**KAUNAS INFORMATION TECHNOLOGY SCHOOL**

**THE INTERNATIONAL PROFESSIONAL MASTERY CONTEST**

**„WEB <dev/> challenge 2023“**

**Module F**

**Frontend Development**

After completing the REST API, you are now tasked to implement the frontend that consumes this API. Users shall have these use cases available, provided by you within the form of a single page (web) app:

* Discover rooms with filters
* Reserve rooms
* Receive the reservation by entering their details into a form
* Cancel the reservation and make the rooms available again

A backend implementation is provided to you in a finished state. It doesn't matter if you didn't finish the phase one or if you had bugs. The score evaluation of phase two will be done partly with mocked responses or with the provided API, not with your phase one solution.

Throughout this specification you will find mock screens. These are given as a general guide but are not to be followed to the pixel. You are expected to come up with your own colors and design ideas and implement those.

**Competitor Information**

The final website has to be available at https://<hostname>/module-f.

The source files of the website have to be at module-f/src.

**Non-Functional Requirements**

* You are free to use the frontend technology of your choice.
* Implement a SPA - Single Page Application.
* The application should run successfully in the versions of Google Chrome and Mozilla Firefox installed on your machine.
* The amount of traffic with the API is to be reduced to a minimum. Rooms are not expected to update frequently (leverage state stores, state services, etc.).

**Page Structure**

The basic structure of each page should contain:

* Header: Visual bar that includes the App Title: " WEB <dev/> challenge 2023 Hotel reservation system".
* Page Title: A title after the header.
* More info button: A button to get all the information about rooms and make a reservation.

**Selectors**

* App Title: Contains text "Hotel Rooms"
* Reservation Hint Text: Contains text "Already reserved?"

**Pages**

**Landing Page - Discover rooms**

The landing page is the first page that is shown when visiting https://<hostname>/module-f

The main feature of this page is that it displays all rooms and allows users to discover what is on offer. The list of rooms is responsive. When scaling the browser between 320px and 1330px, everyhing has to work well.

The card must display the following information:

* Room name
* Capacity
* Wifi availability
* Parking availability
* Image

Graphical user interface, application

Description automatically generated

**Filter Functionality**

The user can filter the list of rooms with the following filters:

* From selector: The user can select the checkin date in a popup calendar.
* To selector: The user can select the checkout date in a popup calendar.

Additional requirements:

* Selected filters are active immediately. The user does not have to click "Apply", "Filter" or similar.
* There is a "clear" button that resets all filters. It only appears if the filter is active.
* When there are no rooms to display, because the filter filtered everything out, a text appears that there are no rooms that match the current filter criteria: "No rooms are matching the current filter criteria."

**API requirements**

The API returns rooms at one endpoint. Once the user opens the application, they should be presented with all rooms. Once they select a checkin and checkout in the date selector, the application loads filtered rooms.

**Selectors**

* Filter:
  + From Date filter: Selector input[type=date]
  + To Date filter: Selector input[type=date]
  + Clear filter button: Selector button with Text contained "Clear"
  + "No rooms"-text because the filter is too strict: Text contained: "No rooms are matching the current filter criteria."
* Room Cards
  + Card: CSS selector .card
  + Date: Date as dd/mm/yyyy within card
  + Room name: Room name within card
  + Capacity: Capacity within card
  + Wifi: Wifi within card as a icon from font awesome icon library
  + Parking: Parking within card as a icon from font awesome icon library
  + Image: Image of room within card

**Reservation page**

Once the user has selected a room that they want to reserve for, they can click on the More info button and land on the room reservation page.

A screenshot of a computer

Description automatically generated with low confidence

On the page, user first clicks reserve button. Once they are happy, they proceed to the reserve form where they enter the following fields:

* Name
* Address
* City
* Zip code
* Country
* From date
* To date

Graphical user interface, application

Description automatically generated

**Selectors**

* Card: The same attributes are expected to be present as on the landing page within .card
* Reservation Form
  + Form Title: Contains text "Please enter your details"
  + Name Field Label: Contains text "Name"
  + Address Field Label: Contains text "Address"
  + From Field Label: Selector input[type=date]
  + To Field Label: Selector input[type=date]
  + ZIP Code Field Label: Contains text "ZIP Code"
  + City Field Label: Contains text "City"
  + Invalid fields: CSS selector .validated :invalid
  + Reserve Button: CSS Selector button and contains text "Reserve"
  + Change room Button: CSS Selector button and contains text "Change room"

**Reservation list page**

The reservations list page is shown immediately after reservation or if the user wants to retrieve them later.

Graphical user interface, website

Description automatically generated

**Reservation retrieval**

The user can reach the reservation list page at a later stage, not only after reservation. When the user clicks on the button to reservation list, they are presented with a form where they enter the following fields:

* name
* reservation code

Graphical user interface

Description automatically generated

The form is validated with all fields required. If the form is not valid, the specific violation is shown.

After the user submits the form, the application checks if there are reservations matching the input. If there are no reservations available, it shows a message with the text "Could not find reservations with these details." on the same page as the form. If there are reservations found with the given details, the reservations are shown and the form disappears.

**Reservations Display**

On the reservations list page the following details are shown:

* Your Details:
  + Name
  + Reservation date in format dd/mm/yyyy
* For each reservation:
  + Code
  + Room number
  + Check in & Check out date

The user can cancel a single reservation with a button. Before the reservation is cancelled, the user must confirm to cancel it with the built-in Browser confirm dialog confirm('...'). After a successful cancellation, the reservation disappears. If the last reservation was cancelled, the user is routed to the landing page.

**Selectors**

* Reservation Form
  + Page Title: Contains text "Retrieve your reservation."
  + Name Label input: CSS selector label and Contains text "Name"
  + Code Label input: CSS selector label and Contains text "Code"
  + Invalid fields: CSS selector .validated :invalid
  + Submit button: CSS selector button[type=submit]
* Reservation Display
  + Page Title: Contains text "Your reservation list!"
  + Reservation Details:
    - Title: Contains text "Your Details"
    - Name Label: Contains text "Name"
    - Reserved on Label: Contains text "Reserved On"
  + Each reservation container: CSS selector .reservation
  + Reservation Code: Contains text "Code: {code}"
  + Check in: Contains text "Row: Check in"
  + Check out: Contains text "Check out
  + Cancel button: CSS selector button and contains text "Cancel reservation"