Web Application Documentation

Project Overview

This application is designed for WeDeliverTECH™.

This application serves as a platform for receptionists of WeDeliverTECH™ to manage their reception activities, focusing on staff member out-of-office clocking functionality and notification system as well as tracking deliveries of orders to customers. The application provides an intuitive Reception Management dashboard to help the receptionist manage staff presence and delivery information effectively.

Features

1. **Staff Member Out-of-Office Management**: Allows the receptionist to log staff members in and out of the office, track their out time, absence duration and calculated expected return time. Multiple staff members can be clocked out at the same time to facilitate efficient management. A notification is shown if a staff member does not return on time. This notification will stay visible on the page until the receptionist closes it manually.
2. **Dlivery Tracking**: Provides a Schedule Delivery area in the dashboard with functionality to track current deliveries by allowing the receptionist to manually enter driver information such as name, vehicle type, telephone, adress and expected return time. Notifications are shown if a delivery driver does not return in time.
3. **Digital Clock**: Displays the current date and time in the specified format ("Day Month Year Hour:Minute:Second") and updates every second.

Additional Feature

- **Map Component**: Allows users to use their current browser location in the adress input field or click the map icon to manually find an adress. This feature is optional and not part of the core requirements.

Project Structure

The project is organized as follows:

- `index.html`: The main entry point of the application, containing the main HTML structure.
- `src/`: Contains all the source code and assets for the application.
  - `api/wdt_api.js`: Manages API interactions, including fetching staff data from the Random User API.
  - `assets/Company logo.png`: The WeDeliverTECH™ logo used for branding.
  - `classes/`: Defines various components of the application logic:
  - `wdt_employee.js`: Contains a parent class with basic employee properties: name and surname.
  - `wdt_staff.js`: Inherits from wdt_employee and handles staff-related data and operations, with additional properties unique to staff members.

    - `wdt_delivery.js`: Inherits from wdt_employee and handles delivery operations, including driver name and surname, vehicle type, adress, and expected return time.
    - `wdt_time.js`: Handles time calculations and time-based events, such as calculating expected return times and durations.
- `components/wdt_map.js`: Implements map features to allow users to input addresses using their current location or by selecting a location manually.
- `events/wdt_event.js`: Handles event-based interactions.
- `styles/wdt_style.css`: Styles for the web application to ensure a visually appealing UI that adheres to the WeDeliverTECH™ branding profile.
- `utils/wdt_utility.js`: Contains utility functions used throughout the application to avoid redundancy.

Getting Started

Prerequisites

A modern web browser is required to run this application. Supported browsers include:

- Google Chrome (latest version)
- Mozilla Firefox (latest version)
- Microsoft Edge (latest version)
- Safari (latest version)
- Opera (latest version)

Running the Application

Simply open the `index.html` file in your web browser to use the application.

Usage

- **Staff Member Management**:

    - To clock out one or multiple staff members, simply select the staff members and click the **Out** button. Enter the expected absence duration, and the system will calculate the expected return time as well as visually provide a new status for the staff.
    - To clock in one or multiple staff members, simply select the staff members and click the **In** button. This will clear their out-of-office status.
    - If a staff member does not return in time, a notification with their picture, name and overdue duration will be shown.

- **Deliveries**:

    - To add a delivery, manually enter the driver information and click the **Add** button. The delivery board will display the driver details and their expected return time.
    - If a driver has not returned by the expected return time, a notification with relevant details for the receptionist to reach out to the driver will be shown.
    - To clear a driver from the delivery board, select one or more rows and click the **Clear** button. A comfirmation popup will be shown to confirm the removal of the driver to prevent

accidental clearing.

- **Map (Additional feature)**: Use the **crosshair** button to apply the current browsers location to the input adress field or the **Map** button to look up and select an adress manually.
- **Digital Clock**: The current date and time are displayed at the bottom of the page and updates every second.

External Libraries

The application uses several external libraries to provide different functionalities. Bootstrap and Random User API are part of the core project requirements, while others add extra features to enhance user experience. The key libraries include:

- **Bootstrap**: Provides a responsive CSS framework for consistent styling. (This is a core requirement).
    - **Usage Instructions**:
    - By default, we have implemented Boostrap through CDN for linking the CSS and JavaScript.
    - These will be found in the `<head>` of `index.html`:

      ```
      <link
      href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">
      <script
      src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>
      ```

    - However another option is to download the CSS and JavaScript compiled versions from Bootstrap's official website https://getbootstrap.com/
    - **Prerequisites** for proper bootstrap use:
        - Ensure that the `<head>` section of the `index.html` page as `<meta name="viewport" content="width=device-width, initial-scale=1">`
        - Aso make sure the HTML5 doctype `<!DOCTYPE html>` to ensure proper styling.
- **Random User API**: Used to generate demo information for five unique staff members (Thisis a core requirement).
    - **Usage Instructions**:
    - The API is used to fetch random user data to populate the staff table.
    - To use it, we simply make a GET request to `https://randomuser.me/api/?results=5&seed=wdttm`.
        - the paremeter `results` ensures we get 5 users.
        - the paremeter `seed` ensures we receive unique users each time. This seed can be anything really, for this project we have just provides `wdt` as short for **WeDeliverTECH™**
        - The response contains JSON data, which is then processed by the `staffUserGet` function in ``wdt_api.js`
- **Leaflet**: Used to implement map features for address input and selection (This is an additional feature).

    - **Usage Instructions**:
    - For this project, we are using the Leaflet CDN.

    ```
    <link rel="stylesheet"
    href="https://unpkg.com/leaflet@1.9.3/dist/leaflet.css" />
    <script src="https://unpkg.com/leaflet@1.9.3/dist/leaflet.js">
    </script>
    ```

    - We are using our map inside a `<div>` to display the map, which will be toggled upon click of hte map button. The rest of the javascript code happens in `wdt_map.js`.
- **Leaflet Geosearch**: Provides search capabilities within the Leaflet map to allow users to locate addresses (This is an additional feature).
    - **Usage Instructions**:
    - For this project, we are using the Leaflet Geosearch CDN.

    ```
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/leaflet-
    geosearch@3.4.0/dist/geosearch.css" />
    <script src="https://cdn.jsdelivr.net/npm/leaflet-
    geosearch@3.4.0/dist/geosearch.umd.js"></script>
    ```

- **Nominatim API**: Converts latitude and longitue coordinates to actual adresses (This is an additional feature).
    - **Usage Instructions**:
    - This API is used i `wdt_api.js` to convert coordinates into readable adresses.
    - The function `fetchAdress` makes an API call and returns a formatted string.
    - To use it, we simply make a GET request to `https://nominatim.openstreetmap.org/reverse?format=json&lat={lat}&lon={lng}`.
    - Our `fetchAdress` function replaces `{lat}` and `{lng}` with the current browsers location.

Developer Notes

- The main application logic and DOM elements are defined in `wdt_app.js`, which acts as the main entry point for managing components and initializing features.
- The **staffUserGet** function makes the API call to Random User API and processes the response to create staff objects and populate the DOM table.
- Modular JavaScript is used to keep code maintainable, with each component and class serving a specific purpose.
- CSS styles are defined in `wdt_style.css` to maintain consistent styling throughout the application and adhere to the branding guidelines.
- Hover animations are implemented for the navbar and buttons to enhance the user experience.

License