SOFE 2720 Principles of Software and Requirements Project

Section: 5, CRN: 73612

Thursday, April. 3rd 2025

Name	**Student Number**
Jacob Coelho	100909899
Jessica Broda (MIA)	100819562 (MIA)
Gabriel Mclachlan	100919944

Calendar Application

1) Project Requirements

- ❖ HTML:
  - ➢ Includes headers for days of the week.
  - ➢ Rows for weeks are structured appropriately.
  - ➢ Navigation buttons for previous/next months.
  - ➢ Interactive date elements for selection.
  - ➢ Event Modal: Contains a pop-up/modal for booking appointments.
  - ➢ Code includes comments to explain key sections.
- ❖ CSS:
  - ➢ Sets colors, fonts, and spacing for a clean and modern look.
  - ➢ Uses a grid layout to structure days and dates.
  - ➢ Adds hover effects and active states for date selection.
  - ➢ Different colors for booked, selected, and highlighted dates.
  - ➢ Uses visual indicators like background color changes.
  - ➢ Styles the modal for entering booking details.
  - ➢ Positions and designs the event list for displaying booked events.
  - ➢ Styles the previous/next month buttons.
  - ➢ Includes a clock element for real-time display.
  - ➢ Code includes comments to explain key sections.
- ❖ JavaScript:
  - ➢ Implements navigation between months.

- ➢ Updates the calendar dynamically when switching months.
- ➢ Adds event listeners to each date.
- ➢ Opens a modal when a date is clicked for inputting a reason.
- ➢ Saves booked data and marks the date as booked.
- ➢ Uses local storage to save and retrieve booked dates and reasons.
- ➢ Loads saved data when the page reloads.
- ➢ Displays booked dates visually on the calendar.
- ➢ Shows a list of booked events dynamically.
- ➢ Code includes comments to explain key sections.

2) Use Cases

- **Use Case Example 1:**
  - **Name**: Timed Reminders.
  - **Brief Description**: Allow the user to pick a time in the day or a certain amount of time before an event for the app to send them a reminder of the set event.
  - **Actors**: User & Device.
  - **Preconditions**: Users will be given options on set times/days for when to be reminded or can set one themselves.
  - **Basic Flow**: 1. Users will create new events, 2. While inputting event details, an option for a reminder will be at the bottom, 3. When the user clicks on it, it will show times/days for when they can be reminded, 4. Once the event is saved, the device sends a notification on the specified time/day.
  - **Alternate Flow**: 1. Users will create new events, 2. While inputting event details, an option for a reminder will be at the bottom, 3. When the user clicks on it, it will have slider options for: month, day, hour, minute, and

time of day, 4. Once the options are imputed, the app will ask to save it and the user can reuse it for future events, 5. Once the event is saved, the device sends a notification at the time that was imputed.

- **Exception Flows**: 1. Users will create new events, 2. While inputting event details, an option for a reminder will be at the bottom, 3. Once the user picks a time/day or inputs a specific day, if there is already an event with that same reminder the app will tell the user to pick a different time and won't save the time/day they picked/inputted.
- **Post Conditions**: The app must be able to have selectable times and allow the user to input their own, and it must ask the user to allow notifications to send them to the device and alert the user at their selected time/day.

- **<u>Use Case Example 2:</u>**
  - **Name**: Device Clock Sync.
  - **Brief Description**: The app should be synchronized with the device's clock to allow events to be accurate with when they happen.
  - **Actors**: Supplier & Device.
  - **Preconditions**: The app will automatically sync with the device date and show what month and day it currently is along with the time.
  - **Basic Flow**: 1. User's time settings are set automatically via the internet, 2. When the app is opened, it'll show what month the user is in, along with what date it is, 3. Users can then create an event on the current day or a day on a future date.
  - **Alternate Flow**: 1. User's time settings are not set via the internet, 2. When the app is opened, it'll ask the user to set the time automatically via the internet, 3.The app will send the user to their device's time settings, 4. Once it's set automatically, when they return to the app they can add events like normal.
  - **Exception Flows**: 1. User's time settings are not set via the internet, 2. When the app is opened, it'll ask the user to set the time automatically via the internet, 3. If the user ignores the message, the app will tell them that any events made will possibly not save correctly, 4.The app will take events as normal but if the user returns, there's no guarantee that they will save.

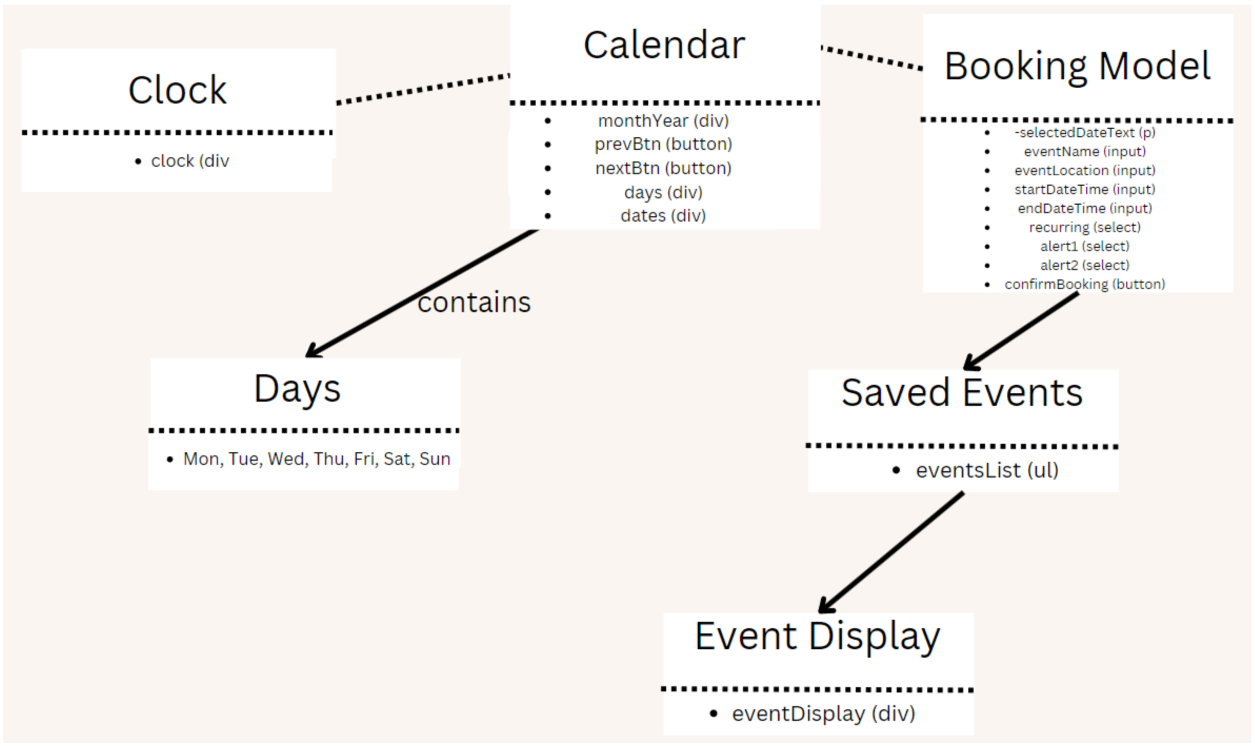
- **Post Conditions**: When the app is opened, it'll show the current month and date along with the time depending on what the device time is. If the device's time setting's aren't set via the internet, the app will ask the user to change it to that and send them to their device's time settings.

- **Use Case Example 3:**
  - **Name**: Repeat Events.
  - **Brief Description**: When the user is creating a new event, an option will be available that will allow that event to be repeated either daily, weekly, bi-weekly, monthly, or customly depending on the user's wishes.
  - **Actors**: User
  - **Preconditions**: The app will copy the event selected to be repeated with the same details on the days it should repeat. The user will also be able to select each one and edit them individually or even remove certain repeats and have the option to do the same for either only the selected event or every future event.
  - **Basic Flow**: 1. Users will create new event, 2. While inputting event details, an option for that event to repeat will be presented, 3. The user can then select daily, weekly, bi-weekly, monthly, etc, options, 4. Once that event is saved, the user can go into future months and see that copied event, 5. The user can then edit any future event and change any details, 6. The user can also choose if the edited details can affect only that copy or every future copy.
  - **Alternate Flow**: 1. Users will create new event, 2. While inputting event details, an option for that event to repeat will be presented, 3. The user can then select a custom event repeater that can allow them to change how daily that event occurs or the certain days of the week, and certain days of each month it can occur, 4. Once that event is saved, the user will see that event created on those specific days they selected, 5. The user can then edit any future event, change any details, and the custom repeating order, 6. The user can also choose if the edited details can affect only that copy or every future copy.
  - **Exception Flows**:1. Users will create new event, 2. While inputting event details, an option for that event to repeat will be presented, 3. The user can then select predetermined repeaters or make custom event repeaters, 4. If the repeated event overlaps with an already existing event the copy won't appear on that date.

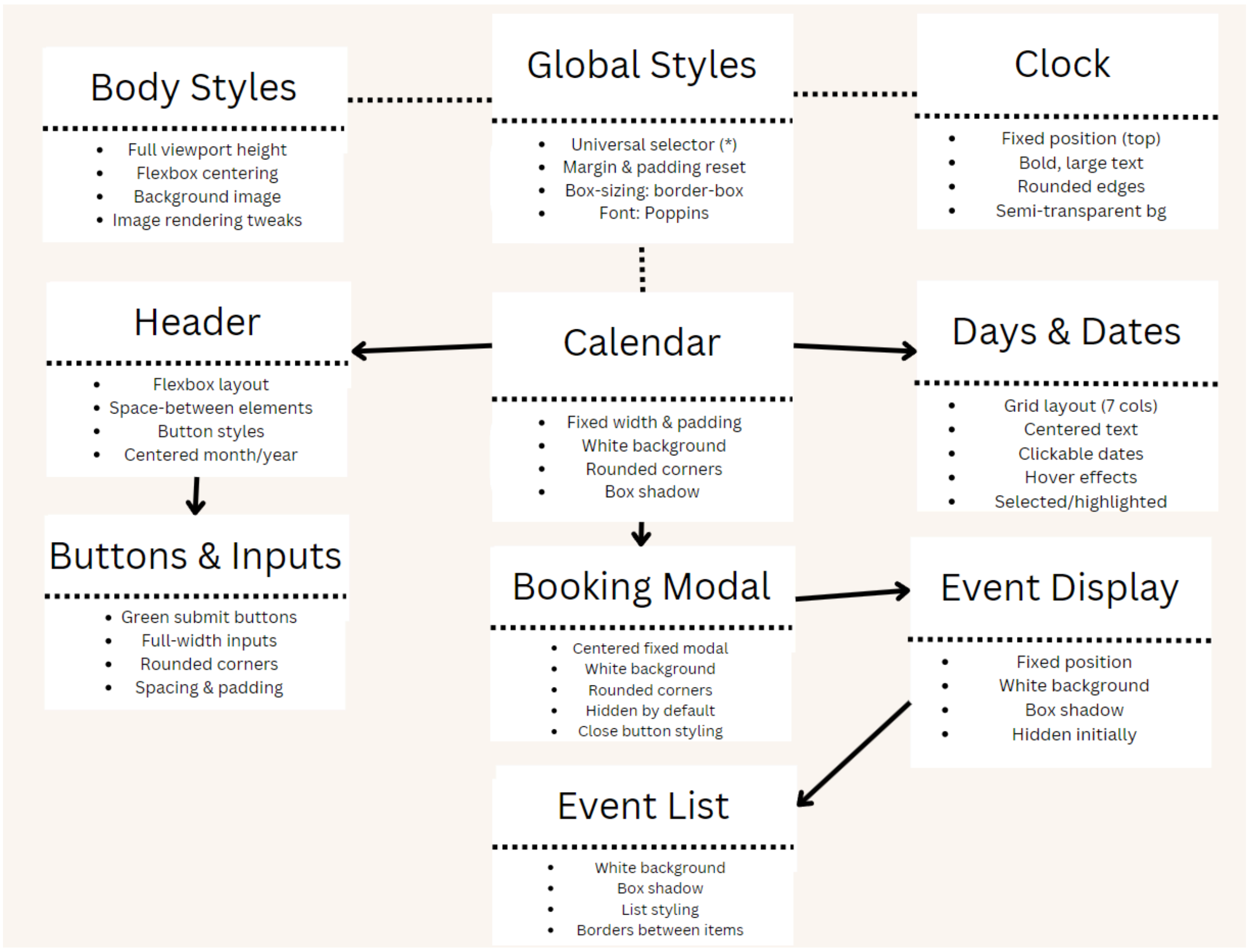
- **Post Conditions**: The app will be able to copy all of the selected event's details and create identical copies on the selected repeater option, it can also let the user change any detail from any copy and make it affect future ones or just the current event.

3) UML Designs

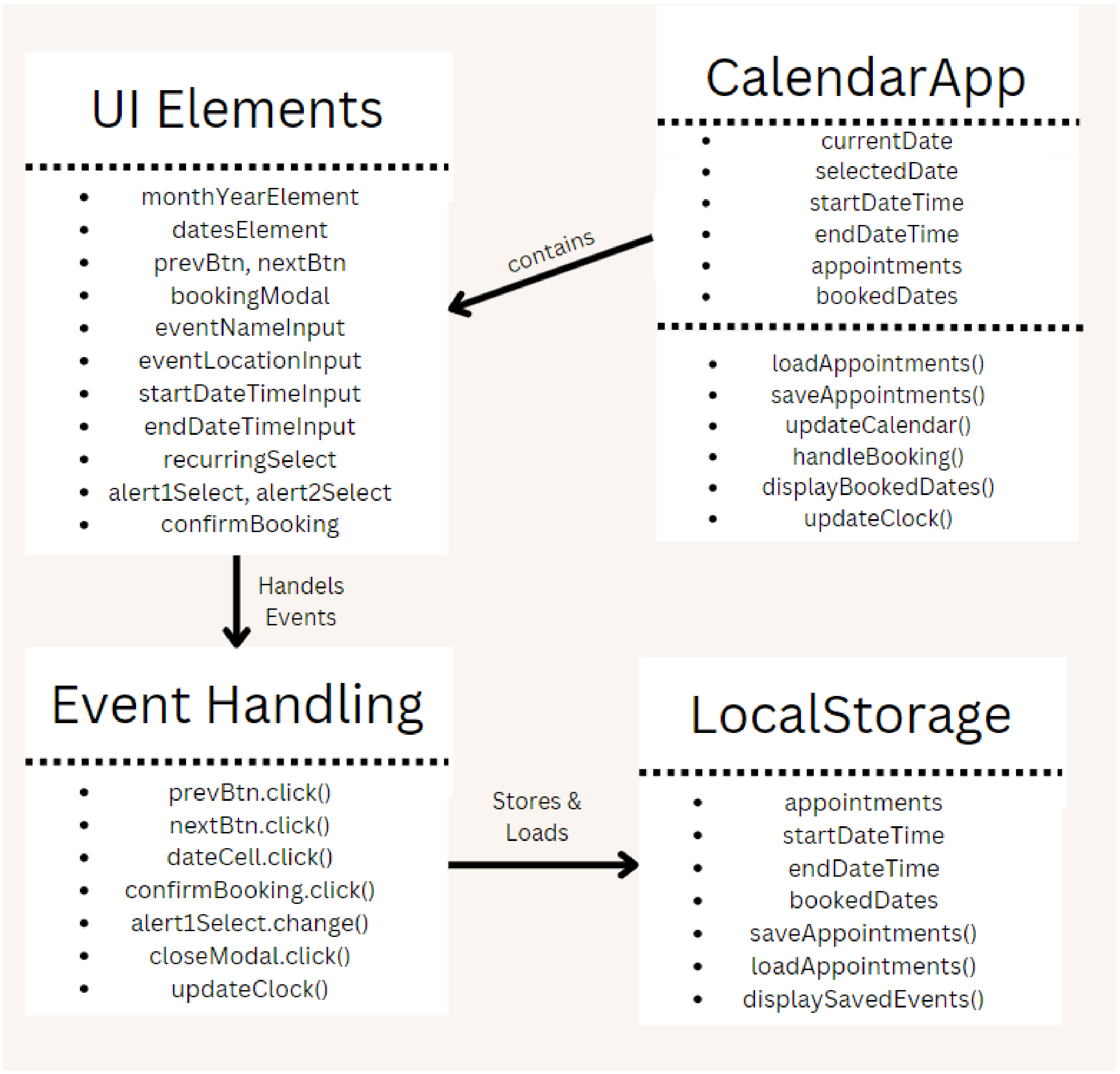
- Index.html



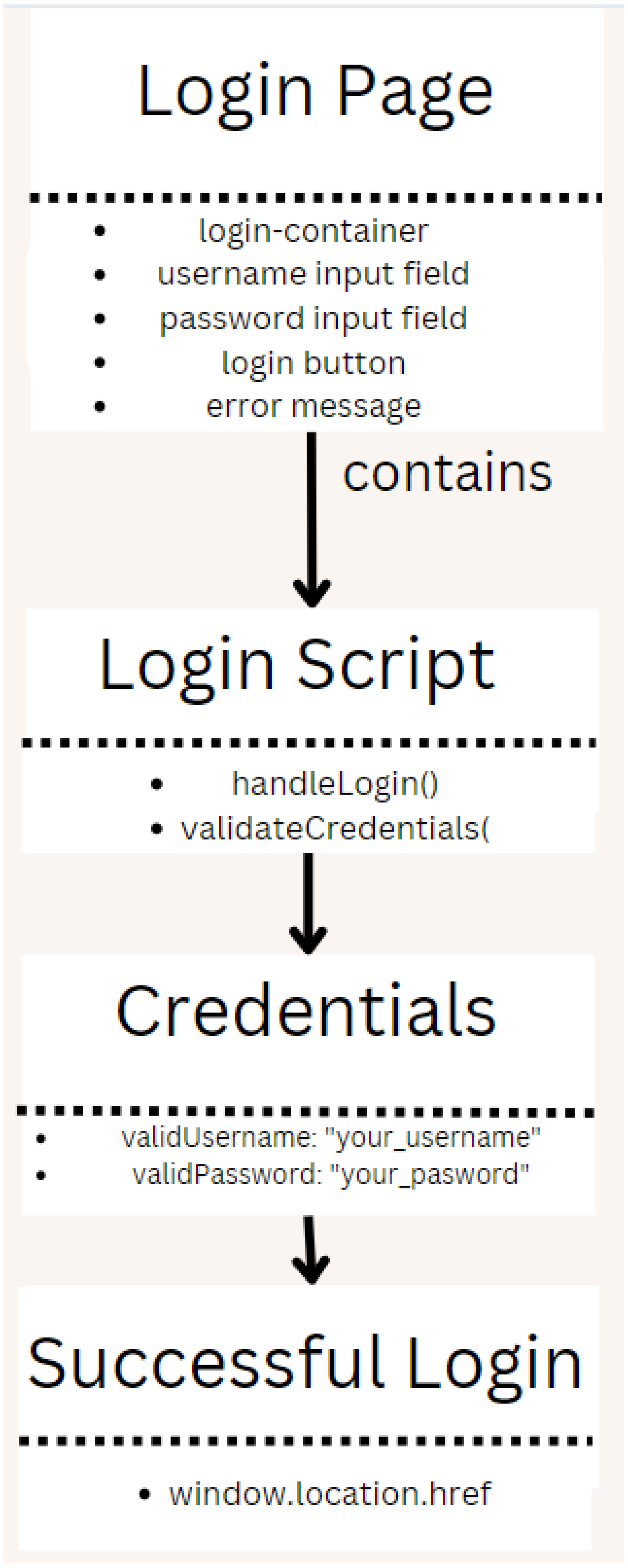
- Style.css


Body Styles
Full viewport height
Flexbox centering
Background image
Image rendering tweaks
Global Styles
Universal selector (*)
Margin & padding reset
Box-sizing: border-box
Font: Poppins
Clock
Fixed position (top)
Bold, large text
Rounded edges
Semi-transparent bg
Header
Flexbox layout
Space-between elements
Button styles
Centered month/year
Calendar
Fixed width & padding
White background
Rounded corners
Box shadow
Days & Dates
Grid layout (7 cols)
Centered text
Clickable dates
Hover effects
Selected/highlighted
Buttons & Inputs
Green submit buttons
Full-width inputs
Rounded corners
Spacing & padding
Booking Modal
Centered fixed modal
White background
Rounded corners
Hidden by default
Close button styling
Event Display
Fixed position
White background
Box shadow
Hidden initially
Event List
White background
Box shadow
List styling
Borders between items

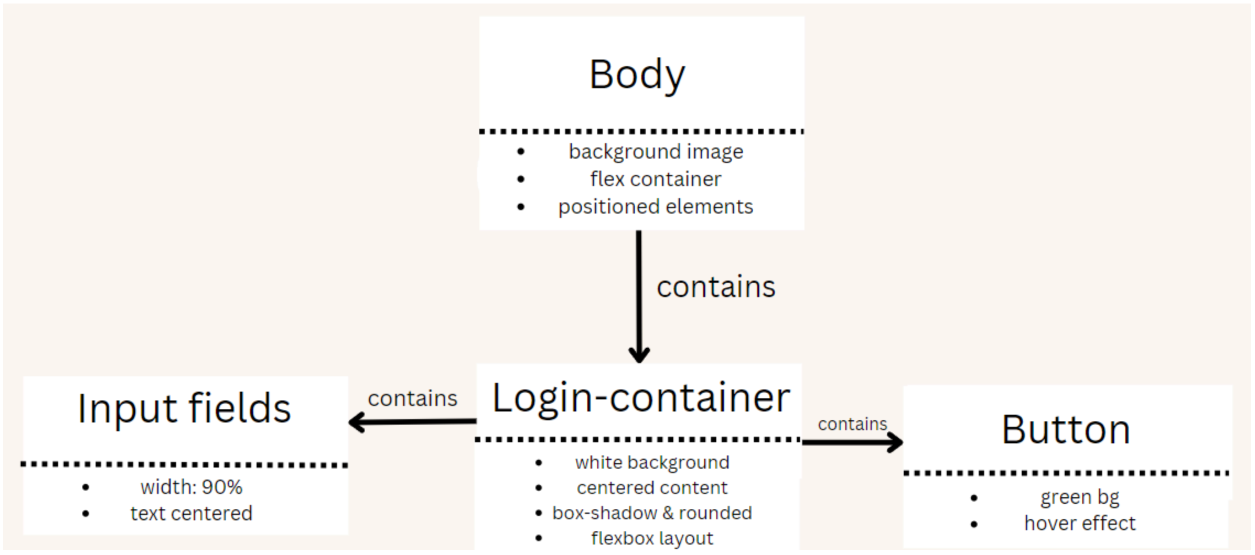
- Main.js


UI Elements
monthYearElement
datesElement
prevBtn, nextBtn
bookingModal
eventNameInput
eventLocationInput
startDateTimeInput
endDateTimeInput
recurringSelect
alert1Select, alert2Select
confirmBooking
CalendarApp
currentDate
selectedDate
startDateTime
endDateTime
appointments
bookedDates
loadAppointments()
saveAppointments()
updateCalendar()
handleBooking()
displayBookedDates()
updateClock()
contains
Handels Events
Event Handling
prevBtn.click()
nextBtn.click()
dateCell.click()
confirmBooking.click()
alert1Select.change()
closeModal.click()
updateClock()
Stores & Loads
LocalStorage
appointments
startDateTime
endDateTime
bookedDates
saveAppointments()
loadAppointments()
displaySavedEvents()

- Login.html


Login Page
login-container
username input field
password input field
login button
error message
contains
Login Script
handleLogin()
validateCredentials(
Credentials
validUsername: "your_username"
validPassword: "your_pasword"
Successful Login
window.location.href

- Styles.css



4) Test Cases

- Test Case ID: BDC_01
    - Title: Verify that a user can book an appointment on a selected date.
- Requirements Fulfilling:
    - Handle Date Selection: Add event listeners to individual date cells.
    - When a date is clicked, highlight it and prompt the user for a reason.
- Preconditions:
    - The user has the calendar application open in a browser.
    - No appointments are currently booked for the selected date.
- Test Steps:
    1) Open the calendar application.
    2) Click on any available date in the calendar.

3) A pop-up appears prompting the user to enter a reason for the appointment.
4) Enter a reason
5) Click "Confirm"
6) The modal should close, and the selected date should be visually marked as booked.
7) Refresh the page.
8) The previously booked date should still appear as booked.

- Expected Result:
  - The selected date should be marked as booked with a different color.
  - The booking should persist even after a page refresh (using local storage)
- Actual Result:
  - The User clicks on a date, enters the reason
  - The date is highlighted in a different color and the event is saved in the local storage
  - Page refreshes and event stays saved
- Status
  - Pass: The program was able to received event information, store it and display on the calendar which day it was on and the reason

- Test Case ID: EDE_02
  - Title: Verify that a user can add more details to events.
- Requirements Fulfilling:

- Book Appointments: Allow users to input a reason for booking
- Mark the selected date as booked and display a visual indicator (e.g., a different color).

- Preconditions:
  - The user has the calendar application open in a browser.
- Test Steps:
  1) Open the calendar application.
  2) Click on any available date in the calendar.
  3) A form should appear prompting the user to enter a name, location, start and end date, if it repeats, and alerts, for the event.
  4) User enters all information
  5) Click "Confirm"
  6) The form should close, and a pop-up should be displayed on the top of the screen confirming the information.
  7) Refresh the page.
  8) The previously booked date should still appear as booked.
- Expected Result:
  - The form should look better and give more freedom to what users can save
  - The pop-up should appear letting the user confirm all the details
  - The booking should persist even after a page refresh
  - All the details should be saved in local storage
- Actual Result:
  - The User clicks on a date and the form appears

    - The user enters all of the information and clicks confirm
    - A pop-up appears once the event is confirmed
    - The date is highlighted in a different color and the event is saved in the local storage
    - Page refreshes and event stays saved
- Status
    - Pass: The program was able to show the form to the user, retrieve the details for the event, show the pop-up confirming all of it, store it and display on the calendar which day its on

- Test Case ID: NMY_03
    - Title: Verify that users can navigate between months using the previous and next buttons.
- Requirements Fulfilling:
    - Implement Navigation: Add buttons or links to allow users to navigate to previous or next months.
- Preconditions:
    - The user has the calendar application open in a browser.
    - The current month is displayed in a grid.
- Test Steps:
    1) Open the calendar application.
    2) Click the circle buttons on the top corners of the grid to move to the next or previous month.

3) Observe if the displayed month updates correctly.
4) Click the opposite button to return to the current month.
5) Verify that the calendar restores the correct month view.
6) Repeat steps 2-4 multiple times to check if all months navigate correctly.
7) Make an event to verify date stays booked even after flipping between months

- Expected Result:
  - The form should l0ok better and give more freedom to what users can save
  - Clicking the top right button should update the calendar to the following month.
  - Clicking the top left button should return to the previous month.
  - The month and year should update correctly.
  - No duplicate, missing, or overlapping dates should appear.
  - Events should stay saved and marked even if calendar date is moved
- Actual Result:
  - The User clicks on one of the buttons and the month (and year if needed) changes correctly
  - The user returns to the current month
  - All the pre-made events are still saved and still appear correctly on the dates they were made on
- Status
  - Pass: The program is able to advance to future months and years as well as the opposite way around, and dates stay saved to the dates they were made

for and don’t show up in the future/past months and years and stay saved when you return to them.