

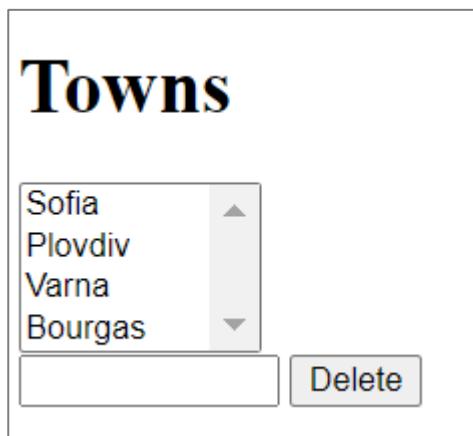
Exercises: Git Branching and Pull Requests

Problems for exercises for the "[Software Engineering and DevOps](#)" course @ SoftUni.

Note that for this exercise you should have a GitHub account.

I. The "Towns" Project

The "Towns" project provides a simple HTML and JavaScript based Web front-end interface to **view, add, delete** and **shuffle a list of towns**. The project is **unfinished**, so some of the **functionality is already implemented** (view and delete) and **other functionality is to be implemented** (add and shuffle towns + CSS styling). This is how the "Towns" project looks like at the beginning:



At the start, in the GitHub repo you will have **3 source code files**: **towns.html + towns.css + towns.js**, and a **README.md** documentation file:

	README.md	4 hours ago
	towns.css	1 minute ago
	towns.html	1 minute ago
	towns.js	1 minute ago

1. Roles Assignments

Your task is to **work in teams of 3 students or work alone with several roles to simulate multi-user interaction**, where each role follows the instructions for certain team member – **Editor, Shuffler and Styler**.

If you work in a **team of 3 students**, one of you should also take the role of the **Team Leader**.

If you work **alone**, you should work with a **fourth** role – the **Team Leader**.

Editor

The Editor should implement "**add new town**" functionality:

Towns

Sofia
Plovdiv
Varna
Burgas

Pleven

Delete Add

Towns

Plovdiv
Varna
Burgas
Pleven

Pleven added.

Delete Add

Shuffler

The **Shuffler** should implement "shuffle towns" functionality.

Towns

Sofia
Plovdiv
Varna
Burgas

Shuffle

Delete

Towns

Varna
Burgas
Sofia
Plovdiv

Shuffle

Towns shuffled.

Delete

Styler

The **Styler** should add some **CSS styling** and **improve** the **HTML UI**.

Towns

Sofia
Varna
Burgas

Delete Existing Town

Delete

Plovdiv deleted.

2. Fork the Repo

The **team leader** does the following:

Fork the "Towns" repo from GitHub: <https://github.com/SUContent/Towns>

3. Invite the Team Members

The **team leader** invites the **other team members** as **collaborators** in the new GitHub repo.

The screenshot shows the GitHub repository settings page. At the top, there are several tabs: Code, Issues, Pull requests, Actions, Projects, Security, Insights, and Settings. The Settings tab is highlighted with a blue border. Below the tabs, there's a section titled "Who has access". Under "Access", the "Collaborators" option is selected and highlighted with a blue box. To the right of this, there are two sections: "PRIVATE REPOSITORY" (with a note that only those with access can view it) and "DIRECT ACCESS" (showing 0 collaborators). A large blue arrow points from the "Collaborators" section down to a "Manage access" section. This section contains a "Manage" button and a note: "You haven't invited any collaborators yet". A green "Add people" button is also present. On the left side of the main content area, there's a sidebar with various repository management options like Branches, Tags, Actions, Webhooks, Codespaces, and Pages.

- An **email invitation** will be sent for each **invited collaborator**.

4. Team Members Should Clone the Project

Each **team member** clones the project from the **team leader's GitHub repository**:

```
git clone https://github.com/<team-leader-username>/Towns
```

5. Edit the Project Description

This step should be executed only **after each team member have already cloned the project locally**.

The **team leader** makes **changes** in **README.md** file from the GitHub's project Web site, to describe with a text which team member which role will take, e.g.,

Roles

- {Name1} takes the role "Editor"
- {Name2} takes the role "Shuffler"
- {Name3} takes the role "Styler"

Commit changes

Added roles

Add an optional extended description...

Commit directly to the `main` branch.

Create a new branch for this commit and start a pull request. [Learn more about pull requests.](#)

Commit changes **Cancel**

6. Implement Project Functionalities

Each **team member** implements different **functionality locally** (each **member** has its **own instructions**).

- **Member #1:** Editor's functionality ([add town](#))
- **Member #2:** Shuffler's functionality ([shuffle towns](#))
- **Member #3:** Styler's functionality ([improve styling and CSS](#))

NOTE: If you are working alone, don't forget that you should fulfil the three roles **simultaneously**.

Editor

The **Editor** should already have cloned the forked repo.

Step 1: New Town Textbox + Button

Now, in `towns.html`, the editor should add a textbox + button for creating a new town:

```
<div>
  <input type="text" id="townNameForAdd" />
  <button id="btnAdd">Add</button>
</div>
```

Step 2: New Town JavaScript Code

In `towns.js`, the editor has to add a new function for adding a new town:

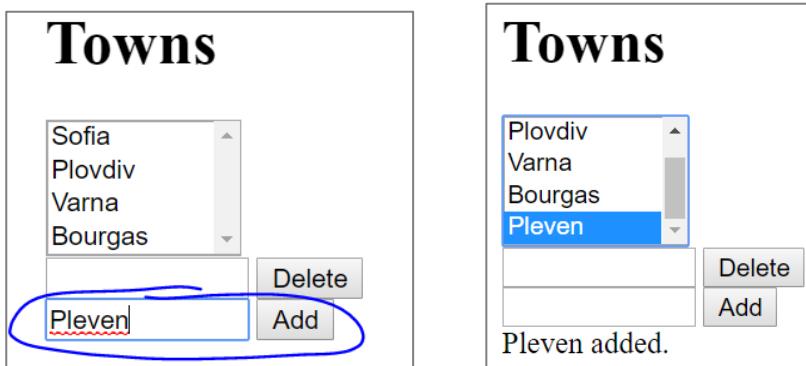
```
function addTown() {
  let townName = $('#townNameForAdd').val();
  $('#townNameForAdd').val('');
  $('#towns').append($('').text(townName));
  $('#result').text(townName + " added.");
}
```

Also, the editor should add a code to **attach an event handler** to invoke the new function when the **[Add]** button is pressed. In the start of the JS code, he adds the code marked in **blue** below:

```
$(document).ready(function() {
  ...
  $('#btnAdd').click(addTown);
});
```

Step 3: Test the Project functionality

Editor now should **test the project functionality** to see whether the styling and effects work correctly, as well as whether the entire project works as expected:



Step 4: Commit All Changes in the Local Repo

Editor adds and **commits** in Git all local changes:

```
git commit -a -m "Implemented functionality to add a new town"
```

Step 5: Push the Local Commits to GitHub

Editor pushes all the changes to Git:

```
git push
```

Step 6: Resolve Any Conflicts

```
git pull
```

Editor should edit all files and fixes the code in order to **merge** all concurrent **changes correctly**.

Then, Editor adds and **commits** in Git the **merged** files:

```
git commit -a -m "Implemented functionality to add a new town + merged the conflicting files"
```

Finally, Editor should **push all his changes** again to Git:

```
git push
```

Shuffler

The **Shuffler** should already have cloned the forked repo.

Step 1: New Town Textbox + Button

Now, in **towns.html**, the editor should add a textbox + button for shuffling the towns:

```
<div>
  <button id="btnShuffle">Shuffle</button>
</div>
```

Step 2: New Town JavaScript Code

In **towns.js**, the editor has to add a new function for shuffling the towns:

```
function shuffleTowns() {
  let towns = $('#towns option').toArray();
  $('#towns').empty();
```

```

shuffleArray(towns);
$('#towns').append(towns);
$('#result').text("Towns shuffled.");

function shuffleArray(array) {
    for (var i = array.length - 1; i > 0; i--) {
        var j = Math.floor(Math.random() * (i + 1));
        var oldElement = array[i];
        array[i] = array[j];
        array[j] = oldElement;
    }
}

```

Also, the editor should add a code to **attach an event handler** to invoke the new function when the **[Add]** button is pressed. In the start of the JS code, he adds the code marked in **blue** below:

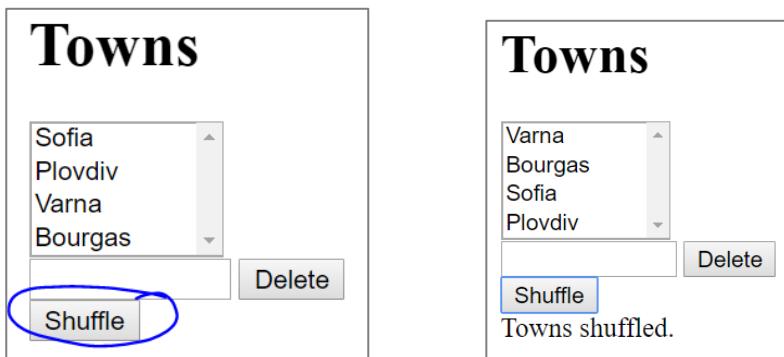
```

$(document).ready(function() {
    $('#btnShuffle').click(shuffleTowns);
});

```

Step 3: Test the Project functionality

Shuffler now should **test the project functionality** to see whether the styling and effects work correctly, as well as whether the entire project works as expected:



Step 4: Commit All Changes in the Local Repo

Shuffler adds and **commits** in Git all local changes:

```
git commit -a -m "Implemented functionality to shuffle the existing towns"
```

Step 5: Push the Local Commits to GitHub

Shuffler pushes all the changes to Git:

```
git push
```

Step 6: Resolve Any Conflicts

```
git pull
```

Editor should edit all files and fixes the code in order to **merge** all concurrent **changes correctly**.

Then, **Shuffler** adds and **commits** in Git the **merged** files:

```
git commit -a -m "Implemented functionality to shuffle the existing towns + merged the conflicting files"
```

Finally, **Shuffler** should push all his changes again to Git:

```
git push
```

Styler

The **Styler** should already have cloned the forked repo.

Step 1: Improve the HTML Structure

Styler wants to style the site to look better but the **HTML structure** does not allow writing CSS, so they modify **towns.html** and introduces a new way to structure the content as a sequence of **articles** holding **headers** and other elements after the header:

```
<article>
    <header>Towns</header>
    <select id="towns" size="4">
        <option>Sofia</option>
        <option>Plovdiv</option>
        <option>Varna</option>
        <option>Burgas</option>
    </select>
</article>

<article>
    <header>Delete Existing Town</header>
    <input type="text" id="townName" />
    <button id="btnDelete">Delete</button>
</article>
```

Step 2: Write the CSS Code

Styler now rewrites the entire **towns.css** file from scratch:

```
@import url('https://fonts.googleapis.com/css?family=Rubik');

body {
    font-family: 'Rubik', sans-serif;
}

* {
    box-sizing: content-box;
}

article {
    background: #CCC;
    width: 180px;
    padding: 10px;
    margin: 10px;
    display: inline-block;
    vertical-align: top;
}

article>header {
    background: #5F5F5F;
    color: white;
    margin: 0px 0px 10px 0px;
    padding: 4px 6px;
}
```

```

article>header>h1 {
    margin: 0px;
}

article>select {
    width: 178px;
}

article>input {
    width: 176px;
}

article>button {
    display: block;
    margin: 10px auto 0px auto;
    border: none;
    border-radius: 3px;
    padding: 5px 15px;
    background: green;
    color: white;
    font-weight: bold;
}

article>button:hover {
    box-shadow: 0px 0px 10px white;
    cursor: pointer;
}

button#btnDelete {
    background: red;
}

#result {
    display: none;
    width: 50%;
    margin: 10px auto;
    padding: 10px 15px;
    background: #DDD;
    border-radius: 5px;
    border: 1px solid #777;
}

```

The new CSS code assumes the HTML uses **articles** with **headers** for its major areas. It displays the articles in a nice-looking way. The CSS also hides by default the result info box and assumes it will be shown by the JS code later.

Step 3: Add "Auto Hide" Effect for the Info Messages

After a button is clicked (e.g. [Delete]), the result of the performed action is shown into an info box (**#result**). Styler modifies this behavior, so that the info box is by default hidden, then it displays a message for 3 seconds, then it disappears. First, he adds a JS library in **towns.html** to enable animation effects with jQuery:

```
<script src="http://ajax.googleapis.com/ajax/libs/jqueryui/1.8.5/jquery-
ui.min.js"></script>
```

Next, he modifies the **deleteTown()** function in **towns.js** to display messages through a new function **showMessage(msg)**:

```

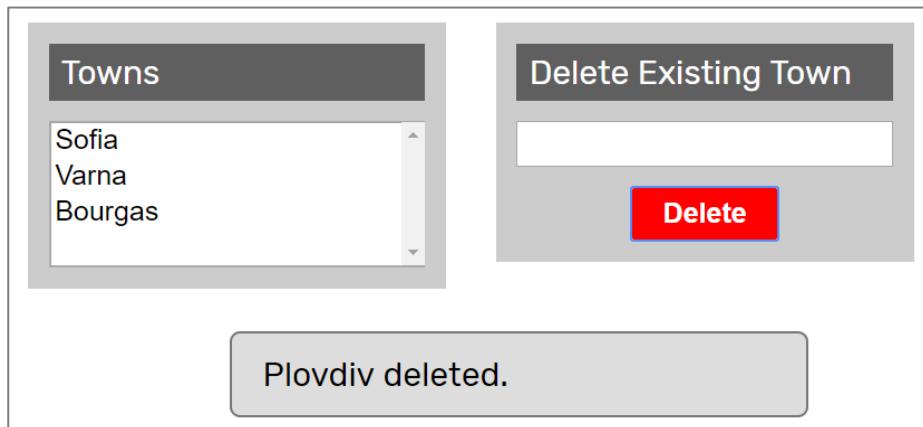
function deleteTown() {
    let townName = $('#townName').val();
    $('#townName').val('');
    let removed = false;
    for (let option of $('#towns option')) {
        if (option.textContent == townName) {
            removed = true;
            option.remove();
        }
    }
    if (removed)
        showMessage(townName + " deleted.");
    else
        showMessage(townName + " not found.");
}

function showMessage(msg) {
    $('#result').text(msg).css("display", "block");
    setTimeout(function () {
        $('#result').hide('blind', {}, 500);
    }, 3000);
}

```

Step 4: Test the Project Functionality

Styler now tests the project functionality to see whether the styling and effects work correctly, as well as whether the entire project works as expected:



Step 5: Commit All Changes in the Local Repo

Styler adds and commits in Git all local changes:

```
git commit -a -m "Implemented functionality to add a new town"
```

Step 6: Push the Local Commits to GitHub

Styler pushes all his changes to Git:

```
git push
```

Step 7: Resolve Any Conflicts

In case of conflict **Styler** pulls, merges, then pushes again:

```
git pull
```

Styler edits all files and fixes the code in order to merge all concurrent changes correctly.

Then, **Styler adds and commits in Git** the **merged files**:

```
git commit -a -m "Improved the UI: added CSS styles + HTML structure + JS effects"
```

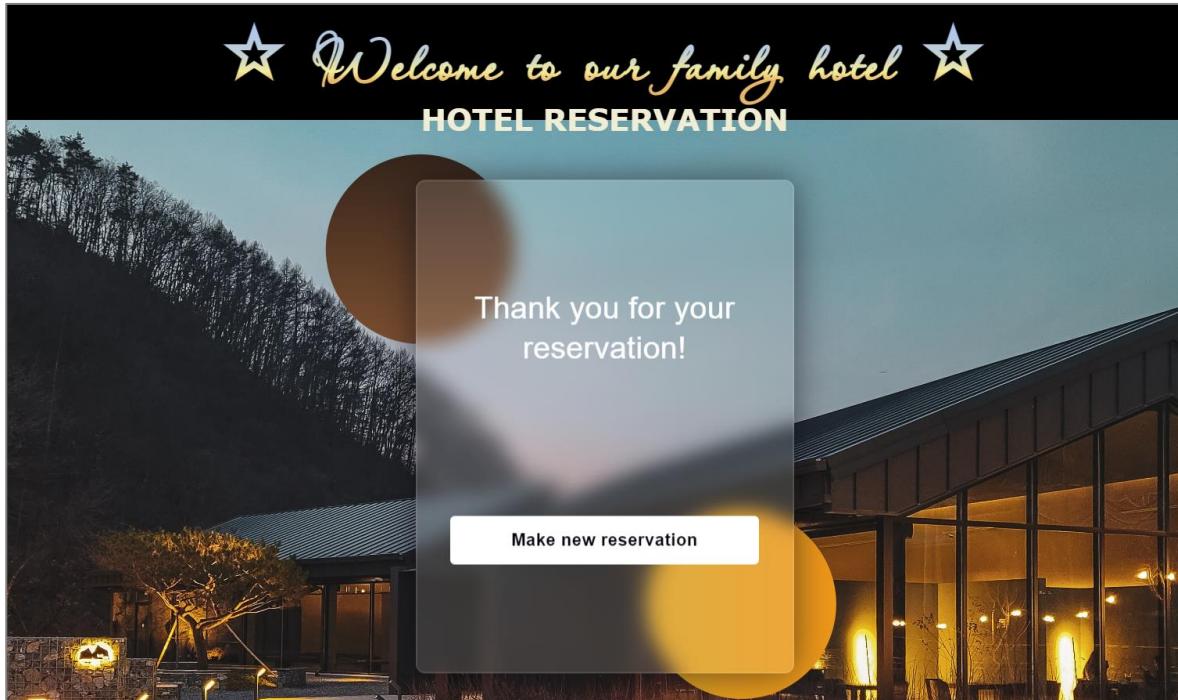
Finally, **Styler pushes all his changes** again to Git:

```
git push
```

II. The "Hotel Reservation" Project

1. Create and Clone Repo

The project "Hotel Reservation" provides a simple HTML and JavaScript based Web front-end interface to **create a reservation to family hotel**. The project is **unfinished**, so some of the **functionalities are already implemented** ("Thank you" page) and **other functionalities are to be implemented** (Search Form, Search Result Form, Guest Details Form, Confirm Page). This is how the "Hotel Reservation" project looks like at the beginning:



In the beginning you should create an **empty GitHub repo** and add the files from resources (**index.html**, **solution.js**, **static folder**).

2. Team Assignment – Overview

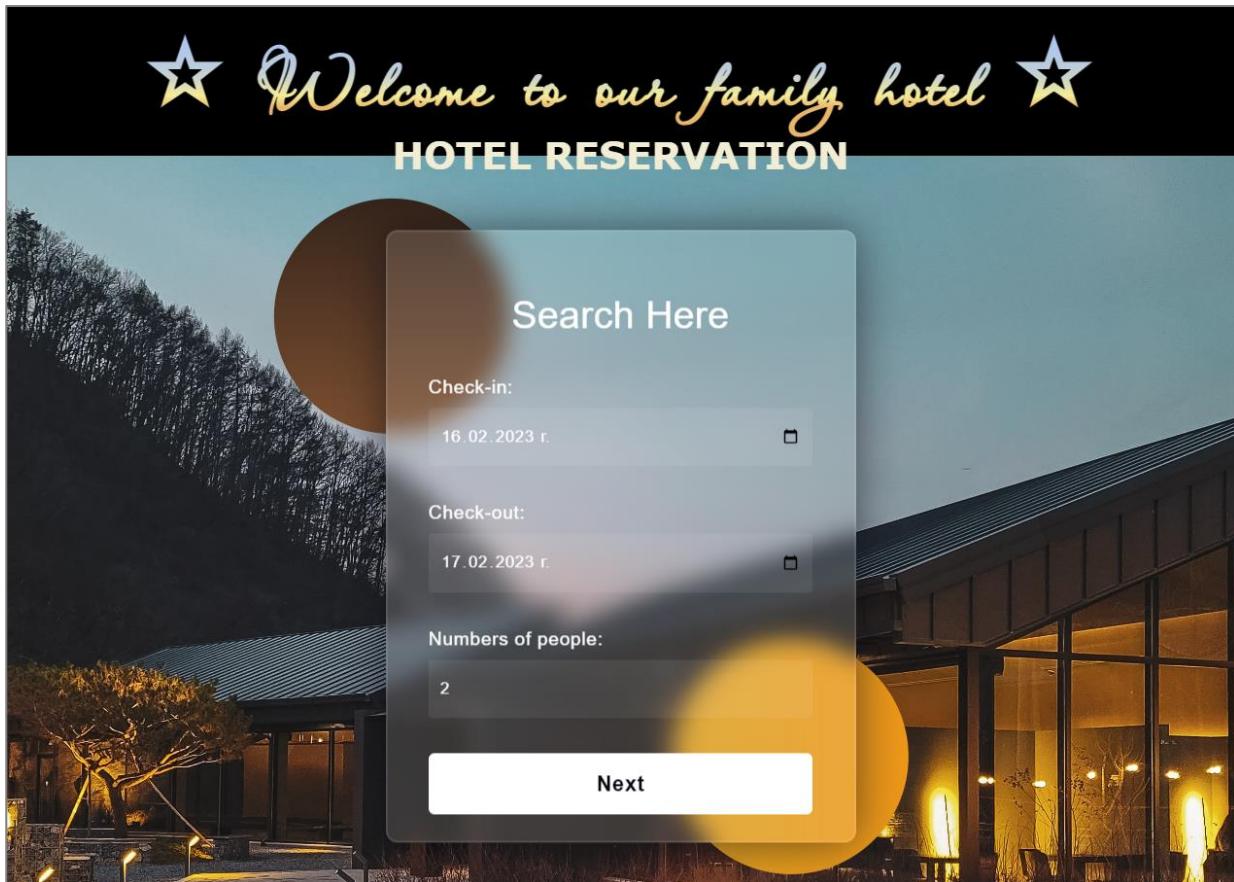
NOTE: You can work in teams of 4 students or work alone with several roles to simulate multi-user interaction, where each role follows the provided instructions for the given team member.

Each team member **chooses a role**:

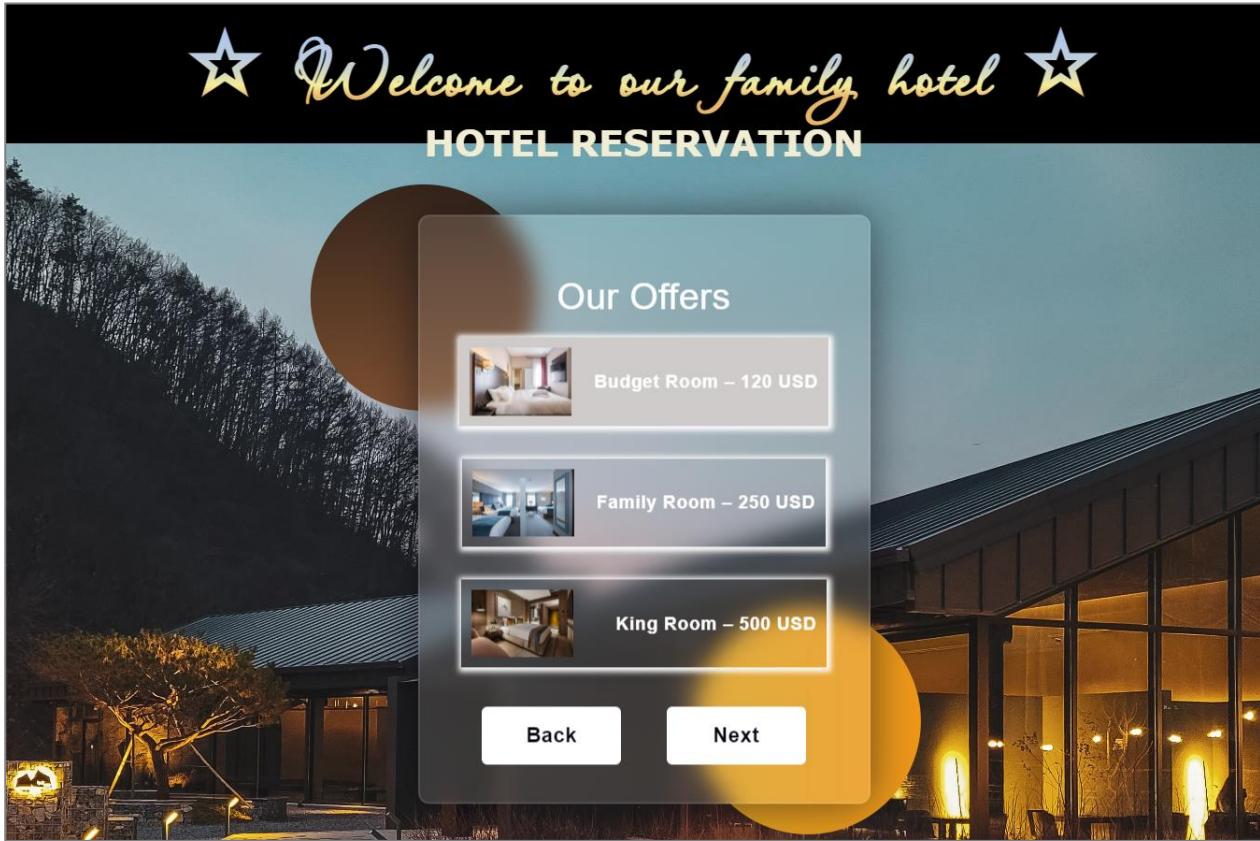
- **Member #1:** takes the role of **Questioner**.
- **Member #2:** takes the role of **Offerer**.
- **Member #3:** takes the role of **Admin**.
- **Member #4:** takes the role of **Verifier**.

One of the team members takes an additional role: **team leader**.

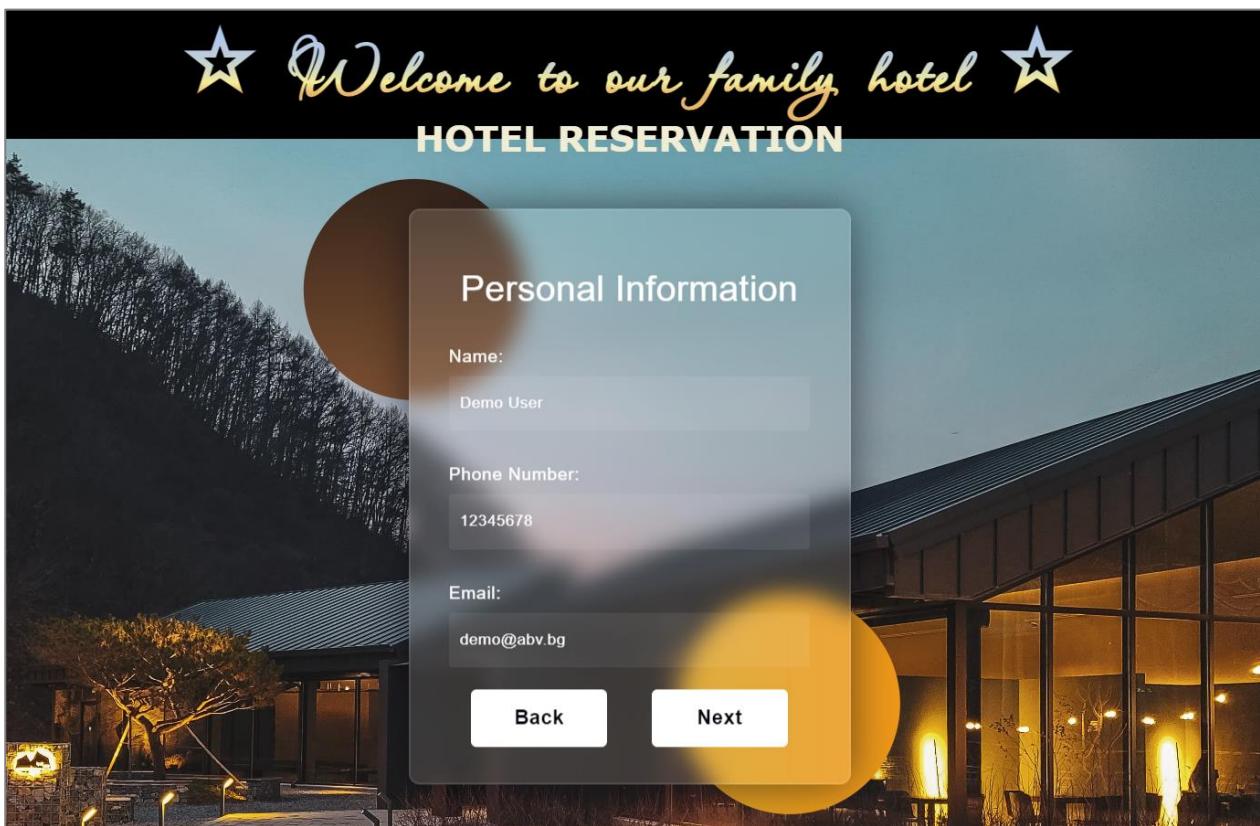
Questioner should implement the "Search Form" functionality. This feature will allow the inquirer to **provide** and **manage** offers through a **special form** for **date** and **number of people**. The provider should include this **functionality** in their **application** to **improve** the **user experience**.



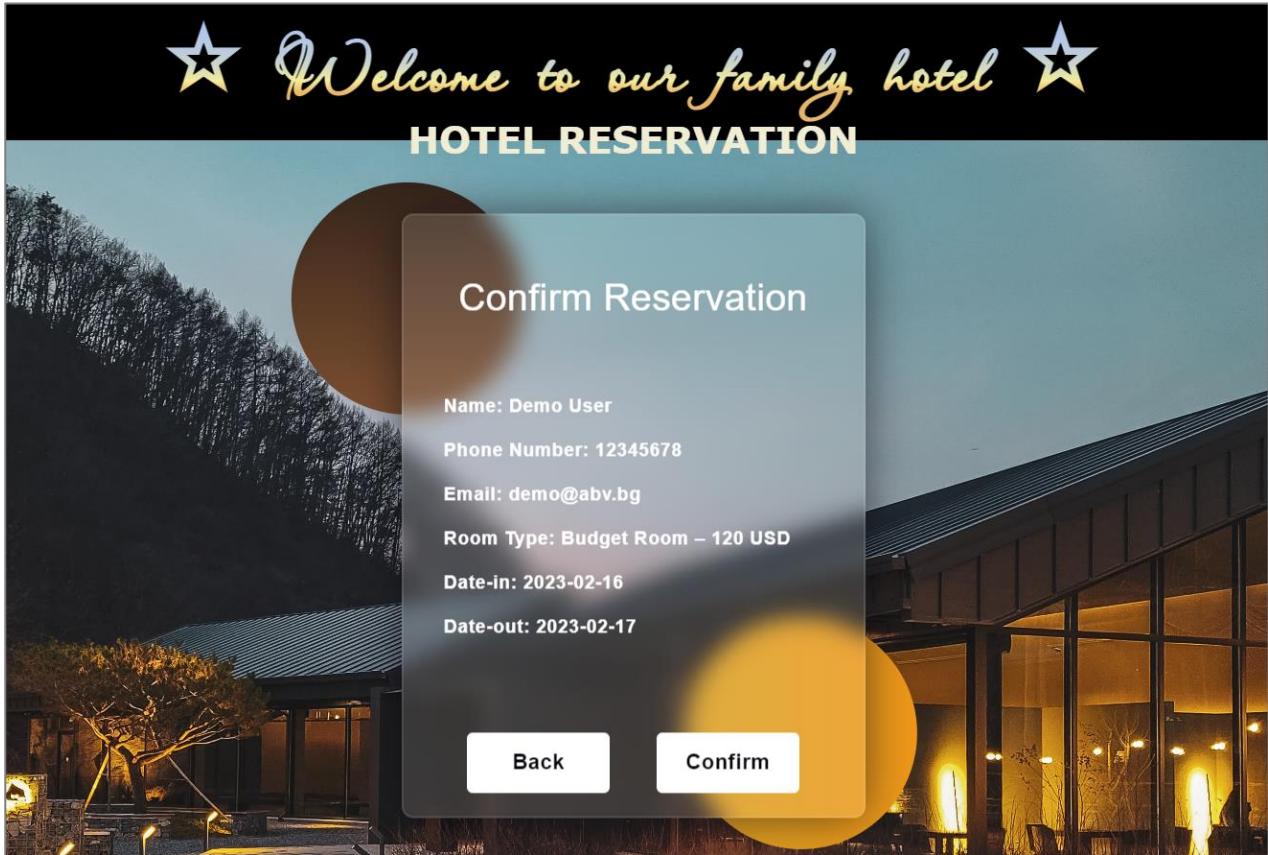
Offerer should implement the "Our Offers Form" functionality. This feature will enable the Offerer to provide and manage offers through a dedicated form. **Offerer** should **incorporate** this **functionality** into their **application** to **enhance** the **user experience**.



Admin should implement the "**Guest Details Form**" functionality. This feature will allow him to gather and manage guest information through a dedicated form. **Admin** should incorporate this functionality into their application to efficiently collect and handle guest details.



Verifier should implement "**Confirm Reservation Form**" functionality. This feature will enable him to confirm reservations through a dedicated form. **Verifier** should incorporate this functionality into their system to efficiently handle and validate reservation requests.



Step 1: Create the Repo

The **team leader** creates an **empty GitHub repo** and **add the files** from resources (**index.html**, **solution.js**, **static folder**).

Step 2: Invite the Team Members

The **team leader invites** the other team **members** as **collaborators** in the **new GitHub repo**.

The screenshot shows the 'Settings' tab selected in the GitHub repository header. In the sidebar, the 'Collaborators' button is highlighted with a blue box and an arrow pointing to it from the left. The main area displays the 'Who has access' section, which is labeled as a 'PRIVATE REPOSITORY'. It states that only those with access to the repository can view it. Below this, there is a 'DIRECT ACCESS' section showing that 0 collaborators have access to the repository. A 'Manage' link is provided for both sections. In the bottom right corner of the main area, there is a 'Manage access' section with a heading 'You haven't invited any collaborators yet' and a green 'Add people' button.

An **email invitation** will be sent for each **invited collaborator**.

Step 3: Clone the Project

Each **team member** clones the project from the **team leader's GitHub repository** to a local folder:

```
git clone {GitHub repo URL}
```

Step 4: Implement Project Functionalities

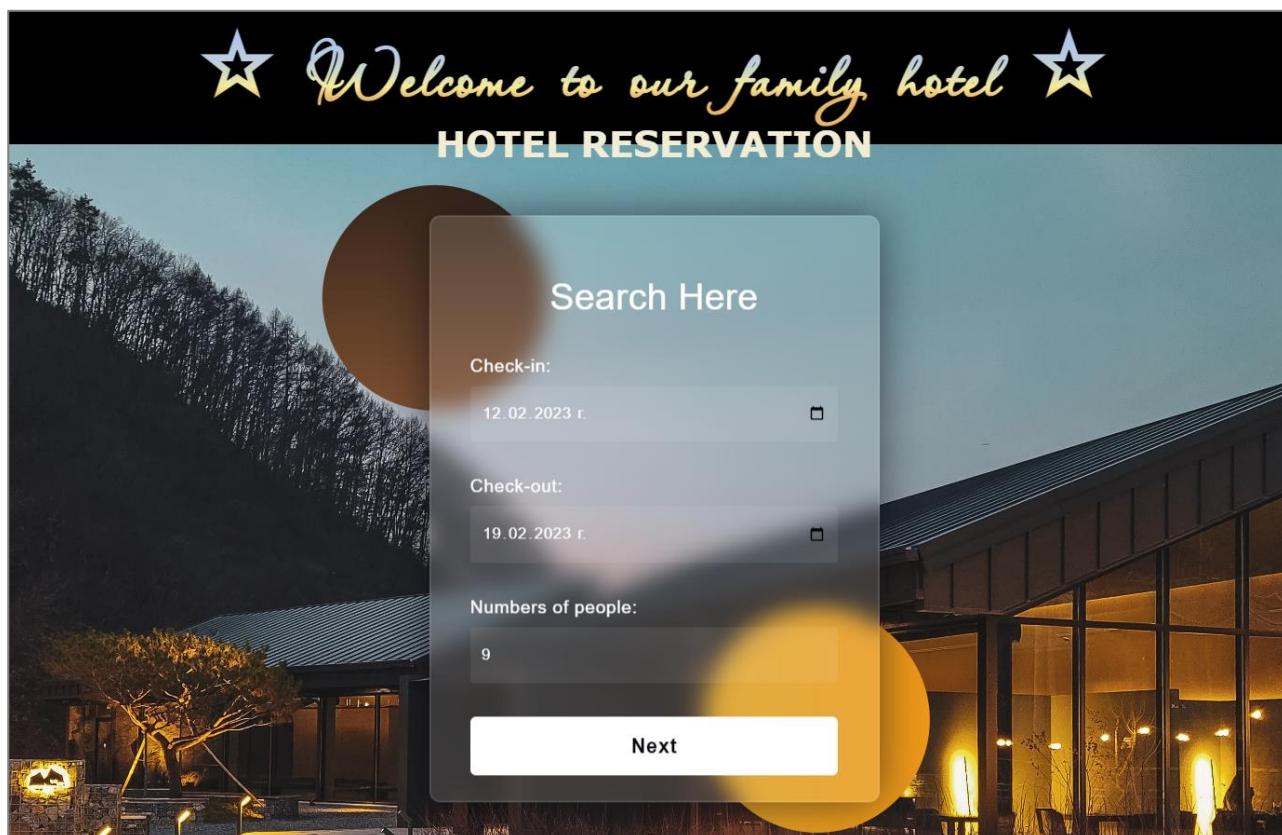
Each **team member** implements different **functionality locally** (each **member** has its **own document** with **instructions** in the provided **resources**).

- **Member #1: Questioner's functionality (Search Form)**
- **Member #2: Offerer's functionality (Our Offers Form)**
- **Member #3: Admin's functionality (Guest Details Form)**
- **Member #3: Verifier's functionality (Confirm Reservation Form)**

You should have a **separate branch in GitHub** for each functionality, e.g., **each member** should have a **branch of their own**.

Questioner

Questioner should implement the "**Search Form**" functionality:



Searched form contains **information** about the **check-in date**, the **check-out date**, and the **numbers of people** accommodated.

Questioner clones the project repository, makes a sequence of **changes** in the source code files, **commits** locally in repository, then **pushes** the committed changes to GitHub.

Step 1: Clone the "Hotel Reservation" Repository

Questioner should already have cloned the repo.

Step 2: Create a Local Branch

We want to **add some new features** to our SPA app. We are working in our **local repository**, and we do not want to disturb or wreck the main project. So, we create a **new local branch**:

```
git branch search-form
```

We have **two branches** now: **main** and **search-form**.

Now we should check out the **new branch**, e.g., switch from the **current branch** to the **new one**.

```
git checkout search-form
```

We have moved our **current workspace** from the **main branch**, to the **search-form branch**.

Step 3: Search Form: HTML

In **index.html**, in **div** with class "**site-content**" they add some **HTML** (**Questioner/index.html** file from the resources) for the **search form**.

Step 4: Search Form: JavaScript Code

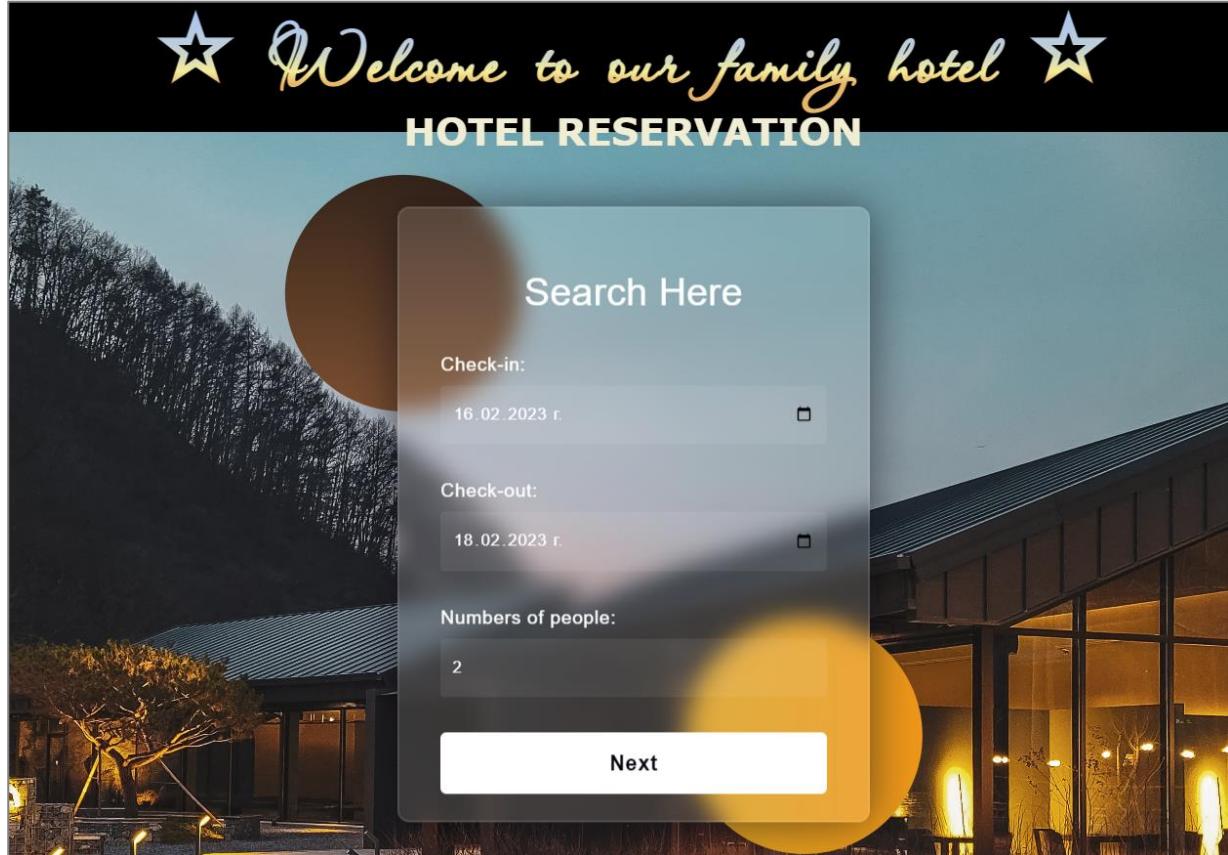
In **solution.js**, they add a new function (**Questioner/solution.js** file from the resources) to the **search form**.

Step 5: Search Form: CSS

In **styles.css**, they add **styles** (**Questioner/styles.css** file from the resources) to the **search form**.

Step 6: Test the Project Functionality

Questioner now **tests the project functionality** to see whether SPA app works correctly, as well as whether the entire project works as expected:



```
solution.js:31
```

```
{startDate: '2023-01-31', endDate: '2023-02-28', guestsCount: '2',  
roomType: null, name: null, ...} ⓘ  
email: null  
endDate: "2023-02-28"  
guestsCount: "2"  
name: null  
phone: null  
roomType: null  
startDate: "2023-01-31"  
► [[Prototype]]: Object
```

```
✖ ► Uncaught TypeError: Cannot read properties of null (reading 'classList')  
at changeContent (solution.js:14:44)  
at searchFormData (solution.js:32:9)  
at HTMLButtonElement.<anonymous> (solution.js:18:80)
```

```
>
```

Don't worry about the error!

Step 7: Commit All Changes to the Local Branch

Questioner adds and commits in Git all local changes:

```
git commit -a -m "Implemented Search form functionality"
```

Now we have a **new branch**, that is different from **main**. If we now **push** to the **repo**, we will get an **error** because the **search-form branch** is **local**, **not upstream**, e.g., it is **not in GitHub** yet, only **locally**:

```
git push
```

Questioner / Hotel-Reservation Public

Code Issues Pull requests Actions Projects Wiki ...

Overview Yours Active Stale All New branch

Search branches...

Default branch

main Updated 46 minutes ago by questioner

To fix this **error** we should run the **following command**.

```
git push --set-upstream origin search-form
```

The "**--set-upstream**" option is **utilized** to set the **remote** as the **upstream** directory and **fix** the above-encountered error.

The screenshot shows a GitHub repository interface. At the top, there are navigation links: Code, Issues, Pull requests, Actions, Projects, Wiki, and three dots. Below the navigation is a horizontal bar with tabs: Overview (highlighted in blue), Yours, Active, Stale, All, and New branch (highlighted in green). A search bar below the tabs contains the placeholder "Search branches...". Under the "Default branch" section, there is a card for the "main" branch, which was updated 47 minutes ago by "questioner". In the "Your branches" section, there is a card for the "search-form" branch, which was updated 19 minutes ago by "questioner". In the "Active branches" section, there is another card for the "search-form" branch, also updated 19 minutes ago by "questioner".

We have the **upstream branch search-form**.

Step 8: Create a Pull Request

The screenshot shows a GitHub pull request creation dialog. It starts with a message: "search-form had recent pushes less than a minute ago" and a "Compare & pull request" button (highlighted in blue). Below this, the title "Open a pull request" is displayed. A message says "Create a new pull request by comparing changes across two branches. If you need to, you can also compare across forks." Two dropdown menus are shown: "base: main" and "compare: search-form" (both highlighted in blue). A note indicates "Able to merge. These branches can be automatically merged." The main area for the pull request contains the commit message "Implemented Search form functionality". There are "Write" and "Preview" buttons, and a rich text editor toolbar with various icons. A comment input field says "Leave a comment" and a file attachment field says "Attach files by dragging & dropping, selecting or pasting them.". Finally, a "Create pull request" button (highlighted in blue) is at the bottom right.

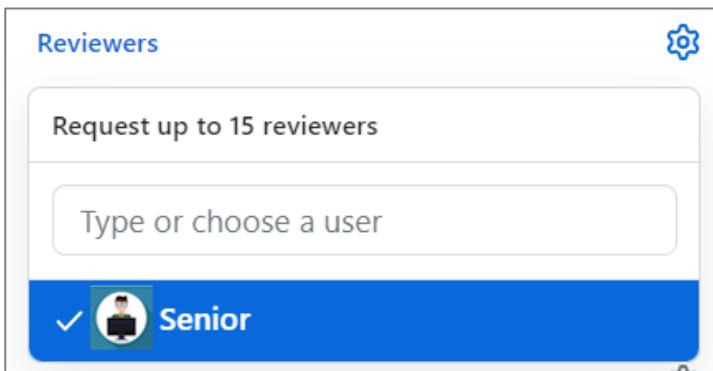
Don't forget to **add a reviewer**.

Reviewers

Request up to 15 reviewers

Type or choose a user

✓ Senior



Add a **title** and **create the pull request**:

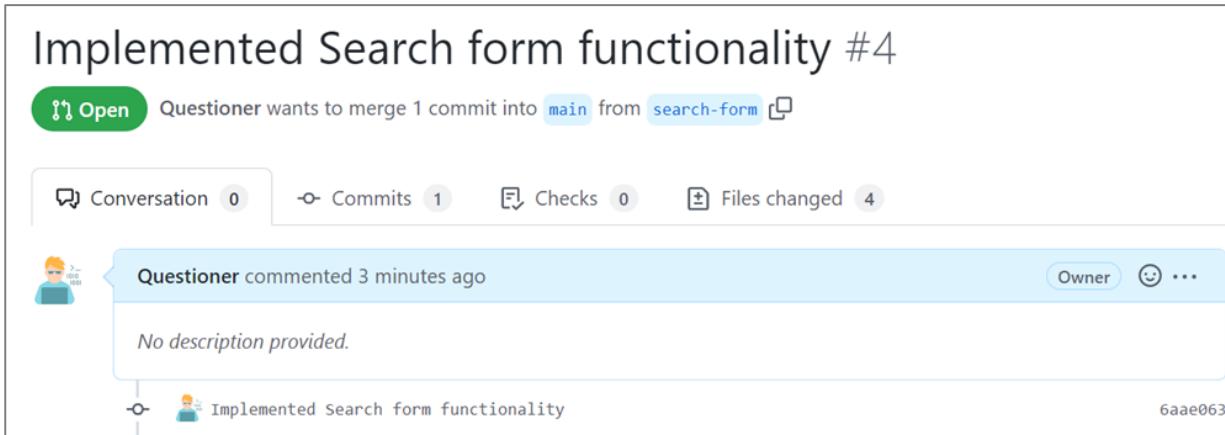
Implemented Search form functionality #4

Open Questioner wants to merge 1 commit into `main` from `search-form` 

Conversation 0 Commits 1 Checks 0 Files changed 4

Questioner commented 3 minutes ago  ...
No description provided.

Implemented Search form functionality 6aae063



Step 9: Approve a Pull Request

Now you should **approve** one of your friend's **pull requests**. But before that, we should **resolve the conflicts**.

Code Issues **Pull requests 1** Actions Projects Wiki Security ...

Label issues and pull requests for new contributors 

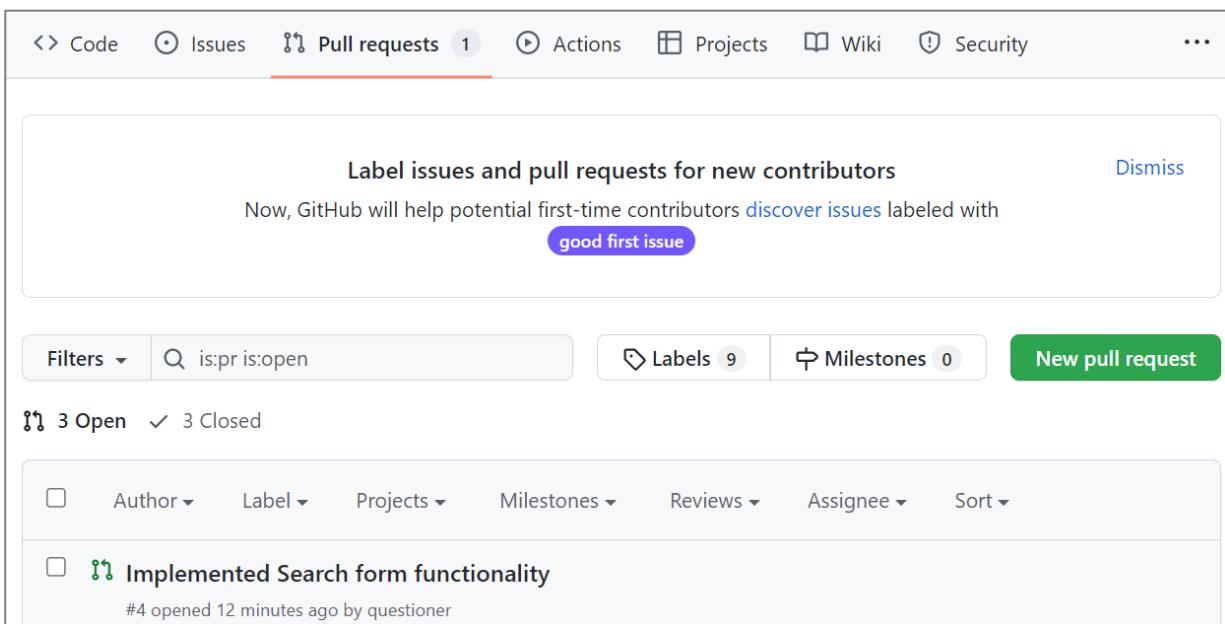
Now, GitHub will help potential first-time contributors [discover issues](#) labeled with 

Filters  Labels 9 Milestones 0 New pull request

3 Open 3 Closed

Author Label Projects Milestones Reviews Assignee Sort

Implemented Search form functionality
#4 opened 12 minutes ago by questioner



This branch has conflicts that must be resolved

Resolve conflicts

Conflicting files

index.html

Merge pull request

You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

Implemented Search form functionality #4

Resolving conflicts between `search-form` and `main` and committing changes → `search-form`

1 conflicting file	index.html	1 conflict	Prev ⌂	Next ⌄		Mark as resolved
index.html index.html	<pre> 1 <!DOCTYPE html> 2 <html lang="en"> 3 4 <head> 5 <meta charset="UTF-8"> 6 <meta http-equiv="X-UA-Compatible" content="IE=edge"> 7 <link rel="stylesheet" href="./static/css/styles.css"> 8 <title>Hotel reservation</title> 9 </head> 10 <body> 11 <section id="welcome"> 12 <h1>Welcome to our family hotel </h1> 13 <div class="home-container"> 14 <div class="info"> 15 <h1>Hotel reservation</h1> </pre>					

When **conflicts are resolved**, merge the changes:

Implemented Search form functionality #4

Resolving conflicts between `search-form` and `main` and committing changes → `search-form`

1 conflicting file	index.html	✓ Resolved
index.html index.html	<pre> 27 28 29 30 31 32 33 34 35 36 37 38 39 </div> <form> <h3>Search Here</h3> <label for="check-in">Check-in:</label> <input type="date" id="check-in" name="check-in"> <label for="check-out">Check-out:</label> <input type="date" id="check-out" name="check-out"> <label for="people">Numbers of people:</label> <input type="number" min="1" id="people" name="phone-number"> </pre>	✓ Resolved

Step 10: Merge Pull Request

Merge the `search-form` branch into the `main` branch.

 Continuous integration has not been set up
GitHub Actions and several other apps can be used to automatically catch bugs and enforce style.

 This branch has no conflicts with the base branch
Merging can be performed automatically.

Merge pull request ▾
You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

Merge pull request #4 from questioner/search-form

Implemented Search form functionality

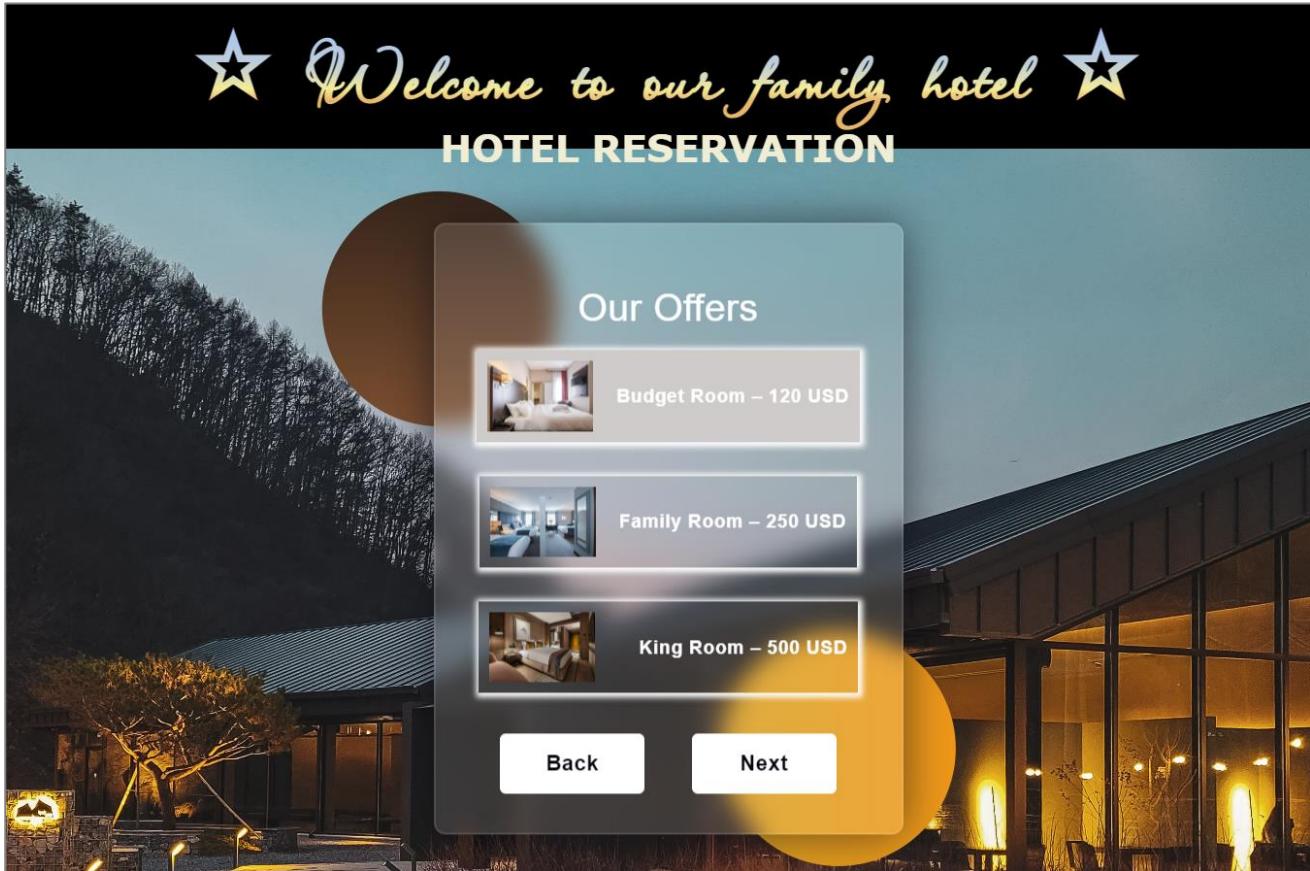
This commit will be authored by questioner

Confirm merge **Cancel**

 Pull request successfully merged and closed **Delete branch**
You're all set—the `search-form` branch can be safely deleted.

Offerer

Offerer should implement the "**Our Offers Form**" functionality:



The **purpose of Offerer** is to find out what kind of **room** our **client prefers**.

Offerer **clones** the project repository, makes a sequence of **changes** in the source code files, **commits** locally in his repository, then **pushes** the committed changes to GitHub.

Step 1: Clone the "Hotel Reservation" Repository

Offerer should already have cloned the repo.

Step 2: Create a Local Branch

We want to **add some new features** to our SPA app. We are working in our **local repository**, and we do not want to disturb or wreck the main project. So, we create a **new local branch**:

```
git branch our-offers-form
```

We have **two branches** now: **main** and **our-offers-form**.

Now we should check out the **new branch**, e.g., switch from the **current branch** to the **new one**.

```
git checkout our-offers-form
```

We have moved our **current workspace** from the **main branch**, to the **our-offers-form branch**.

Step 3: Search Form: HTML

In **index.html**, in **div** with **class "site-content"** he adds some **HTML** (**Offerer/index.html** file from the resources) for the offers form.

Step 4: Search Form: JavaScript Code

In **solution.js**, they add a new function (**Offerer/solution.js** file from the resources) to the **offers form**.

Use this **code box** only **while testing functionality** in your **custom branch**.

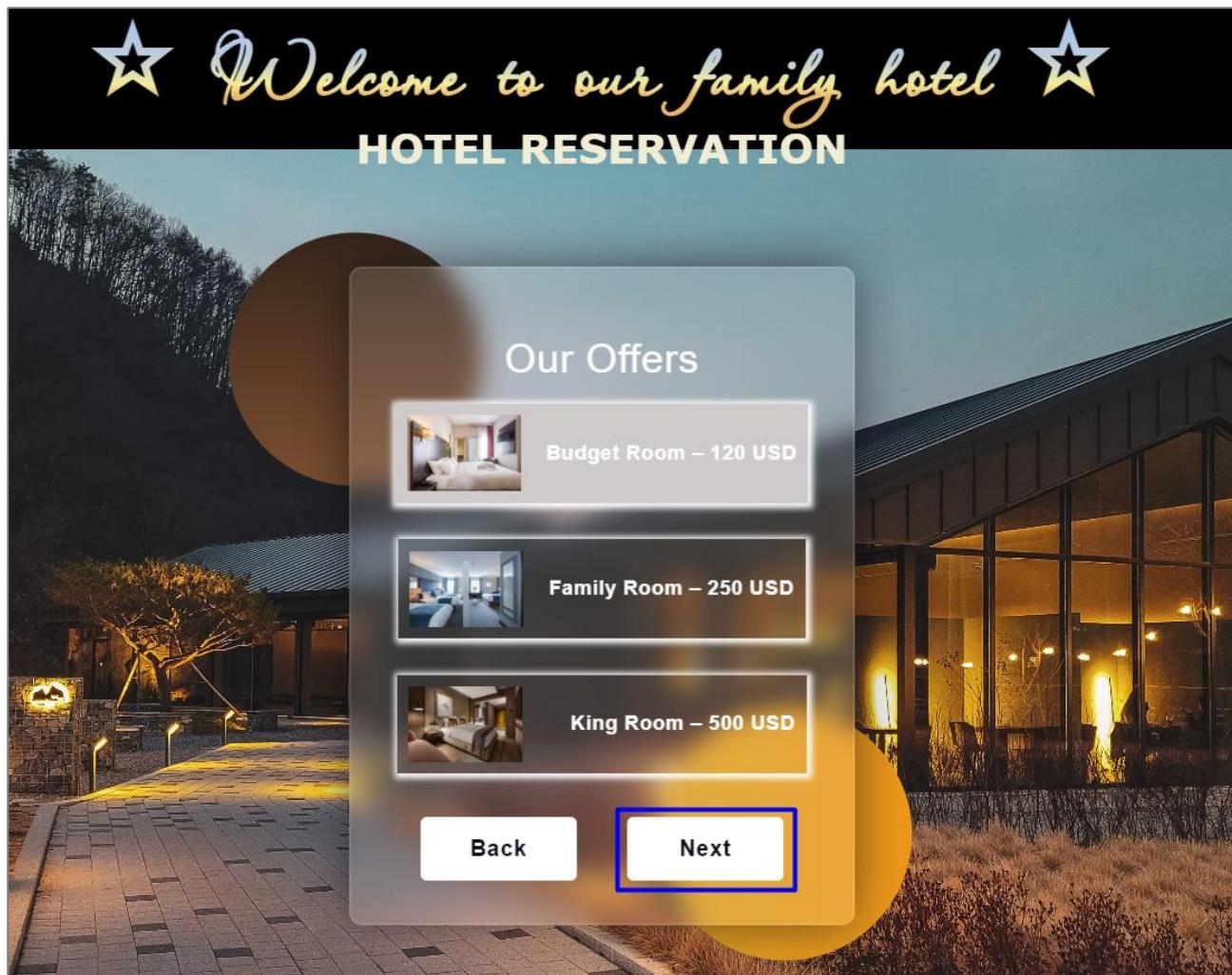
```
changeContent('search-result-form-content');
```

Step 5: Search Form: CSS

In **styles.css**, they add **styles (Offerer/styles.css** file from the resources) to the **offers** form.

Step 6: Test the Project Functionality

Offerer now **tests the project functionality** to see whether SPA app works correctly, as well as whether the entire project works as expected:



```
solution.js:59
{
  startDate: null,
  endDate: null,
  guestsCount: 0,
  roomType: 'Budget Room - 120 USD',
  name: null,
  email: null,
  endDate: null,
  guestsCount: 0,
  name: null,
  phone: null,
  roomType: "Budget Room - 120 USD",
  startDate: null
}
▶ [[Prototype]]: Object
```

```
✖ ▶ Uncaught TypeError: Cannot read properties of null (reading
  'classList')
  at changeContent (solution.js:14:44)
  at findRoom (solution.js:60:5)
  at HTMLButtonElement.<anonymous> (solution.js:53:77)
```

>

Don't worry about the error!

Step 7: Commit All Changes to the Local Branch

Offerer adds and commits in Git all **local changes**:

```
git commit -a -m "Implemented Our Offers form functionality"
```

Now we have a **new branch**, that is different from **main**. If we now **push** to the **repo**, we will get an **error** because the **our-offers-branch** branch is **local**, not **upstream**, e.g., it is **not** in **GitHub** yet, only **locally**:

```
git push
```

The screenshot shows a GitHub repository interface for 'Offerer / SoftUni-Branches'. The repository is public. The navigation bar includes 'Code', 'Issues', 'Pull requests', 'Actions', 'Projects', and a three-dot menu. Below the navigation is a tab bar with 'Overview' (highlighted in blue), 'Yours', 'Active', 'Stale', 'All', and a green 'New branch' button. A search bar labeled 'Search branches...' is present. Under 'Default branch', the 'main' branch is listed, updated 31 minutes ago by 'offerer'. The overall layout is clean and modern.

To fix this **error** we should run the **following command**.

```
git push --set-upstream origin our-offers-form
```

The "**--set-upstream**" option is **utilized** to **set** the **remote** as the **upstream** directory and **fix** the above-encountered **error**.

The screenshot shows the same GitHub repository interface after running the command. Now, under 'Default branch', the 'main' branch is listed, updated 34 minutes ago by 'offerer'. In the 'Your branches' section, the 'our-offers-form' branch is listed, updated 3 minutes ago by 'offerer'. In the 'Active branches' section, the 'our-offers-form' branch is also listed, updated 3 minutes ago by 'offerer'. This indicates that the local branch has been successfully set up as the upstream branch.

We have the **upstream branch our-offers-form**.

Step 8: Create a Pull Request

Now go to your **GitHub repo** and **create a pull request** for merge from the **our-offers-form** to the **main branch**:

Don't forget to add a **reviewer**.

Reviewers

Request up to 15 reviewers

Type or choose a user

  Senior

Add a title and **create the pull request**:

Implemented Our Offers form functionality #5

Open offerer wants to merge 1 commit into `main` from `our-offers-form`

Conversation 0 Commits 1 Checks 0 Files changed 3

offerer commented 1 minute ago Owner ...

No description provided.

offerer Implemented Our Offers form functionality a213a97

Step 9: Approve a Pull Request

Now you should **approve** one of your friend's **pull requests**. But before that, we should **resolve** the **conflicts**.

Code Issues Pull requests 1 Actions Projects Wiki Security ...

Label issues and pull requests for new contributors Dismiss

Now, GitHub will help potential first-time contributors [discover issues](#) labeled with [good first issue](#)

Filters ▾ Q is:pr is:open Labels 9 Milestones 0 New pull request

3 Open ✓ 3 Closed

Author ▾ Label ▾ Projects ▾ Milestones ▾ Reviews ▾ Assignee ▾ Sort ▾

Implemented Search form functionality #4 opened 12 minutes ago by questioner

This branch has conflicts that must be resolved Resolve conflicts

Use the [web editor](#) or the [command line](#) to resolve conflicts.

Conflicting files

index.html

Merge pull request ▾

You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

Implemented Search form functionality #4

Resolving conflicts between `search-form` and `main` and committing changes → `search-form`

1 conflicting file

	index.html	1 conflict	Prev ⏪	Next ⏹	⚙️	Mark as resolved
index.html	<pre>1 <!DOCTYPE html> 2 <html lang="en"> 3 4 <head> 5 <meta charset="UTF-8"> 6 <meta http-equiv="X-UA-Compatible" content="IE=edge"> 7 <link rel="stylesheet" href="./static/css/styles.css"> 8 <title>Hotel reservation</title> 9 </head> 10 11 <body> 12 <section id="welcome"> 13 <h1>Welcome to our family hotel</h1> 14 <div class="home-container"> 15 <div class="info"> 16 <h1>Hotel reservation</h1></pre>					

When **conflicts are resolved**, merge the changes:

Implemented Search form functionality #4

Resolving conflicts between `search-form` and `main` and committing changes → `search-form`

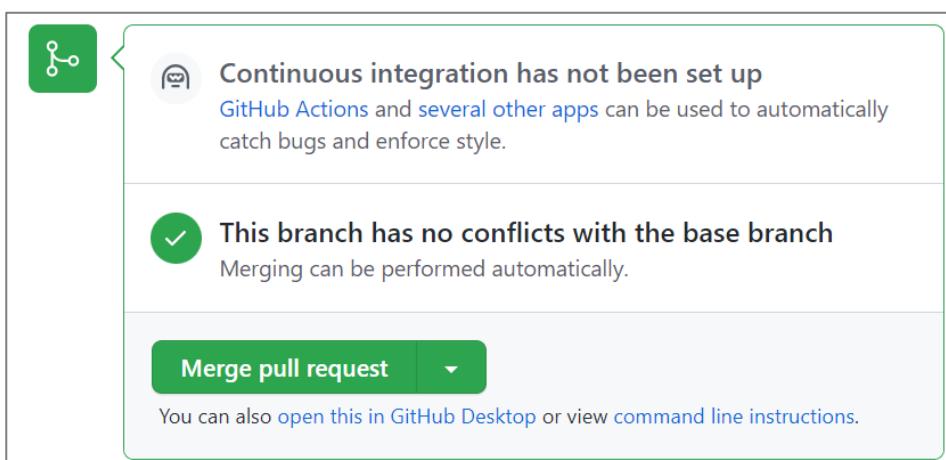
Commit merge

1 conflicting file

	index.html	✓ Resolved
index.html	<pre>27 28 </div> 29 <form> 30 <h3>Search Here</h3> 31 32 <label for="check-in">Check-in:</label> 33 <input type="date" id="check-in" name="check-in"> 34 35 <label for="check-out">Check-out:</label> 36 <input type="date" id="check-out" name="check-out"> 37 38 <label for="people">Numbers of people:</label> 39 <input type="number" min="1" id="people" name="phone-number"></pre>	✓ Resolved

Step 10: Merge Pull Request

Merge the **search-form** branch into the **main** branch.



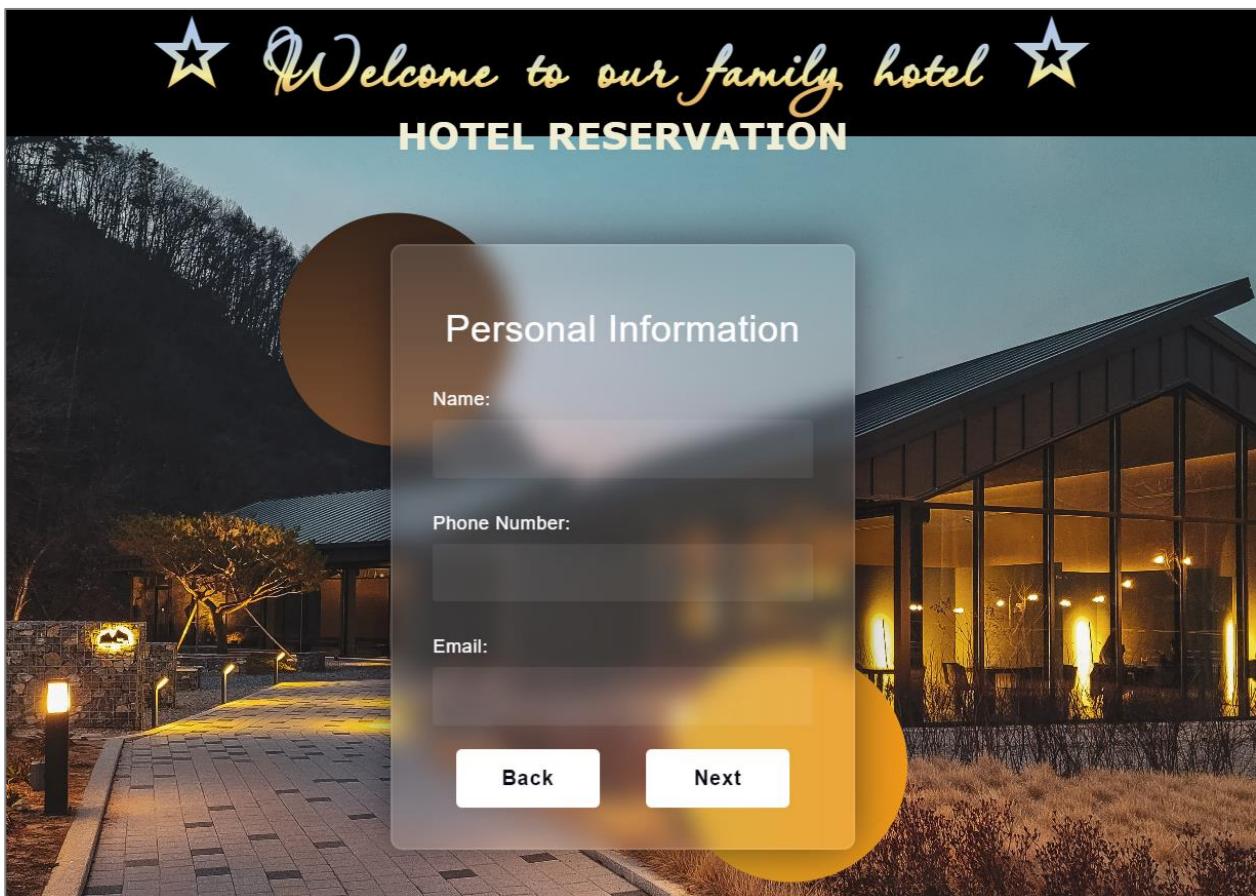
 Pull request successfully merged and closed

You're all set—the `search-form` branch can be safely deleted.

[Delete branch](#)

Admin

The **Admin** should implement the "**Guest Details Form**" functionality:



The **Admin's task** is to **collect** and **store** the **data** for each **person** (**name**, **phone number** and **email**).

The **Admin clones** the project repository, makes a sequence of **changes** in the source code files, **commits** locally in repository, then **pushes** the committed changes to GitHub.

Step 1: Clone the "Hotel Reservation" Repository

The **Admin** should already have **cloned locally** the **main project repo**.

Step 2: Create a Local Branch

We want to **add some new features** to our SPA app. We are working in our **local repository**, and we do not want to disturb or wreck the main project. So, we create a **new local branch**:

```
git branch guest-details-form
```

We have **two branches** now: **main** and **guest-details-form**.

Now we should check out the **new branch**, e.g., switch from the **current branch** to the **new one**.

```
git checkout guest-details-form
```

We have moved our **current workspace** from the **main branch**, to the **guest-details-form branch**.

Step 3: Search Form: HTML

In **index.html**, in **div** with class "**site-content**" he adds some **HTML** (**Admin/index.html** file from the resources) for the **guest details form**.

Step 4: Search Form: JavaScript Code

In **solution.js**, they add a new function (**Admin/solution.js** file from the resources) to the **guest details form**.

Use this **code box** only **while testing functionality** in your **custom branch**.

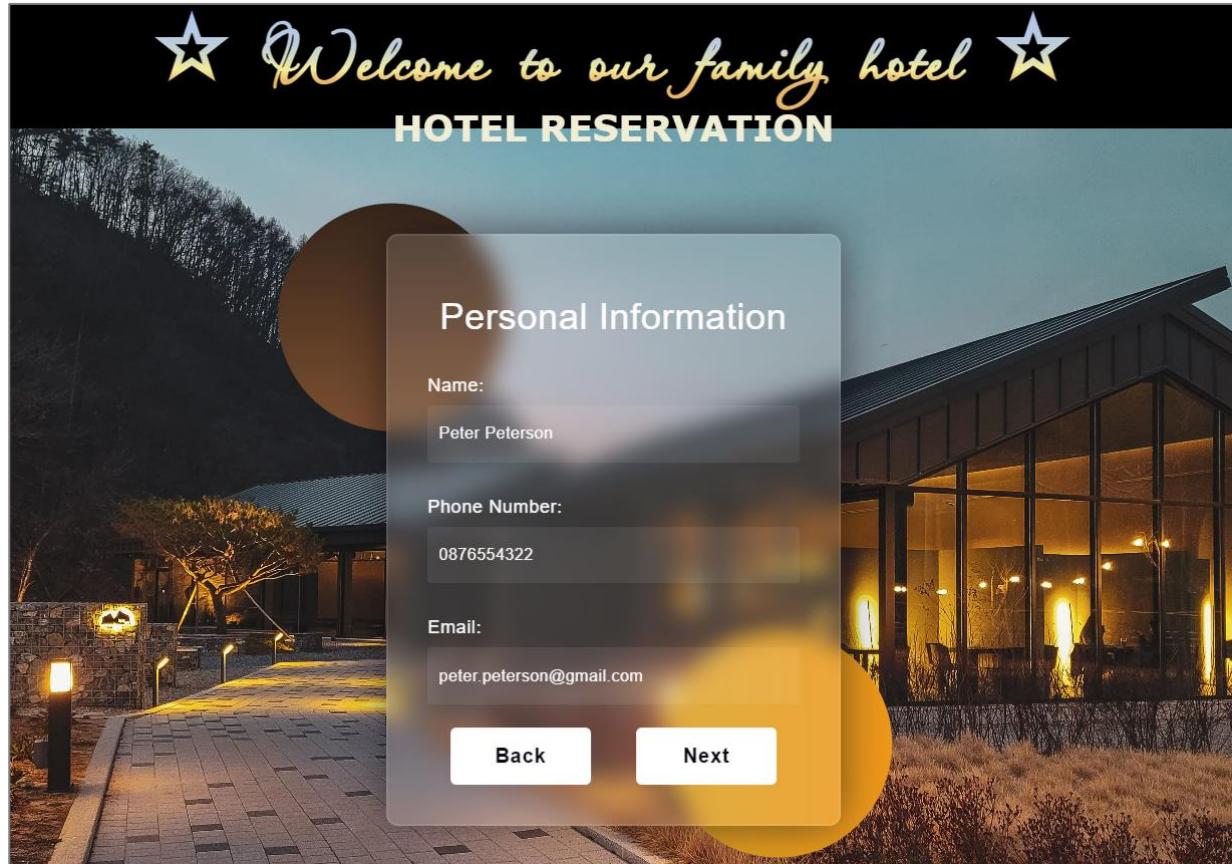
```
changeContent('guest-details-form-content');
```

Step 5: Search Form: CSS

In **styles.css**, they add **styles** (**Admin/styles.css** file from the resources) to the **guest details form**.

Step 6: Test the Project Functionality

The **Admin** now **tests the project functionality** to see whether SPA app works correctly, as well as whether the entire project works as expected:



```

solution.js:48
  {startDate: null, endDate: null, guestsCount: 0, roomType: null, name: 'Peter Peterson', ...} ⓘ
    email: "peter.peterson@softuni.org"
    endDate: null
    guestsCount: 0
    name: "Peter Peterson"
    phone: "0876543222"
    roomType: null
    startDate: null
  ► [[Prototype]]: Object
✖ ► Uncaught TypeError: Cannot read properties of null solution.js:14 ⚭
  (reading 'classList')
    at changeContent (solution.js:14:44)
    at getPersonalData (solution.js:49:9)
    at HTMLButtonElement.<anonymous> (solution.js:34:84)
  >

```

Don't worry about the error!

Step 7: Commit All Changes to the Local Branch

The **Admin adds** and **commits** in Git all **local changes**:

```
git commit -a -m "Implemented Guest Details form functionality"
```

Now we have a **new branch**, that is different from **main**. If we now **push** to the **repo**, we will get an **error** because the **guest-details-form branch** is **local**, **not upstream**, e.g., it is **not in GitHub** yet, only **locally**:

```
git push
```

Admin / SoftUni-Branches Public

Code Issues Pull requests 1 Actions Projects Wiki ...

Overview Yours Active Stale All New branch

Search branches...

Default branch

main Updated 16 minutes ago by admin

To fix this **error** we should run the **following command**.

```
git push --set-upstream origin guest-details-form
```

The "**--set-upstream**" option is **utilized** to set the **remote** as the **upstream** directory and **fix** the above-encountered error.

Admin / SoftUni-Branches Public

<> Code Issues Pull requests 1 Actions Projects Wiki ...

Overview Yours Active Stale All New branch

Search branches...

Default branch

main Updated 17 minutes ago by admin

Your branches

guest-details-form Updated 14 minutes ago by admin

Active branches

guest-details-form Updated 14 minutes ago by admin

We have the **upstream branch guest-details-form**.

Step 8: Create a Pull Request

Now go to your **GitHub repo** and **create a pull request** for merge from the **guest-details-form** to the **main branch**:

guest-details-form had recent pushes less than a minute ago

Compare & pull request

guest-details-... ▾ Go to file Add file ▾ <> Code ▾

This branch is 1 commit ahead of main. Contribute ▾

[Open a pull request](#)

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).

Don't forget to add a reviewer.

Reviewers



Request up to 15 reviewers

Type or choose a user

  Senior

Step 9: Approve a Pull Request

Now you should **approve** one of your friend's **pull requests**. But before that, we should **resolve** the **conflicts**.

Code Issues Pull requests 1 Actions Projects Wiki Security ...

Label issues and pull requests for new contributors Dismiss

Now, GitHub will help potential first-time contributors [discover issues](#) labeled with [good first issue](#)

Filters is:pr is:open Labels 9 Milestones 0 New pull request

3 Open 3 Closed

Author Label Projects Milestones Reviews Assignee Sort

Implemented Search form functionality #4 opened 12 minutes ago by questioner

This branch has conflicts that must be resolved [Resolve conflicts](#)

Use the [web editor](#) or the [command line](#) to resolve conflicts.

Conflicting files

index.html

[Merge pull request](#)

You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

Implemented Search form functionality #4

Resolving conflicts between `search-form` and `main` and committing changes → `search-form`

1 conflicting file	index.html	1 conflict	Prev ⌂	Next ⌂	⚙️	Mark as resolved
<code>index.html</code>	<pre>1 <!DOCTYPE html> 2 <html lang="en"> 3 4 <head> 5 <meta charset="UTF-8"> 6 <meta http-equiv="X-UA-Compatible" content="IE=edge"> 7 <link rel="stylesheet" href="./static/css/styles.css"> 8 <title>Hotel reservation</title> 9 10 </head> 11 <body> 12 <section id="welcome"> 13 <h1>Welcome to our family hotel </h1> 14 <div class="home-container"> 15 <div class="info"> 16 <h1>Hotel reservation</h1></pre>					

When **conflicts are resolved**, merge the changes:

Implemented Search form functionality #4

Resolving conflicts between `search-form` and `main` and committing changes → `search-form`

[Commit merge](#)

1 conflicting file index.html ✓ Resolved

index.html ✓

```
27      </div>
28      <form>
29          <h3>Search Here</h3>
30
31          <label for="check-in">Check-in:</label>
32          <input type="date" id="check-in" name="check-in">
33
34          <label for="check-out">Check-out:</label>
35          <input type="date" id="check-out" name="check-out">
36
37          <label for="people">Numbers of people:</label>
38          <input type="number" min="1" id="people" name="phone-number">
39
```

Step 10: Merge Pull Request

Merge the `search-form` branch into the `main` branch.

Continuous integration has not been set up
GitHub Actions and several other apps can be used to automatically catch bugs and enforce style.

This branch has no conflicts with the base branch
Merging can be performed automatically.

[Merge pull request](#)

You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

Merge pull request #4 from admin/search-form

Implemented Search form functionality.

This commit will be authored by questioner

[Confirm merge](#) [Cancel](#)

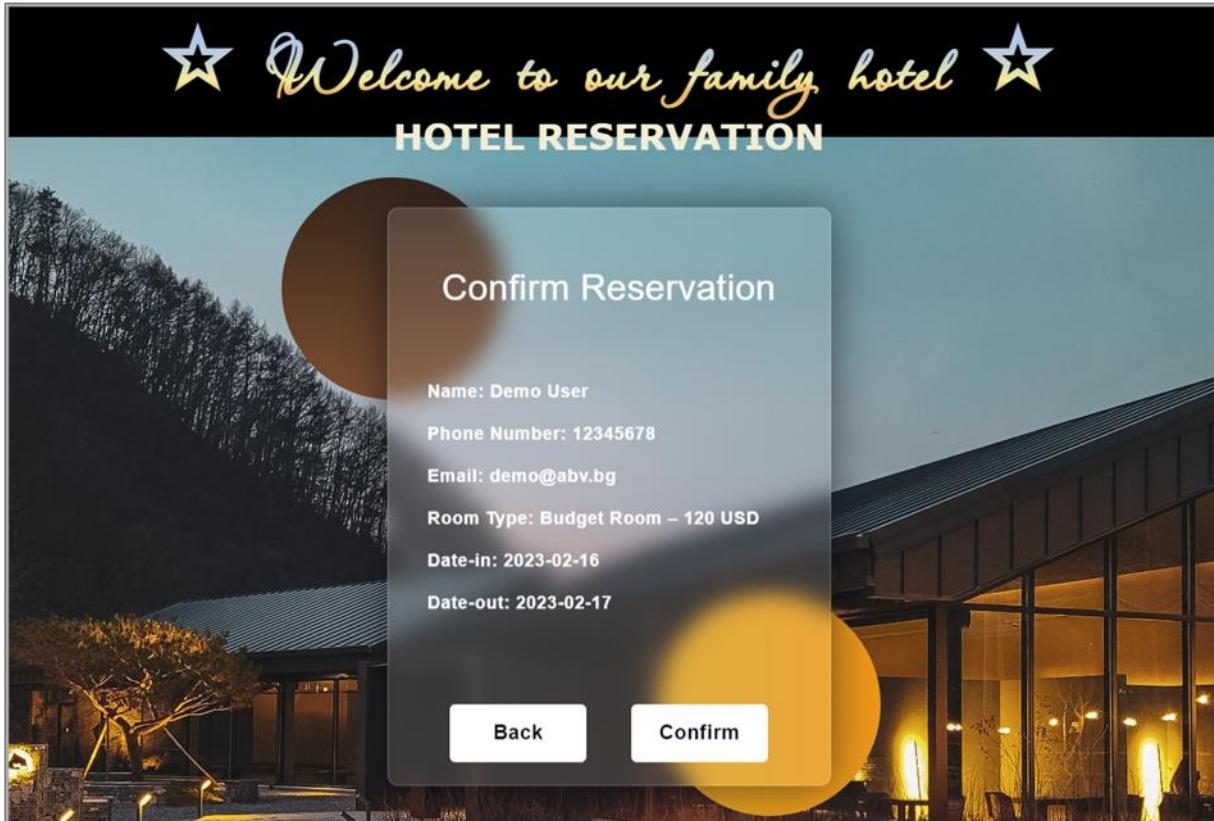
Pull request successfully merged and closed

[Delete branch](#)

You're all set—the `search-form` branch can be safely deleted.

Verifier

Verifier should implement the "Confirm Reservation Form" functionality:



The **Verifier's** task is to **confirm personal data**.

Verifier clones the project repository, makes a sequence of **changes** in the source code files, **commits** locally in repository, then **pushes** the committed changes to GitHub.

Step 1: Clone the "Hotel Reservation" Repository

The **Verifier** should already have **cloned locally** the **main project repo**.

Step 2: Create a Local Branch

We want to **add some new features** to our SPA app. We are working in our **local repository**, and we do not want to disturb or wreck the main project. So, we create a **new local branch**:

```
git branch confirm-reservation-form
```

We have **two branches** now: **main** and **confirm-reservation-form**.

Now we should check out the **new branch**, e.g., switch from the **current branch** to the **new one**.

```
git checkout confirm-reservation-form
```

We have moved our **current workspace** from the **main branch**, to the **confirm-reservation-form branch**.

Step 3: Search Form: HTML

In **index.html**, in **div** with **class "site-content"** he adds some **HTML** (**Verifier/index.html** file from the resources) for the **confirm reservation form**.

Step 4: Search Form: JavaScript Code

In **solution.js**, he adds a new function (**Verifier/solution.js** file from the resources) to the **confirm reservation form**.

Use this **code box** only **while testing functionality** in your **custom branch**.

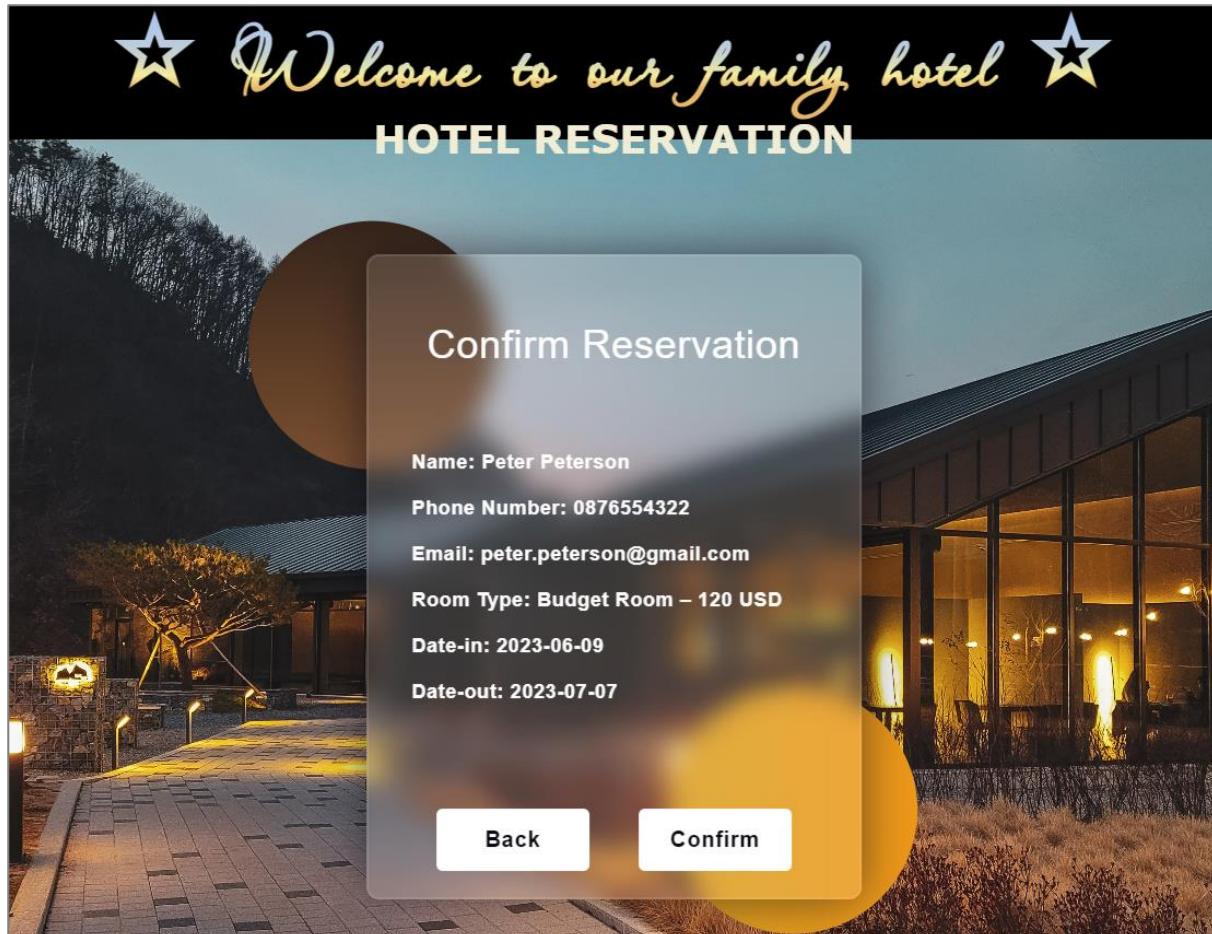
```
changeContent('confirm-reservation-content');
```

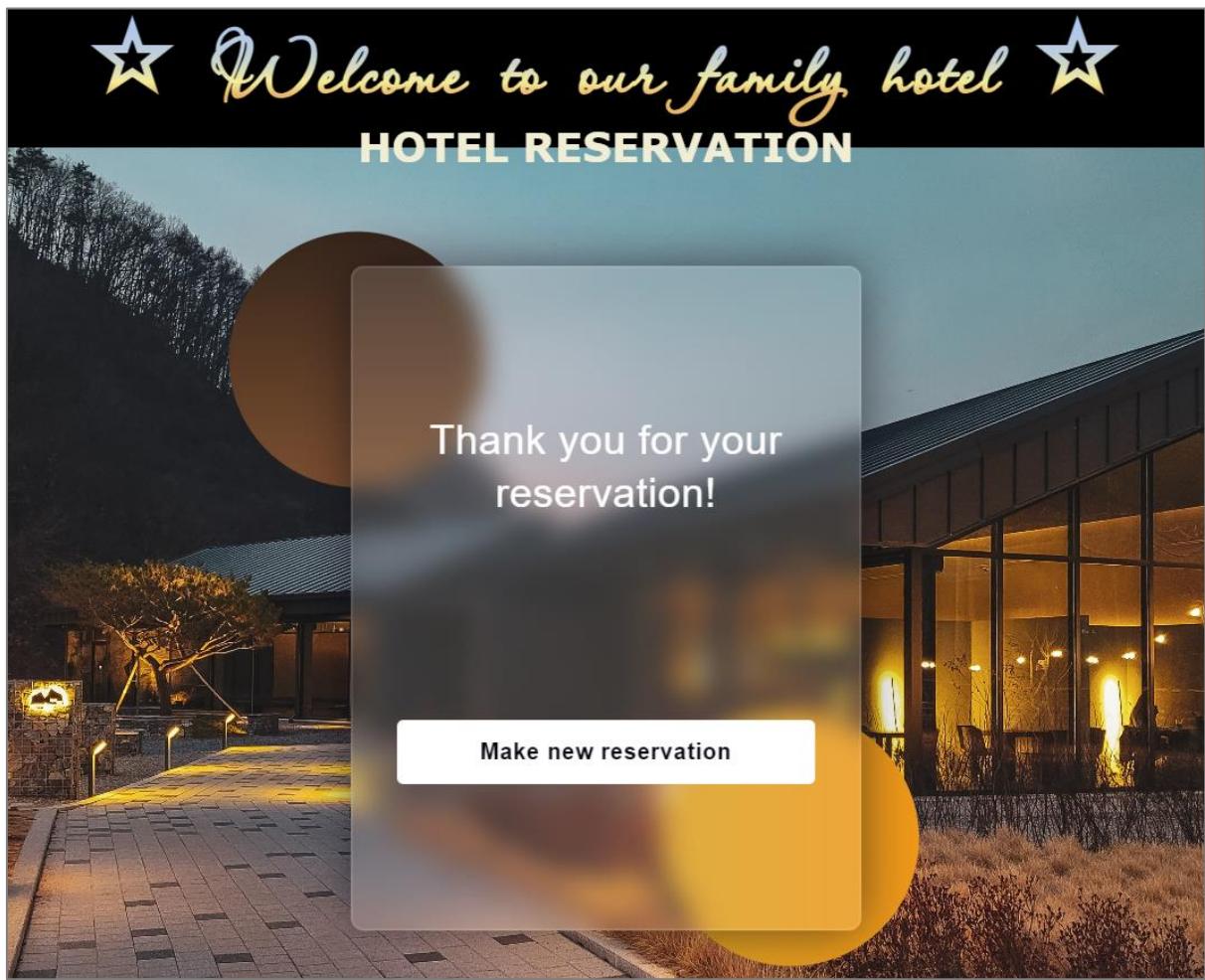
Step 5: Search Form: CSS

In **styles.css**, he adds **styles (Verifier/styles.css** file from the resources) to the **confirm reservation form**.

Step 6: Test the Project Functionality

Verifier now **tests the project functionality** to see whether SPA app works correctly, as well as whether the entire project works as expected:





Step 7: Commit All Changes to the Local Branch

Verifier adds and commits in Git all **local changes**:

```
git commit -a -m "Implemented Confirm Reservation form functionality"
```

Now we have a **new branch**, that is different from **main**. If we now **push** to the **repo**, we will get an **error** because the **confirm-reservation-form** branch is **local**, not **upstream**, e.g., it is **not in GitHub** yet, only **locally**:

```
git push
```

To fix this **error** we should run the **following command**.

```
git push --set-upstream origin confirm-reservation-form
```

The "**--set-upstream**" option is **utilized** to **set** the **remote** as the **upstream** directory and **fix** the above-encountered error.

The screenshot shows a GitHub repository interface. At the top, there's a navigation bar with links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, and three dots. Below the navigation bar is a horizontal menu with tabs: Overview (which is highlighted in blue), Yours, Active, Stale, All, and New branch (which is green). A search bar below the menu contains the placeholder "Search branches...". Under the "Default branch" section, there's a card for the "main" branch, which was updated 21 minutes ago by verifier. In the "Your branches" and "Active branches" sections, there's a card for the "confirm-reservation-form" branch, which was updated 3 minutes ago by verifier.

We have the **upstream branch** **confirm-reservation-form**.

Step 8: Create a Pull Request

Now go to your **GitHub repo** and **create a pull request** for merge from the **confirm-reservation-form** to the **main branch**:

The screenshot shows a modal dialog box with a yellow background. On the left, there's a message: "confirm-reservation-form had recent pushes less than a minute ago". On the right, there's a green button with the text "Compare & pull request". The entire dialog is enclosed in a blue border.

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).

The screenshot shows a pull request creation interface. At the top, it says "base: main" and "compare: confirm-reservation-form". A green checkmark indicates "Able to merge. These branches can be automatically merged." Below this, there's a text area with the placeholder "Implemented Confirm Reservation form functionality". There are "Write" and "Preview" buttons, and a toolbar with various icons. A large text input field for "Leave a comment" is below the main text area. At the bottom, there's a file upload section with the instruction "Attach files by dragging & dropping, selecting or pasting them." and a "Create pull request" button.

Don't forget to **add a reviewer**.

The screenshot shows a "Reviewers" section. It has a "Request up to 15 reviewers" input field and a "Type or choose a user" search bar. Below this, a user named "Senior" is selected, indicated by a checked checkbox icon and the name "Senior".

Add a **title** and **create the pull request**:

The screenshot shows a pull request details page. The title is "Implemented Confirm Reservation form functionality #7". It says "verifier wants to merge 1 commit into main from confirm-reservation-form". Below this, there are stats: Conversation 0, Commits 1, Checks 0, Files changed 3. A comment from "verifier" is shown, stating "commented 1 minute ago" and "No description provided.". A commit message "Implemented Confirm Reservation form functionality" is listed with a hash "69398a6".

Step 9: Approve a Pull Request

Now you should **approve** one of your friend's **pull requests**. But before that, we should **resolve the conflicts**.

Code Issues Pull requests 1 Actions Projects Wiki Security ...

Label issues and pull requests for new contributors Dismiss

Now, GitHub will help potential first-time contributors [discover issues](#) labeled with [good first issue](#)

Filters is:pr is:open Labels 9 Milestones 0 New pull request

3 Open 3 Closed

Author Label Projects Milestones Reviews Assignee Sort

Implemented Search form functionality #4 opened 12 minutes ago by questioner

This branch has conflicts that must be resolved [Resolve conflicts](#)

Use the [web editor](#) or the [command line](#) to resolve conflicts.

Conflicting files

index.html

Merge pull request ▾

You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

Implemented Search form functionality #4

Resolving conflicts between `search-form` and `main` and committing changes → `search-form`

1 conflicting file	index.html	1 conflict	Prev ^	Next v	⚙️	Mark as resolved
<code>index.html</code>	<pre>1 <!DOCTYPE html> 2 <html lang="en"> 3 4 <head> 5 <meta charset="UTF-8"> 6 <meta http-equiv="X-UA-Compatible" content="IE=edge"> 7 <link rel="stylesheet" href="./static/css/styles.css"> 8 <title>Hotel reservation</title> 9 </head> 10 <body> 11 <section id="welcome"> 12 <h1>Welcome to our family hotel </h1> 13 <div class="home-container"> 14 <div class="info"> 15 <h1>Hotel reservation</h1></pre>					

When **conflicts are resolved**, merge the changes:

Implemented Search form functionality #4

Resolving conflicts between `search-form` and `main` and committing changes → `search-form`

Commit merge

✓ Resolved

index.html

```
27      </div>
28      <form>
29          <h3>Search Here</h3>
30
31          <label for="check-in">Check-in:</label>
32          <input type="date" id="check-in" name="check-in">
33
34          <label for="check-out">Check-out:</label>
35          <input type="date" id="check-out" name="check-out">
36
37          <label for="people">Numbers of people:</label>
38          <input type="number" min="1" id="people" name="phone-number">
```

Step 10: Merge Pull Request

Merge the `search-form` branch into the `main` branch.

 Continuous integration has not been set up
GitHub Actions and several other apps can be used to automatically catch bugs and enforce style.

 This branch has no conflicts with the base branch
Merging can be performed automatically.

Merge pull request ▾

You can also [open this in GitHub Desktop](#) or view command line instructions.

 Pull request successfully merged and closed Delete branch

You're all set—the `search-form` branch can be safely deleted.

Step 5: Merge and Create / Approve Pull Requests

When done with their functionalities, each team member should **merge their feature branch** into the **main branch** of the **GitHub repo**.

- Each team member should **create a pull request**. When the **pull request is approved** and the **branch is successfully merged**, they should **delete the feature branch**.
- Each team member should **accept one pull request of another collaborator**.

You should also **solve conflicts that appear on merge**. At the end, all **pull requests should be approved** and all **feature branches** should be **merged into main** and **deleted**.