



# UI/UX DESIGN

QTrip Static  
Qtrip Dynamic

# Agenda

---

## QTrip Static

1. Create layout for the landing page and make it responsive
2. Implement the adventures page
3. Add the adventure details page to complete the static web page features
4. Deploy the QTripStatic website

Overview

## QTrip Dynamic

1. Create the city cards for QTrip landing page
2. Create the cards for adventures grid page
3. Create filters for adventures
4. Create the adventure details page
5. Add support for adventure reservations
6. Deploy the QTrip website using render and netlify

Overview

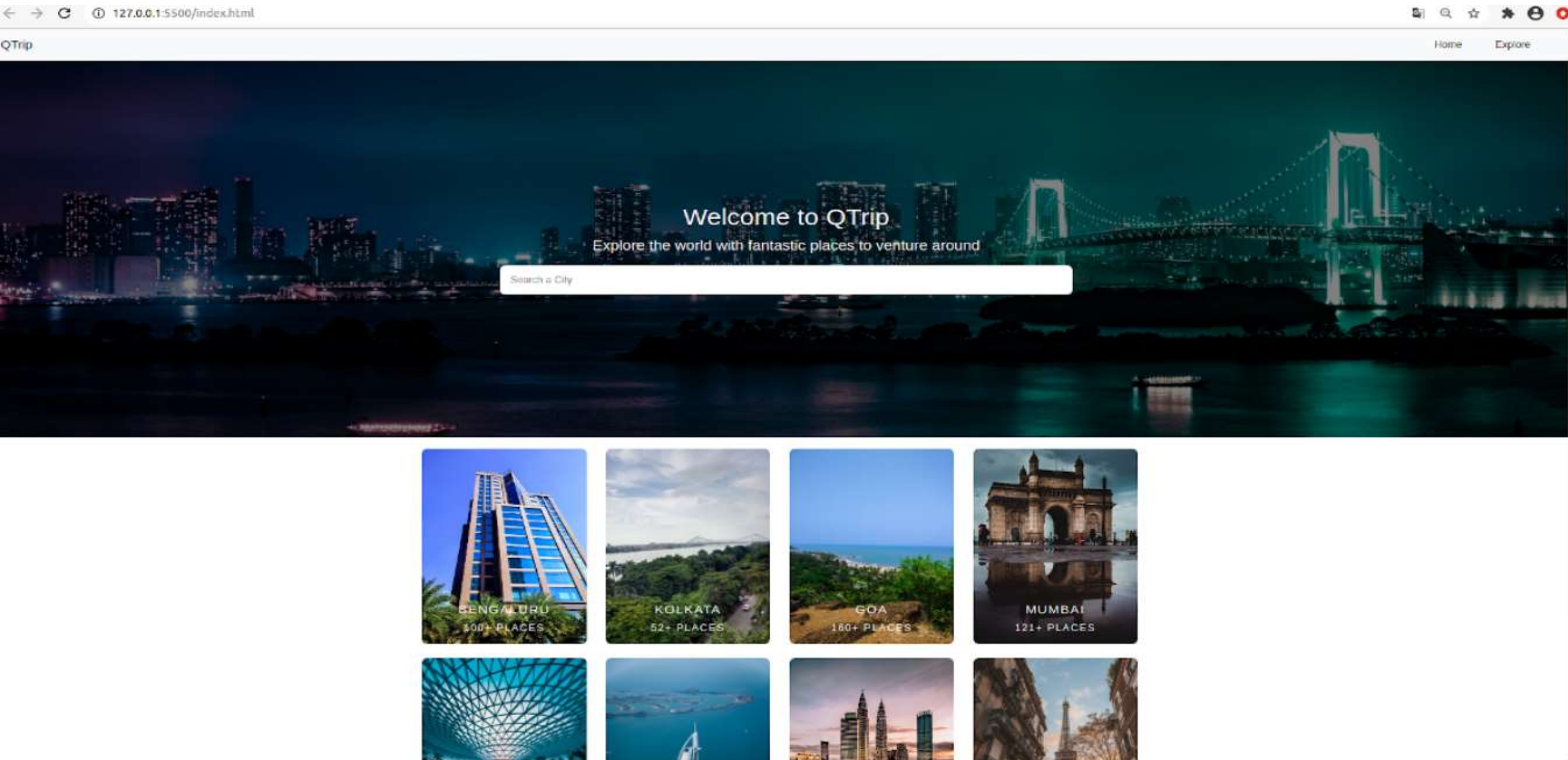
# Qtrip Static

---

QTrip is a travel website aimed at travelers looking for a multitude of adventures in different cities.

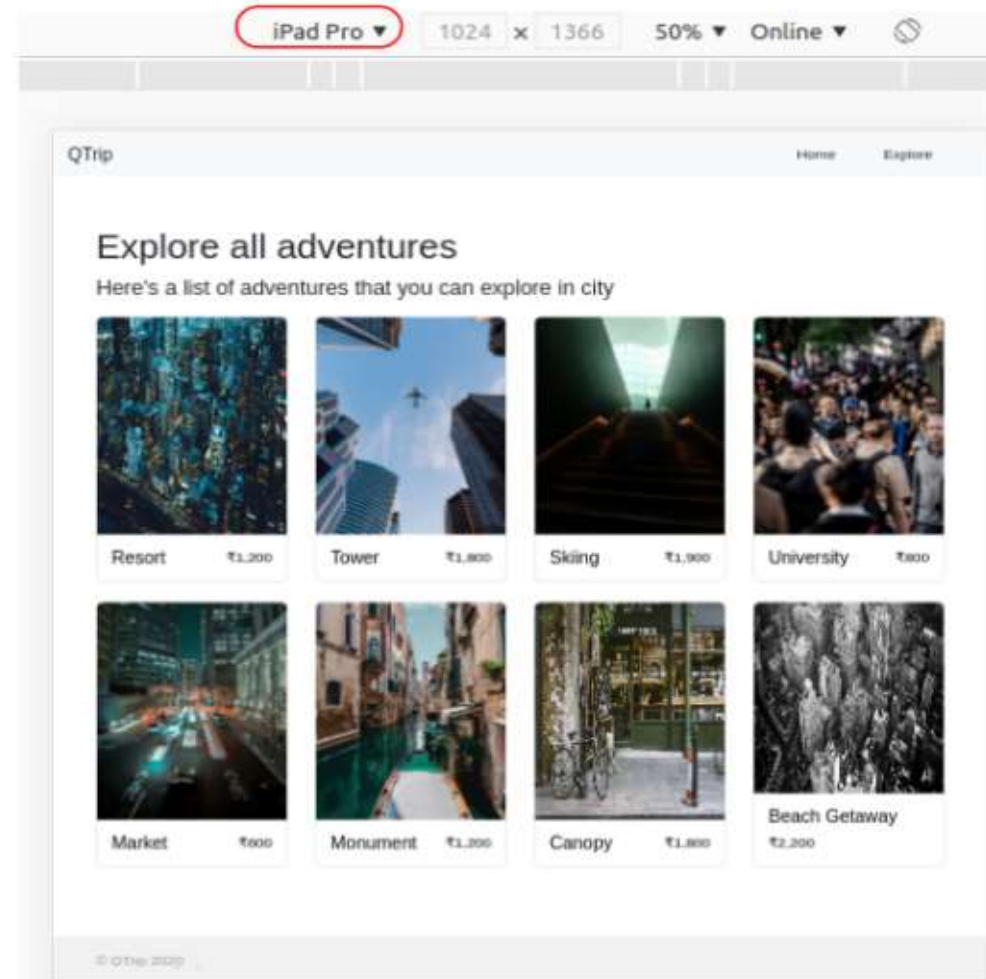
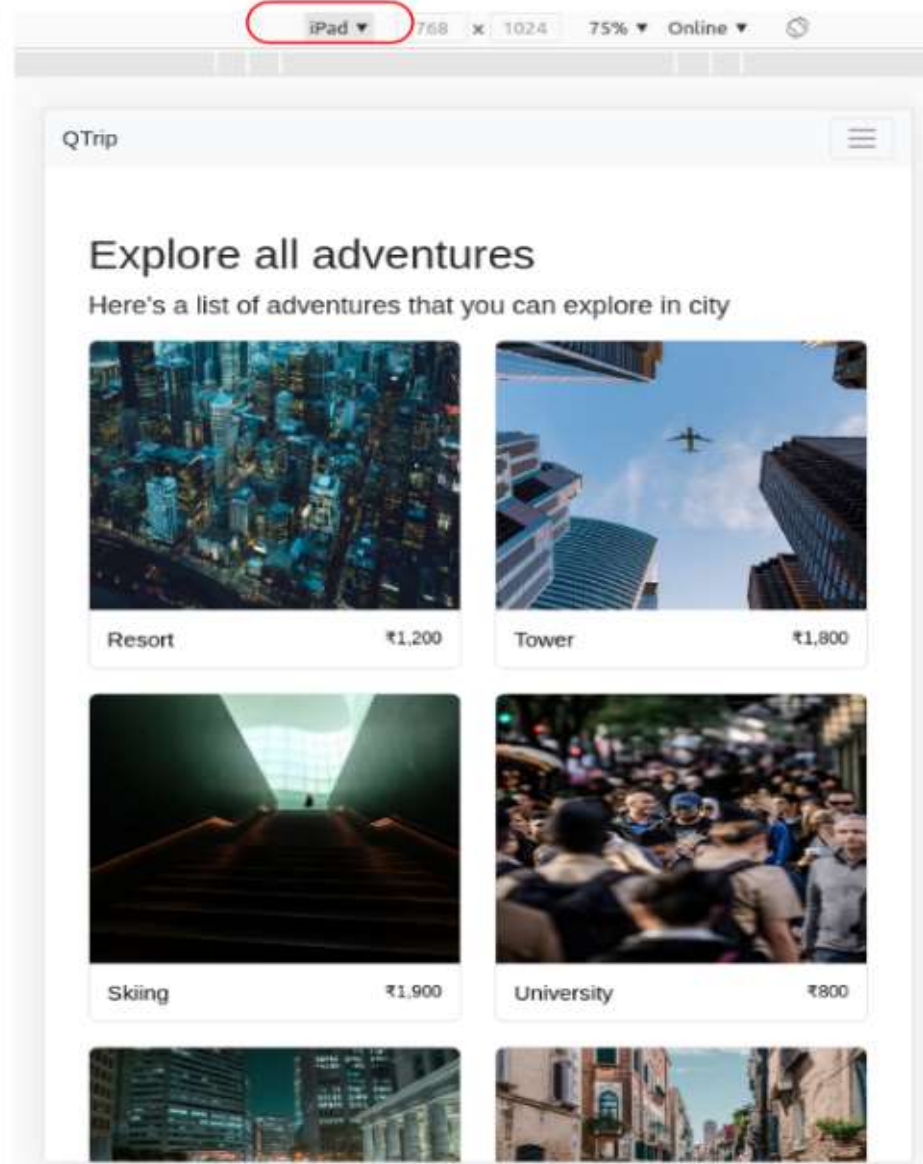
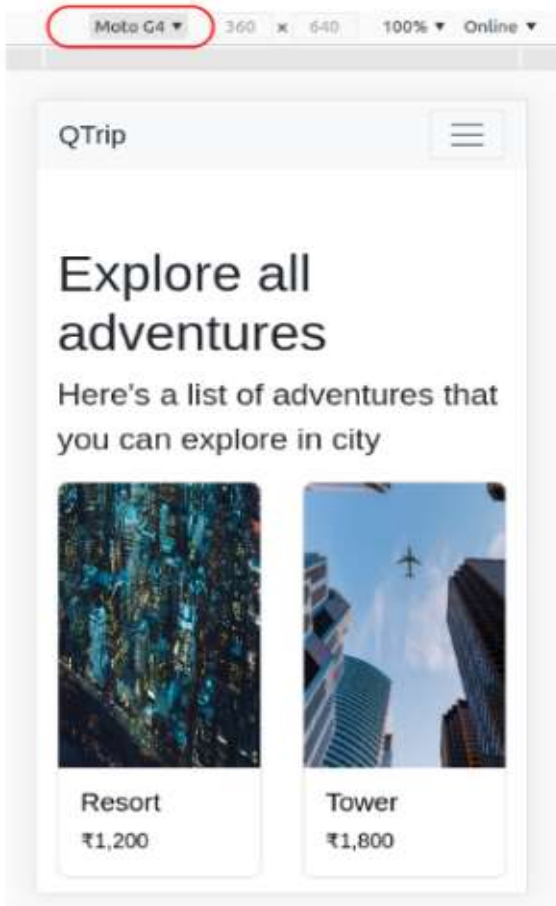


# QTrip Landing page





# QTrip Adventures page



# QTrip Adventure details page

← → ↻ ⓘ 127.0.0.1:5500/pages/adventures/resort/index.html



QTrip

Home

Explore



## Sold Out!

This adventure is currently sold out. But there's a lot more to [explore](#).

## Sunshine Resort

Relax and unwind yourself

### About the Experience

Chairman's Jade Resort is constructed in the most natural setting for a day out that pleases the senses. The property is spread over eight acres of land with facilities for both indoor and outdoor adventures. Get active in the pool with a game of water polo or try some archery. For the indoor sort, there are a host of popular games and adventures. If that is not enticing enough head to the gymnasium!

### Inclusions

- Swimming pool access
- Outdoor sports - Water Polo, Archery, Tennis, and Zorbing
- Indoor games - Indoor Squash, Table Tennis, Carrom, Badminton, Chess, Billiards, and X-Box 360 KINECT
- Parking

### Know before you go

- Game accessories like balls and shuttlecocks will need to be purchased
- All adventures and game accessories are subject to availability
- Nylon swimwear must be worn at all times when using the swimming pool
- Children must be accompanied by parents or guardians at all times
- Complimentary entry for children below the height of three feet A valid photo ID is mandatory to

# 1. Create layout for the landing page and make it responsive

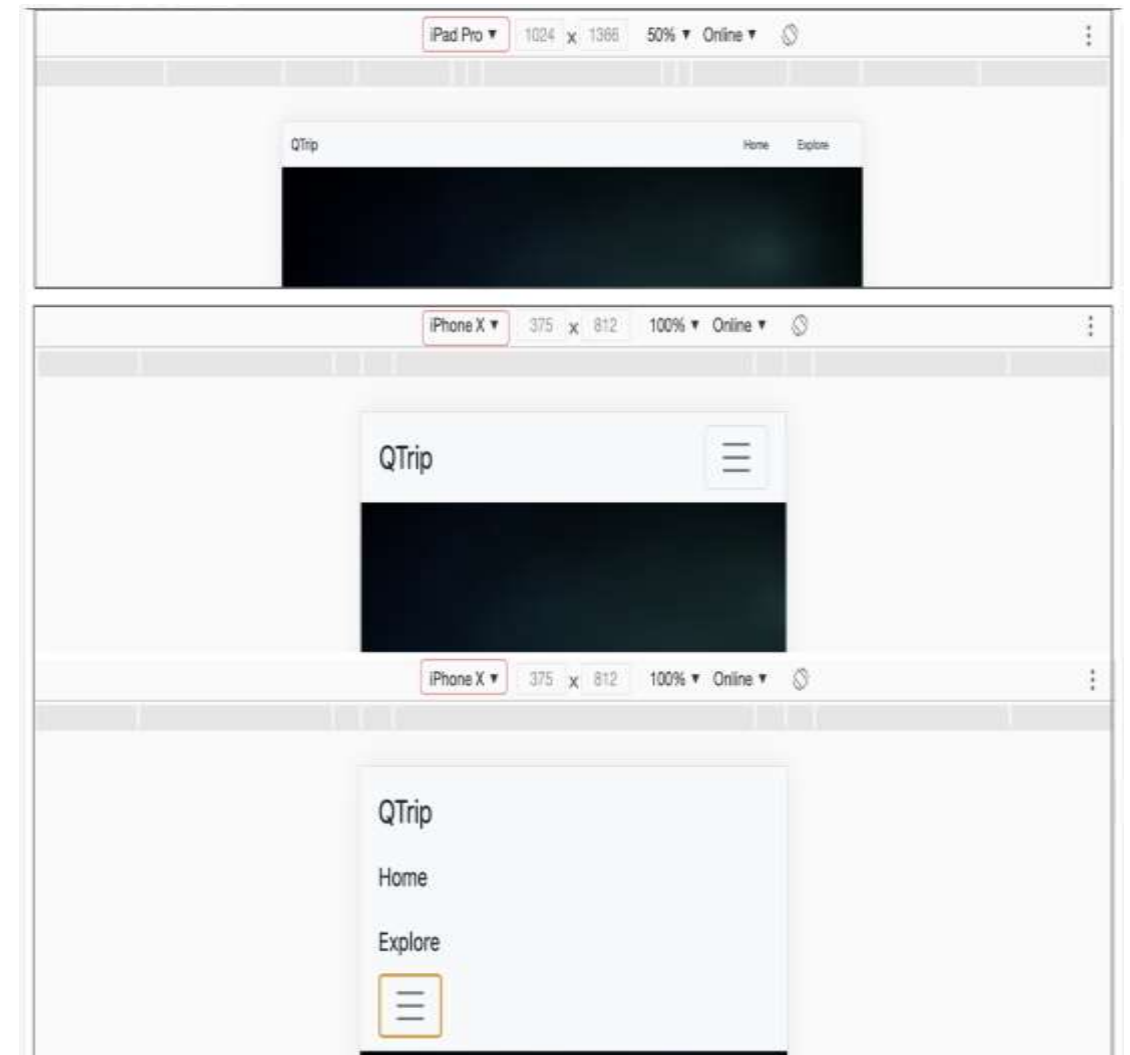
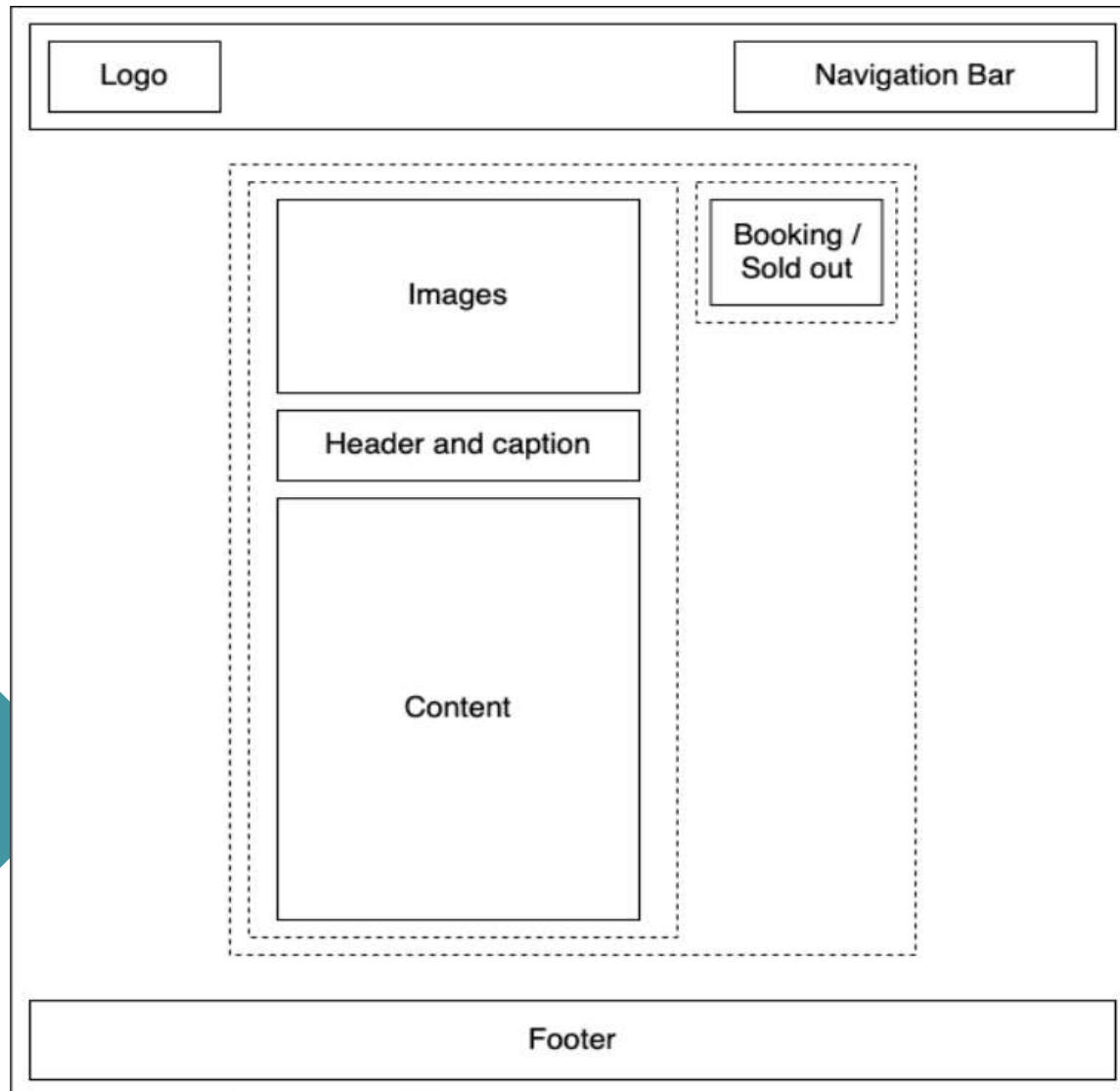
---

- Added a navigation bar which collapses automatically on smaller devices to display a toggle icon
- Utilized CSS properties to add the hero image and align elements in it as per the design
- Implemented the cities grid section by using Bootstrap's grid
- Improved the cities grid section by making images responsive and added effect on hovering over an image tile

## **Skills used**

- HTML, CSS, CSS Flexbox, Responsive Design, Bootstrap, Developer Tools

## Wireframe used for building QTrip landing page



**Responsive navigation bar  
(Top: Expanded,  
Bottom: Collapsed/Expanded)**



# Add Navigation Bar using Bootstrap

1. The `<nav>` element in the Landing Page's `index.html` is empty and needs to be responsive for various devices:

- mobile (375px)
- tablet (640px)
- desktop (1280px)

2. Use Chrome Developer Tools to simulate these screen sizes.

3. Add the required CSS and JS files to set up Bootstrap

4. Use Bootstrap classes to set the behaviour and view of the `<nav>` element similar to the one shown above, for different screen sizes

5. The “QTrip” brand, “Home” and “Explore” are all links and should be left aligned while the other two are to be right aligned

6. Set href of :

- “QTrip” brand and “Home” as “#”
- “Explore” as the Adventures Grid page

# Adding the Hero section elements

---

The Hero header is a large banner image that appears prominently on screen load. It includes a welcome message and a search bar.

1. Add the required HTML to the `<div class="hero-image d-flex">` element to produce the required behaviour.
2. **Styling the Hero image:** The Hero section's black background is the top-left of the image, but its full height isn't displayed. Add styles to adjust the Hero image.
3. Update the `.hero-image` class selector in `css/styles.css` to get the styling of the hero image as required

# Create Grid of Cities

## Add Rows and Columns of Cities in a Grid :

- Add city images to the grid by modifying CSS.
- Use Bootstrap row and col classes to create a clickable grid of city names.
- First city redirects to the Adventures Grid page. Use city names from the assets folder.
- Use Bootstrap Grid's row and col-\* classes to create the cities grid structure.
- Add HTML inside the `<div class="container"></div>` element that follows the hero image section in the Landing Page's index.html page.

# Adding effects to the cards

---

## Improving user experience with Interactive Selection Feedback :

1. By darkening the image (i.e., decrease the brightness)
2. By zooming in effect on the image.
3. You'll also find that the transition between hovered and non-hovered states aren't instantaneous but smooth.
4. Add appropriate styles for the tile image in `css/styles.css` to create the darkening and zooming in effect on hovering over the image as given.



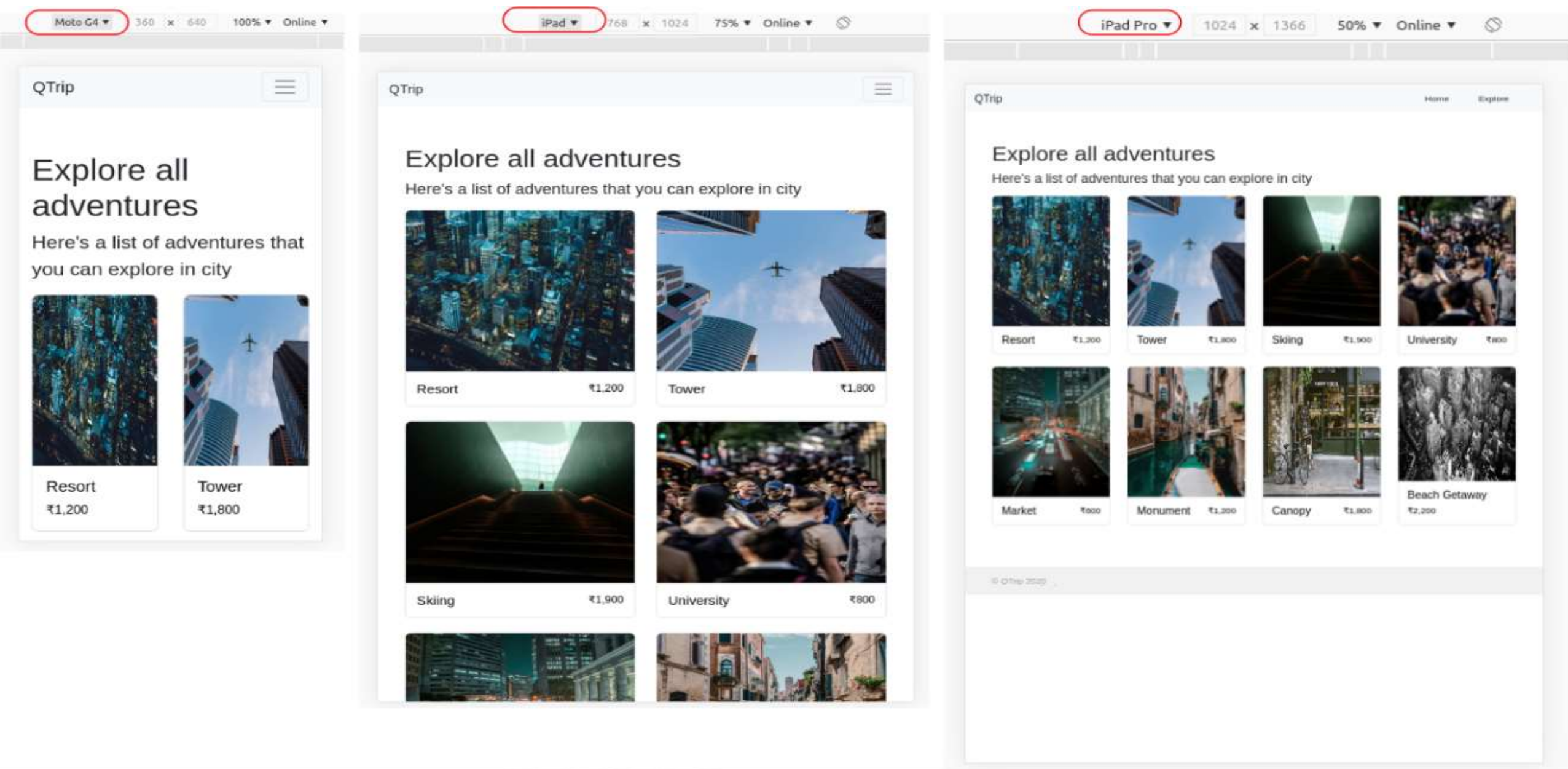
## 2. Implement the adventures page

---

- Created a responsive grid of adventures using Bootstrap's grid
- Made adventure type and price text on the cards responsive using Bootstrap's flex containers
- Added proper spacing between the adventure's grid using Bootstrap spacing shorthands
- Made the images to be responsive using Bootstrap's responsive image classes

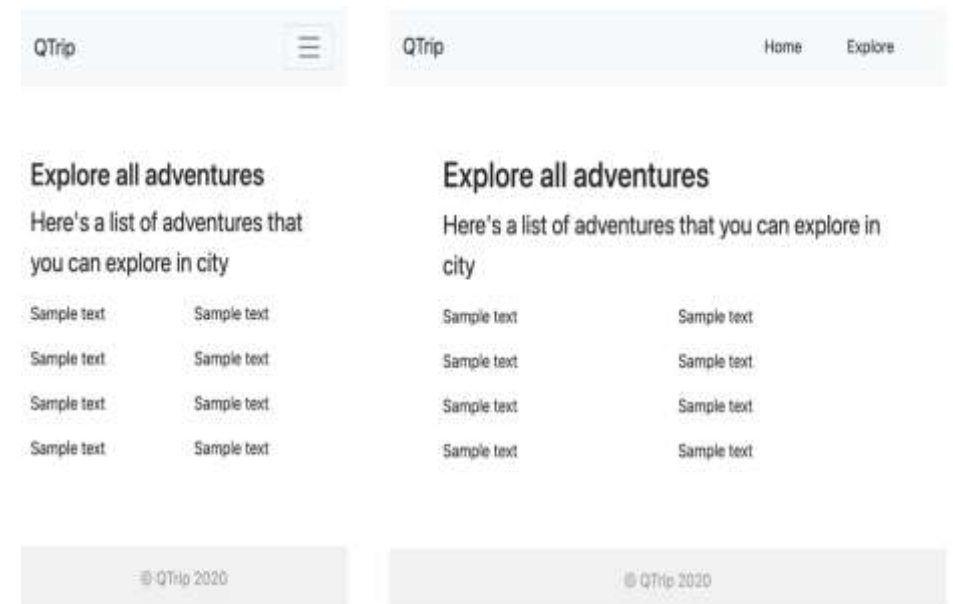


# Adventures page on varying screen sizes



# Create page structure

- Navigation bar and footer are similar to the one used in the Landing page
- The design elements once again strongly suggest making use of a grid ( similar to the grid used in the Landing page)
- Add the HTML to the pages/adventures/index.html file and any CSS to the css/styles.css file to match the Adventures Grid page view as given. The components to implement are
  - a. The navigation bar
  - b. Header texts
  - c. Adventures Grid structure with dummy text
- Clicking on the “Home” link on the navbar should redirect to the Landing page



# Create the adventures grid page

---

Implement the layout for the first card as given:

- Use `<img>` tags to add images
- You can utilise the images given in the `assets/adventures` folder
- Clicking anywhere on the card should redirect to the Adventure Details page

**Improve user experience**

- You'll now add an effect on hover to the cards. The image should dim on hover .
- Add the dim effect to the cards

# 3. Add the adventure details page to complete the static web page features

---

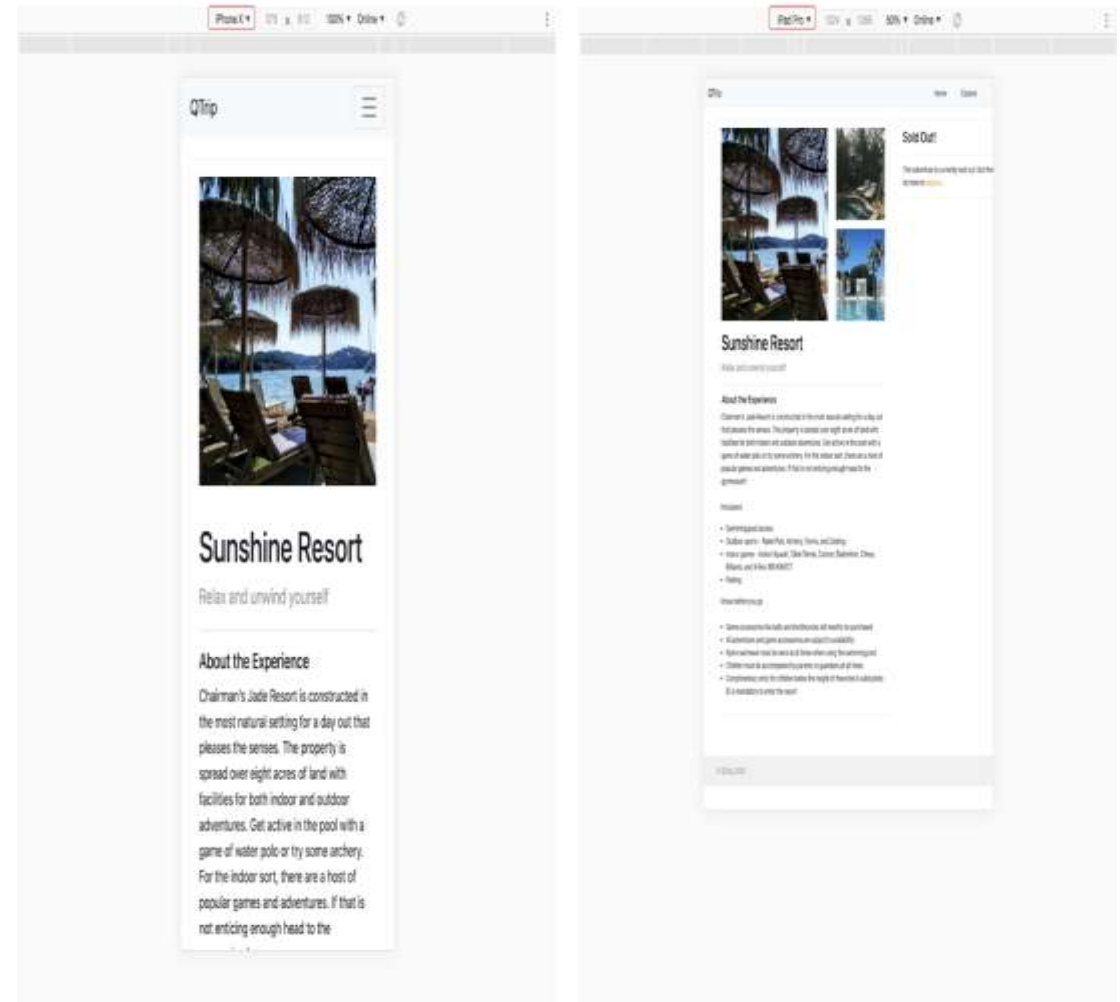
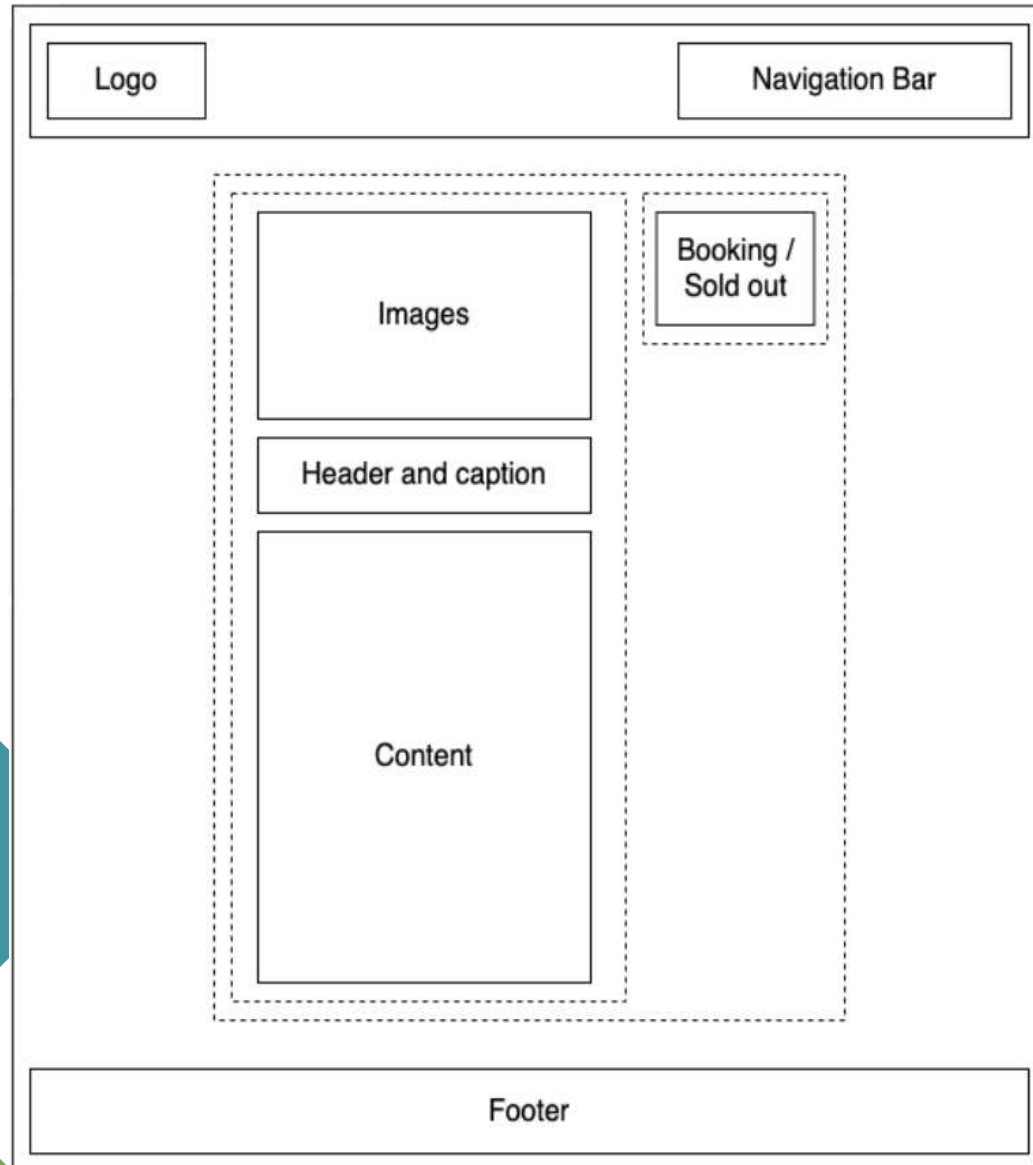
## Scope of work

- Came up with the page's HTML layout from scratch.
- Positioned images of varying sizes and set them to toggle visibility based on screen sizes.
- Created the “Sold out” side section to be fixed on the screen even on scrolling.

## Skills used

- HTML, CSS, Bootstrap, Bootstrap Display, CSS Positioning

# Adventure details page wireframe



Adventure details page on different screen sizes



# Add adventure images

---

- Implement the page layout given to you using Bootstrap Grid. Also, copy HTML for the navigation bar from one of the previous pages
- Plug in the images given at /assets/adventures/resort/ and add any required styles to get the page to look as given. Couple of other requirements are
- The image corners are slightly rounded
- There should be a dim effect on hover for all the images

## Add the header and content sections

- Add the required HTML and CSS to replicate the given page design. The image/content area and sold out sections have a light border with rounded corners

## Add the Sold out banner content

**Sold Out!**

This adventure is currently sold out. But there's a lot more to **explore**.

# 4. Deploy the QTripStatic website

---

- We will be using Vercel to do the deployment.
- Vercel provides the developer tools and cloud infrastructure to build, scale, and secure a faster, more personalized web.
- Vercel link where your QTrip Static website has been deployed:

<https://damuluru-sanjana-me-qtripstatic-mqkefw200.vercel.app/>

# Overview

---

During the course of this project ;

- Created 3 different web pages from Wireframe layout using HTML and CSS.
- Utilized Bootstrap extensively for responsive design.
- Deployed the website to Vercel.

# QTrip Dynamic

---

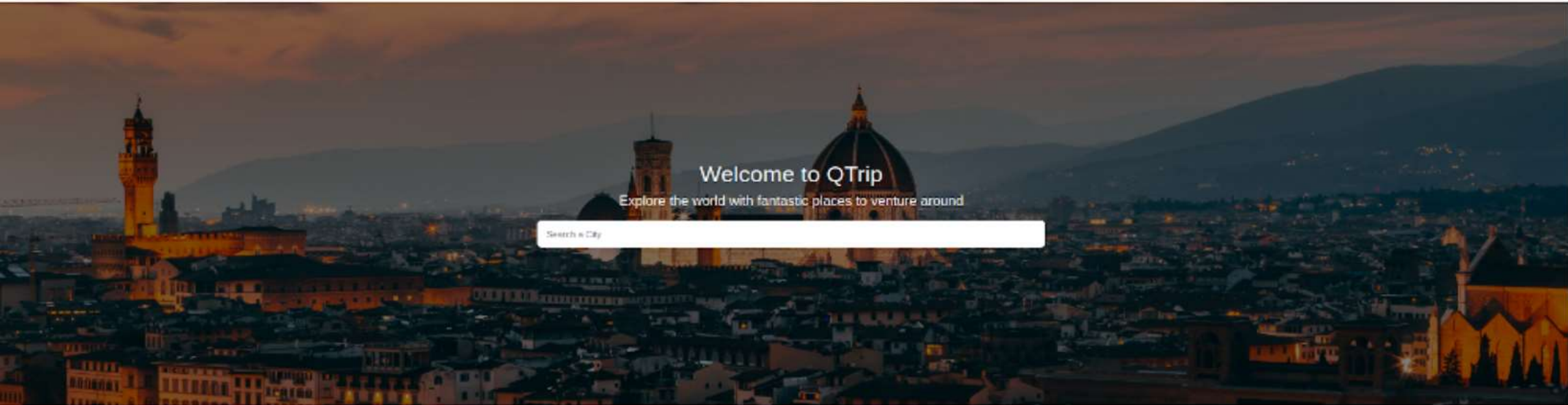
Take QTrip from being a static travel site built using HTML/CSS and make it dynamic using JavaScript.

The site should display cities, adventures and their details, allowing users to make reservations for the adventures.





# Landing Page



Welcome to QTrip

Explore the world with fantastic places to venture around

Search a City



BENGALURU  
100+ PLACES



GOA  
250+ PLACES



KOLKATA  
100+ PLACES



SINGAPORE  
100+ PLACES





# Adventures Page

## Explore all adventures

Here's a list of places that you can explore in city


Filters:

Filter by Duration (Hours) ▾

Clear

Add Category ▾


Clear



Party

Niaboytown ₹4003

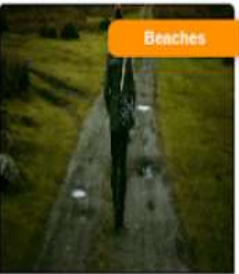
Duration 6 Hours



Cycling

Fort Sionnnn ₹2686


Duration 9 Hours



Beaches

Wooddaux ₹3715


Duration 8 Hours



Cycling

Bageorge With Nonshi Harbour ₹3184


Duration 3 Hours



Hillside

Stonelumhawk ₹4143


Duration 17 Hours



Beaches

La Annmcast ₹3712


Duration 7 Hours



Hillside

Fort Shilbluff ₹795

Duration 19 Hours



Cycling

Shiwood ₹1352

Duration 17 Hours

# Adventure Details Page

QTrip

[Home](#)

[Reservations](#)

## Niaboytown

This is a mind-blowing adventure!



### About the Experience

A random paragraph can also be an excellent way for a writer to tackle writers' block. Writing block can often happen due to being stuck with a current project that the writer is trying to complete. By inserting a completely random paragraph from which to begin, it can take down some of the issues that may have been causing the writers' block in the first place. A random paragraph can also be an excellent way for a writer to tackle writers' block. Writing block can often happen due to being stuck with a current project that the writer is trying to complete. By inserting a completely random paragraph from which to begin, it can take down some of the issues that may have been causing the writers' block in the first place. A random paragraph can also be an excellent way for a writer to tackle writers' block. Writing block can often happen due to being stuck with a current project that the writer is trying to complete. By inserting a completely random paragraph from which to begin, it can take down some of the issues that may have been causing the writers' block in the first place. A random paragraph can also be an excellent way for a writer to tackle writers' block. Writing block can often happen due to being stuck with a current project that the writer is trying to complete. By inserting a completely random paragraph from which to begin, it can take down some of the issues that may have been causing the writers' block in the first place. A random paragraph can also be an excellent way for a writer to tackle writers' block. Writing block can often happen due to being stuck with a current project that the writer is trying to complete. By inserting a completely random paragraph from which to begin, it can take down some of the issues that may have been causing the writers' block in the first place. A random paragraph can also be an excellent way for a writer to tackle writers' block. Writing block can often happen due to being stuck with a current project that the writer is trying to complete. By inserting a completely random paragraph from which to begin, it can take down some of the issues that may have been causing the writers' block in the first place. A random paragraph can also be an excellent way for a writer to tackle writers' block. Writing block can often happen due to being stuck with a current project that the writer is trying to complete. By inserting a completely random paragraph from which to begin, it can take down some of the issues that may have been causing the writers' block in the first place.

Name

Pick a Date

Person(s)

₹ 4003 per head

Total

₹ 0

Reserve

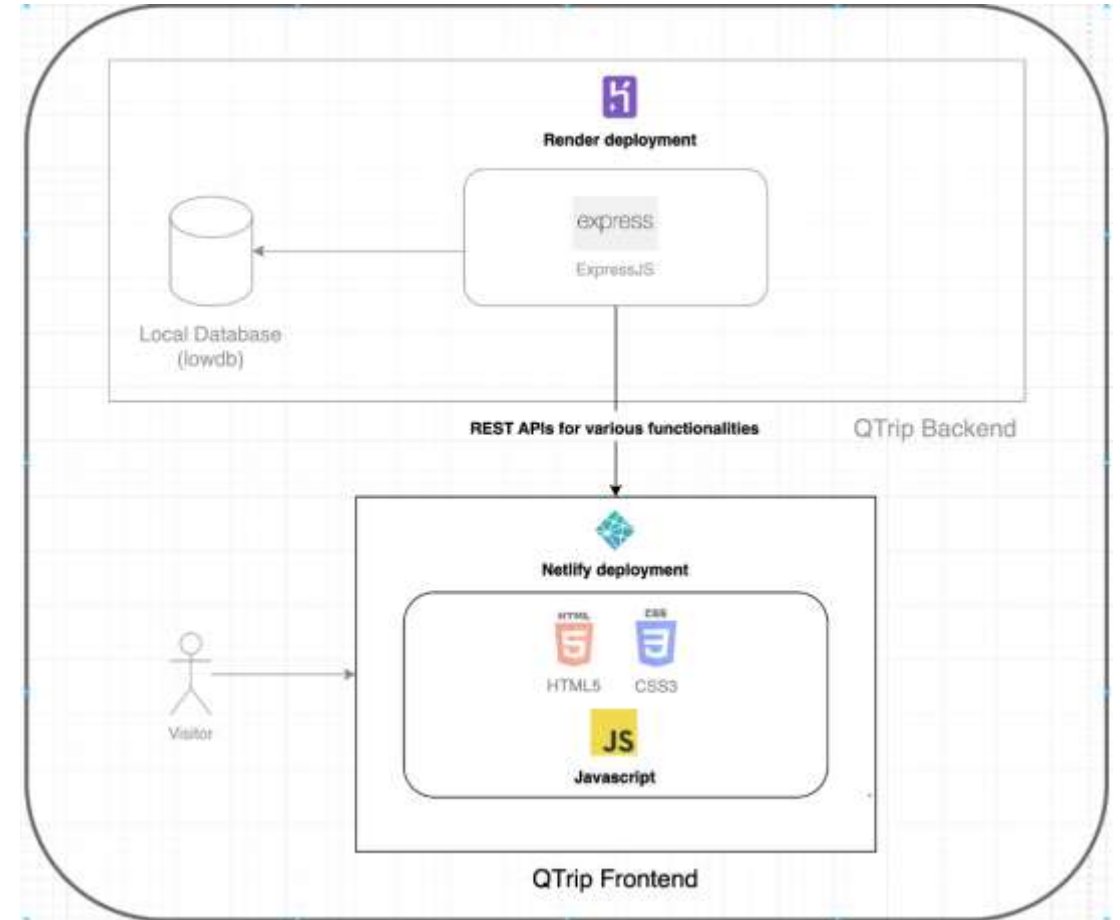
# Reservations Page

## Your Reservations

Transaction ID	Booking Name	Adventure	Person(s)	Date	Price	Booking Time	Action
ecc53b5043fe6116	Kir	Fort Sion	1	8/11/2020	2686	6 November 2020, 2:53:34 pm	<a href="#">Visit Adventure</a>
6c1b9502b8169ed0	New User	Nesbridge	1	11/11/2020	3316	9 November 2020, 9:54:54 am	<a href="#">Visit Adventure</a>

# QTrip Architecture

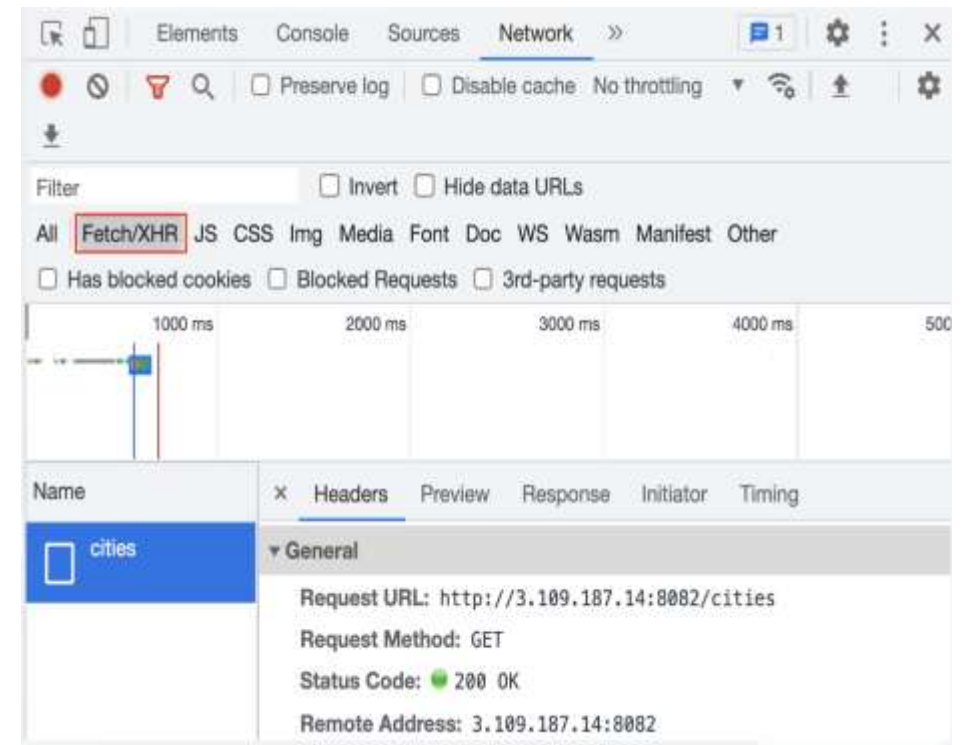
- From the architecture diagram , we can see that there is a Frontend and a Backend component powering the QTrip site.
- The focus of this ME will be the JavaScript content for the Frontend which will use REST API to fetch the required data from the Backend
- It will also send data that needs to be stored to the Backend, using REST API. Without the Backend component, the QTrip site would only have static pages which cannot fetch or store data.
- You will also deploy:
  - Frontend using Netlify
  - Backend using Renderto make your site publicly accessible.



# 1. Create the city cards for QTrip landing page

## Use REST API to fetch city data

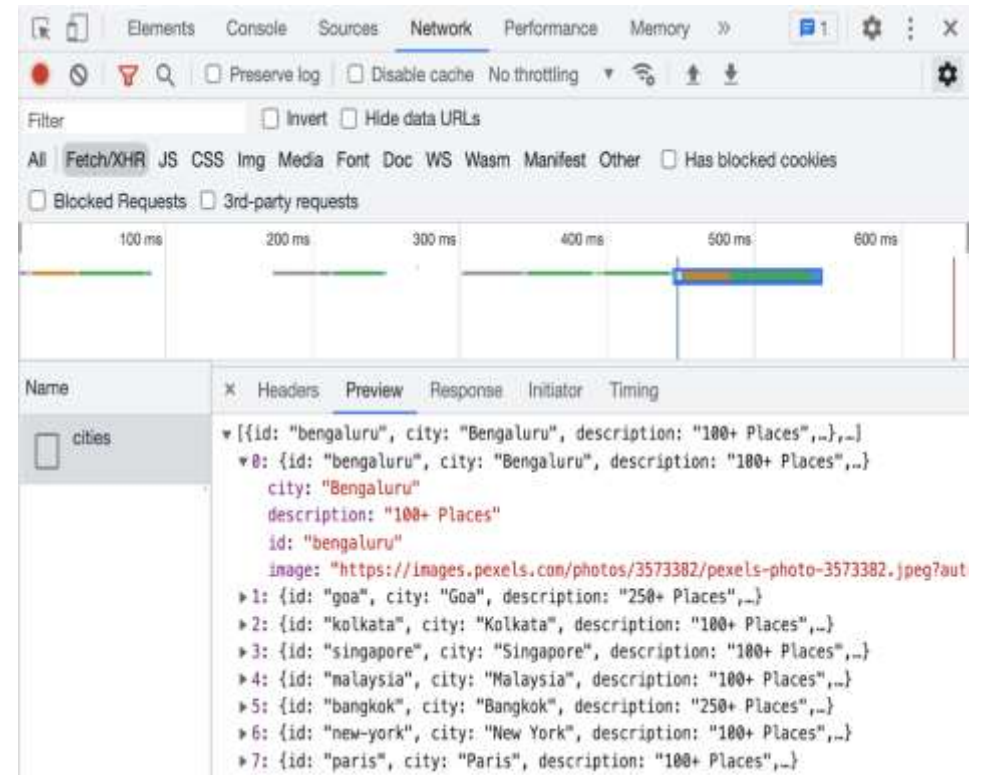
- Start the Backend server : It supports REST APIs which can be invoked by the frontend to GET or POST data.
- Fetch Cities data from backend : Implement the `fetchCities()` method to make an API call to the “/cities” endpoint and return an array with the cities data
- Log relevant values to the console to verify the correctness of the implementation.
- Use the Network tab to monitor API call status and check the website for proper functionality.





# Insert City Cards into DOM

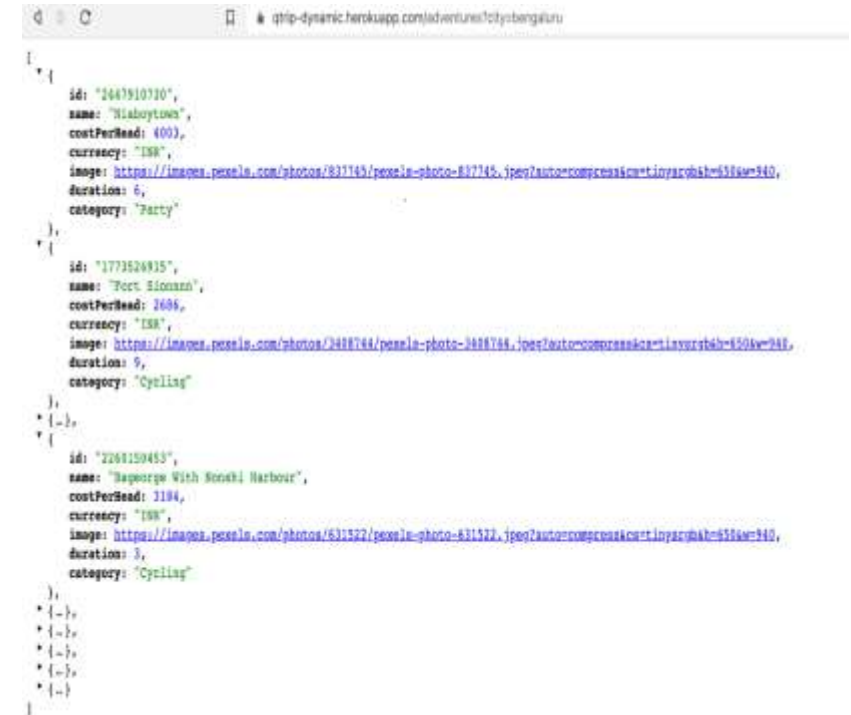
- **Understand and Parse Data:** Inspect the API response to identify fields (id, city, description, image) and use `init()` to process this data, calling `addCityToDOM()` for each city.
- **Implement and Insert City Cards:** Complete the `addCityToDOM()` function to create responsive City cards and insert them into the HTML element with id data, ensuring each card links to `pages/adventures/?city=<id>`.
- **Verify Implementation:** Run `npm test` in `frontend/__tests__` to check the correctness of `addCityToDOM()` and address any issues based on test results.



## 2. Create the cards for adventures grid page

### Extract parameter from URL :

- **Implement and Test `getCityFromURL()`:** Create the `getCityFromURL(search)` function in `adventures_page.js` to return the city query parameter, and verify its correctness by running `npm test` in the `frontend/__tests__` folder.
- **Use REST API to fetch adventures:** Modify the function to call `/adventures?city=<city_name>`, handle exceptions with `try-catch`, and return the JSON data. Test it using `npm test` and check responses with `curl` or `Chrome Dev Tools`.
- **Iterate over Adventures and insert into HTML DOM:** Create and style adventure cards from the list, link each card to `detail/?adventure=<adventure_id>`, and test with `npm test`.



```
{
  "id": "2447910730",
  "name": "Nisiboytown",
  "costPerHead": 400,
  "currency": "INR",
  "image": "https://images.pexels.com/photos/837745/pexels-photo-837745.jpeg?auto=compress&cs=tinysrgb&h=531&w=940",
  "duration": 6,
  "category": "Party"
},
{
  "id": "1773524935",
  "name": "Port Bionn",
  "costPerHead": 2686,
  "currency": "INR",
  "image": "https://images.pexels.com/photos/2488744/pexels-photo-2488744.jpeg?auto=compress&cs=tinysrgb&h=531&w=940",
  "duration": 9,
  "category": "Cycling"
},
{
  "id": "2269159455",
  "name": "Bageorge With Sonshi Harbour",
  "costPerHead": 2184,
  "currency": "INR",
  "image": "https://images.pexels.com/photos/631522/pexels-photo-631522.jpeg?auto=compress&cs=tinysrgb&h=531&w=940",
  "duration": 3,
  "category": "Cycling"
}
]
```

# 3. Create filters for adventures

---

**Filter by Category** : Complete `filterByCategory()`, `filterFunction()`, and `generateFilterPillsAndUpdateDOM()` to filter adventures, update filter pills in the DOM, and handle filter states. Test with provided test cases to ensure correct functionality.

**Filter by Duration** : Complete `filterByDuration()` and update `filterFunction()` to handle duration filters. Implement `selectDuration()` and `clearDuration()` in `index.html` to manage filter application and clearing. Test the functionality to ensure correct filtering and DOM updates.

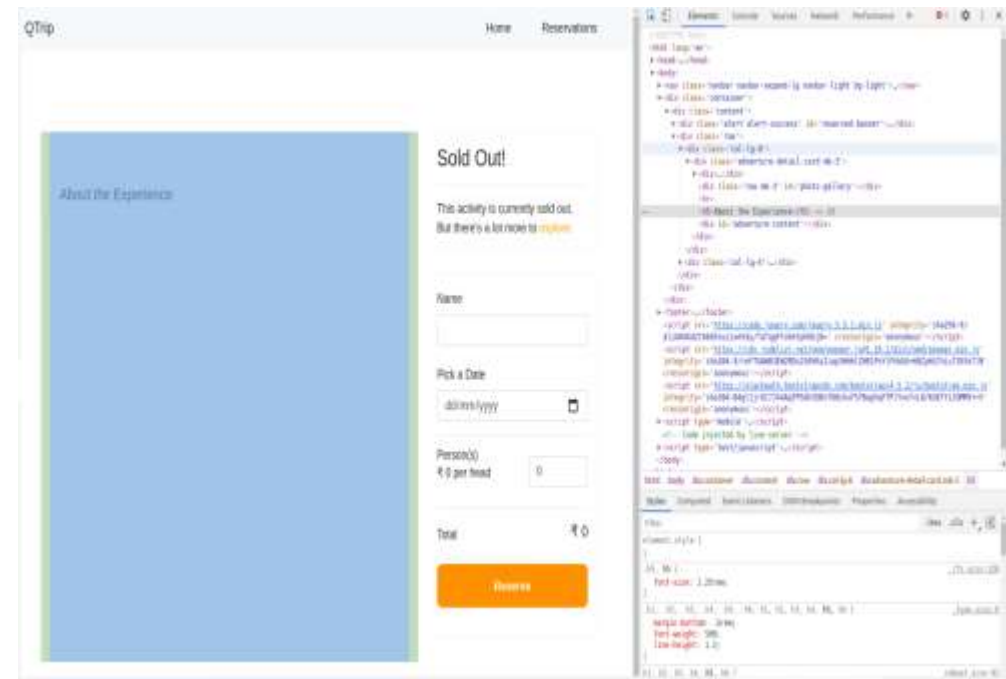
**Data Persistence** : Complete `saveFiltersToLocalStorage()` and `getFiltersFromLocalStorage()` to persist and retrieve filter selections using local storage. Ensure filters are updated on page load and test by refreshing or reopening the page to confirm persistence.

# 4. Create the adventure details page

**Extract parameter from URL and invoke API:** Implement `getAdventureIdFromURL()` to extract the adventure ID from the URL, then update `fetchAdventureDetails()` to call the backend API and handle exceptions. Finally, test with `npm test` and ensure the tests pass.

**Insert Adventure Details into the DOM:** Implement `addAdventureDetailsToDOM()` to populate the adventure details in the DOM and test with `npm test`.

**Add bootstrap photo gallery:** Update `addBootstrapPhotoGallery()` to use a Bootstrap carousel for the photo gallery, ensuring only one image is active at a time. Overwrite the previous gallery content while keeping `addAdventureDetailsToDOM()` intact. Run `npm test` to verify all test cases pass.



# 5. Add support for adventure reservations

---

**Conditional rendering and updates to reservation form:** Update `conditionalRenderingOfReservationPanel()` to show the correct reservation section and `calculateReservationCostAndUpdateDOM()` to display the total cost.

**Make a reservation:** Implement `captureFormSubmit()` to POST form data and handle alerts, and `showBannerIfAlreadyReserved()` to display a banner if the adventure is reserved.

**Show the reservation listing:** Implement `fetchReservations()` to retrieve reservation data via a REST API call and handle exceptions. Then, use `addReservationToTable()` to display reservations in a table, conditionally rendering elements based on availability and formatting date fields correctly. Test your implementation with `npm test` to ensure correctness.

# 6. Deploy the QTrip website using render and netlify

## Backend deployment using Render :

- **Sign in to Render**
- **Fork Repository**
- **Create Web Service**
- **Connect GitHub:**
- **Configure Deployment:** Name the service qtrip-dynamic, set "Root Directory" to backend, and "Start Command" to npm start.
- **Deploy:** Click "Create Web Service" and wait for the deployment logs.
- **API Access:** Use the provided URL to access your deployed API endpoints.

## Frontend deployment using Vercel :

- **Configure Backend URL:** Update frontend/conf/index.js to point to the backend URL
- **Sign Up on Vercel**
- **Open Terminal:** Navigate to the frontend directory in your project.
- **Run Vercel Login:** Execute vercel login and authenticate via email.
- **Run Vercel Command:** In the terminal, execute vercel for deployment setup.
- **Setup Deployment and Get Deployment Link**



# Project Links

---

## QTrip Static

Vercel link where QTrip frontend : <https://damuluru-sanjana-me-qtripstatic-mqkefw200.vercel.app/>

## QTrip Dynamic

Render link where QTrip backend has been deployed : <https://damuluru-sanjana-me-qtripdynamic.onrender.com>

Vercel link where QTrip frontend has been deployed : <https://frontend-adnnwlp1-lakshmi-sanjana-ds-projects.vercel.app/>



# Thank you

---

**Lakshmi Sanjana D**

**HU21CSEN0101118**

**ldamulur@gitam.in**