sign Up” button
  + 
* Fill in the information

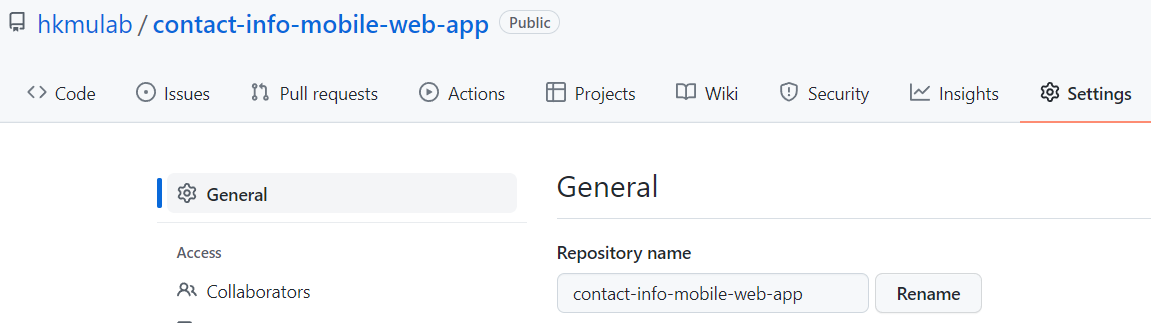
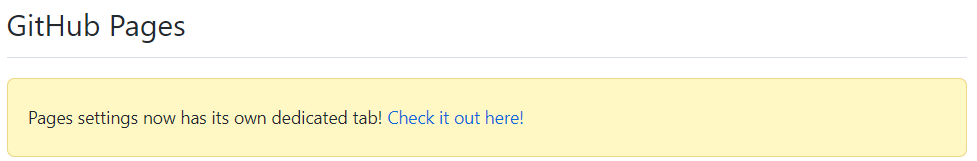
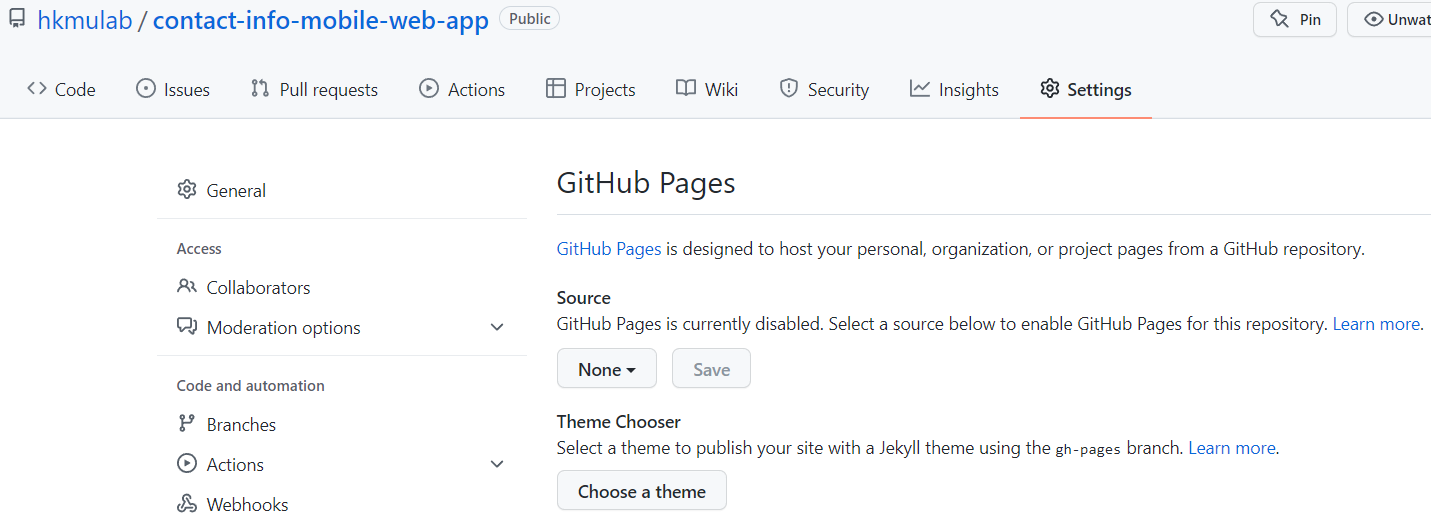
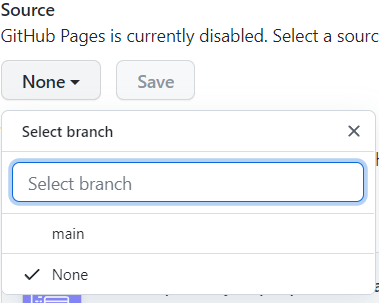
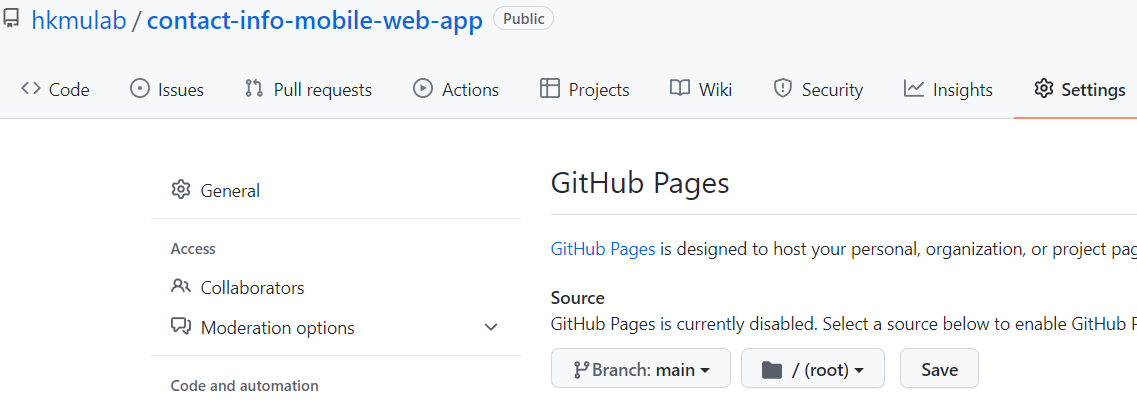
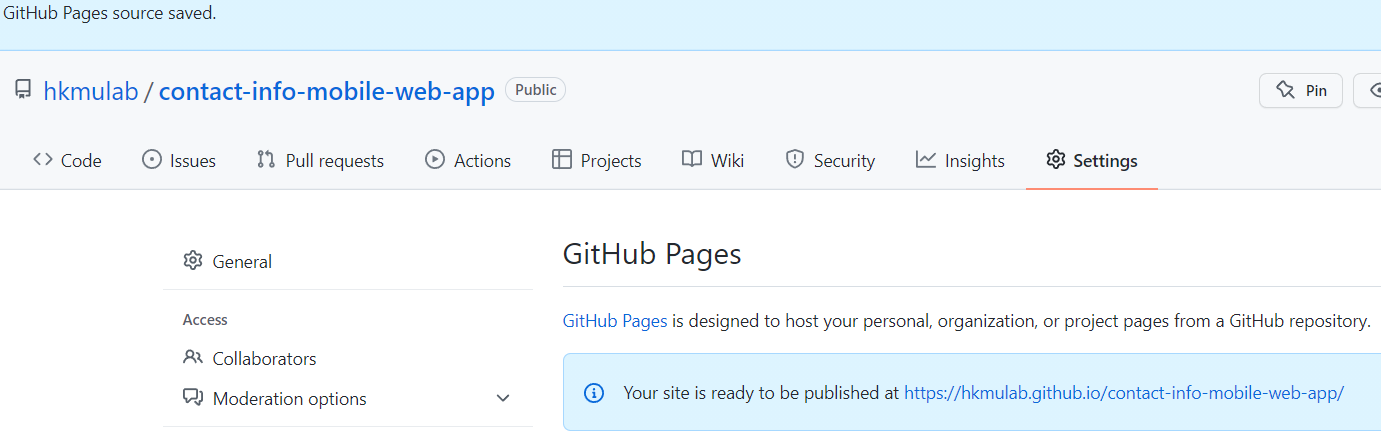
Experiment 2. Create repository

* Login to GitHub
* Click “Create repository” button
  + 
* If it is not the first repository, click “New” button
  + 
* Enter “Repository name”
  + E.g., contact-info-mobile-web-app
  + 
* Click “Create repository” button
  + 
* 

Experiment 3. Upload files

* Select “uploading an existing file”
  + 
* Drag or choose the four files to add them to the repository
  + 
* Click “Commit changes” button
  + 

Experiment 4. Setup GitHub pages

* Click the “Settings” tab
  + 
* Scroll down and reach the “GitHub Pages”
  + 
  + Click the link “Check it out here!”
  + 
* Click “None”
  + 
* Selection option “main”
  + 
* Click “Save”
  + 
* You application is already published with the provide URL

Test the application on mobile device

* User browser of mobile device to access the provide URL for testing the web application
* E.g.,
  + https://<GitHubUsername>.github.io/contact-info-mobile-web-app/