# **טכנולוגיות אינטרנט מתקדמות - 61776 (WEB)**

מועד הגשה: 20.4.2024

|  |  |
| --- | --- |
| **שם חבר.ת הצוות** | **תז** |
| ראמי אמאשה | 322241373 |
| מואיד חמזה | 207532920 |
| יסמין מורה | 207851585 |

תקציר הפרויקט – הפרויקט שלנו הוא אתר של מסעדה, המאפשר ללקוח לבצע כל מיני פעולות, כגון:

* צפייה בתפריט המסעדה וגם במסך הבית ישנה אנימציה המציגה תמונות למסעדה.
* להזמין שולחנות בזמן ובתאריך מסויים והמערכת מודיעה לכך אם ישנו מקום פנוי או לא.
* לקרוא קצת על המסעדה ועל המשימה שלנו.
* לשמור מקומות למופעיים פרטיים ( ימי הולדת, בר-מצווה, ....).

המערכת גם נותנת אפשרות לבעל המסעדה לעדכן את התפריט, לצפות בשולחנות השמורים וגם במופעים הפרטיים שכבר נשמרו.

האתר תומך ברקעים (themes) , האתר משנה את העיצוב לפי המצב הקיים של המערכת שנפתח ממנה האתר ( אם זה במחשב ואם זה בטלפון).

מימוש-

במימוש המערכת השתמשנו ב REACT, בחלק של העיצוב השתמשנו ב TAILWIND וגם ב CSS.

לגבי מסד הנתונים בחרנו ב FIREBASE.

בתיקיית גיטהאב מצורף תיק משתמש USER GUIDE ותיק מתכנת PROGRAMMERS BAG .

קישור ל GITHUB : <https://github.com/RamiAmasha31/A21-flavor-Voayge>

**קישור לאתר :** [**https://ramiamasha31.github.io/A21-flavor-Voayge/**](https://ramiamasha31.github.io/A21-flavor-Voayge/)

**1.** חלוקת העבודה

|  |  |  |
| --- | --- | --- |
| שם חבר הצוות | משימות שהוקצו | משימות שהושלמו |
| יאסמין | * תכנון ועיצוב האפליקציה: הגדרת הדרישות הפונקציונליות והלא-פונקציונליות של האפליקציה. * יצירת סכמות (collection) ב- Firestore לאחסון נתוני ההזמנות והאירועים הפרטיים והמיניו.. * הוספת עיצובים לאופן בהיר LIGHT THEME |  |
| ראמי | * בחירת טכנולוגיות ותשתיות: בחירת React.js כספרייה עיקרית לבניית ממשק המשתמש. בחירת Tailwind CSS כמסגרת עיצוב CSS. * פיתוח הרכיבים והתפקודים השונים: יישום לוגיקת תפקוד, כגון טיפול בטפסים, שליחת נתונים לפיירבייס, אחזור נתונים מפיירבייס ועוד. * עבודה על הוספת תכונת הרקעים (themes( (התכונה עובדת לפי מצב המערכת בה פותחים את האתר) |  |
| מואיד | * הרצה והפצה: הרצת האפליקציה במצב פיתוח ( (development modelלבדיקות סופיות. * בדיקות ואבטחת איכות: ביצוע בדיקות אינטגרציה ( integration tests) לוידוא תפקוד נכון של האפליקציה כמכלול. * הוספת עיצובים לאופן כהה DARK THEME |  |

**2.** דרישות פונקציונאלית :

1. The system allows the administrators to log in securely using a username and password.

2.  The system provides access control, restricting certain features (menu management, reservation management) to authenticated administrators only.

3. The system provides a responsive and user-friendly interface for both desktop and mobile devices.

4. The system displays the restaurant's location on a map, along with the operating hours and parking information.

5. The system allows the users to make reservations by providing their name, email, phone number, and the number of people.

6. The system prevents the users from making reservations if the restaurant's capacity is full for the selected time slot.

7. The system allows the users to make reservations for private events by providing additional information such as event type and number of people.

דרישות לא פונקציונאלית :

1. Performance: The system has fast response times and is able to handle a high volume of user requests without significant performance degradation.

2. Performance: The application responds quickly to user interactions and form submissions.

3. Security: User data (e.g., reservations, private event details) are stored securely in the Firebase Firestore database.

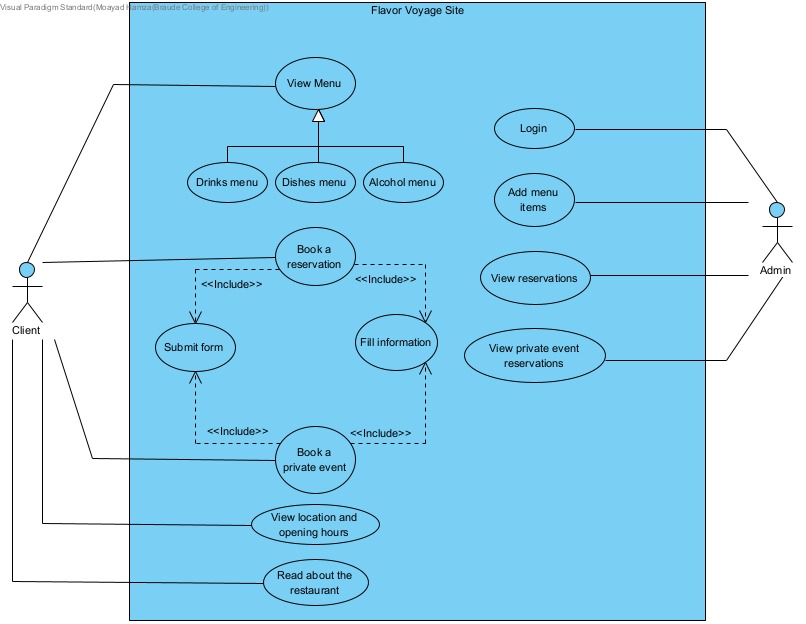
4. Usability: The user interface is intuitive and easy to navigate.

5. Usability: The application is responsive and work well on different devices and screen sizes.

**3.** ארכיטקטורה:

* This Architecture Diagram is a layered architecture diagram that separates the application into different layers: Frontend, Database.
* View: this sub-layer represents the user interface components.
* Model: the model sub-layer handles the application’s state management.
* Controller: this sub-layer consists of the react components.

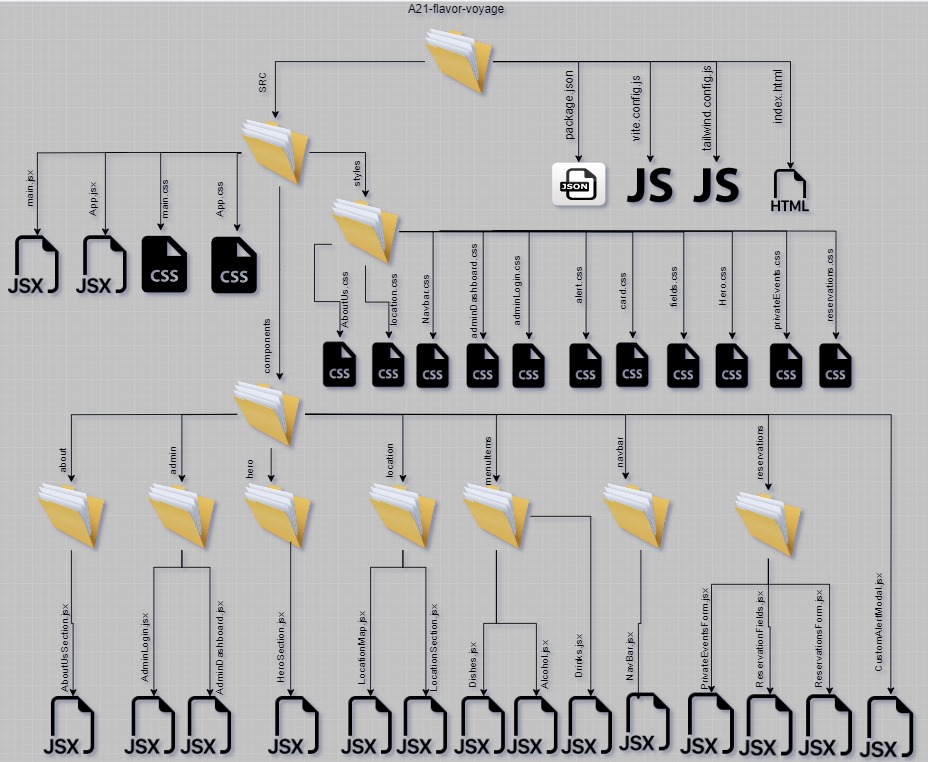
**4.** Use Case



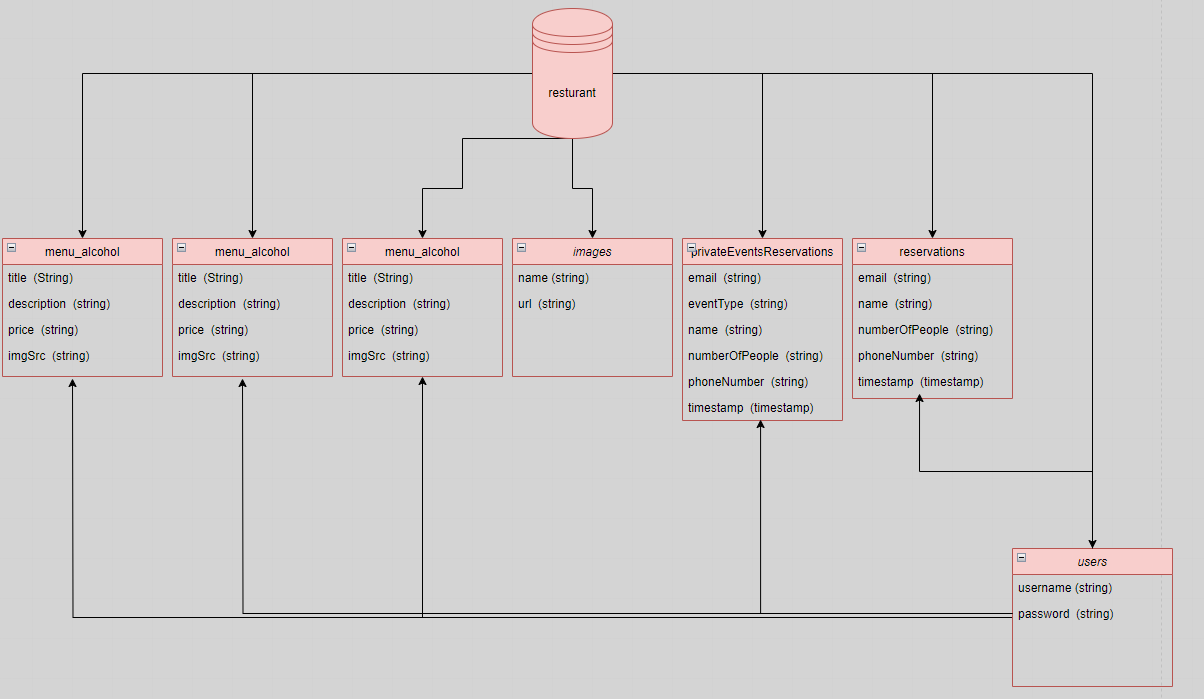
The diagram shows the relationships between the actors and their respective use cases. The client can perform various actions related to viewing the restaurant's offerings, making reservations, booking private events, and navigating the website. The admin, on the other hand, can log in to the admin dashboard, add menu items, and view regular reservations and private event reservations.

This use case diagram provides a clear visual representation of the different functionalities and interactions available to both client and admin users within the restaurant application.

**5.** דיאגרמה המתארת את התיקיות והקבצים השונים:



**6.** דיאגרמת מבנה DB



**Programmer’s Bag**

**AboutUsSection Component Documentation**

**Overview**

The AboutUsSection component is responsible for rendering the About Us section of the website. It displays information about the restaurant's story, mission, and invitation to join the culinary adventure.

**Functional Component Definition**

The AboutUsSection component is defined as a functional component in React using the arrow function syntax.

**Content Definition**

The content variable holds an array of objects, with each object representing a section of the About Us content. Each object contains a title and a description.

**JSX Structure**

The JSX structure returned by the component consists of a <section> element with specific classes for styling purposes. Within this section, a container <div> is used to center the content horizontally.

**Mapping Content**

The content array is mapped over to dynamically render each section. For each item in the content array, a <div> element is created containing the title and description.

**Title Animation**

The title of each section is animated using CSS classes to achieve a text reveal effect. Each character of the title is wrapped in a <span> element, and CSS animations are applied to animate each character sequentially. This animation technique is adapted from Builder.io's stagger text animation.

**Non-Breaking Space**

Spaces within the title are replaced with non-breaking space characters (\u00A0) to ensure proper spacing and alignment of text.

**Export**

Finally, the AboutUsSection component is exported as default, allowing it to be imported and used in other parts of the application.

### **Admin Components Documentation**

#### **AdminLogin Component**

##### **Overview**

The AdminLogin component functions as the login page for accessing the admin dashboard. Users input their username and password to authenticate and access administrative features.

##### **State Variables**

* **username**: Holds the user-entered username.
* **password**: Stores the user-entered password.
* **errorMessage**: Contains error messages for user feedback.
* **loggedIn**: Indicates the user's login status.

##### **Login Functionality**

The handleLogin function manages the login process. It checks the Firestore database for the provided username, verifies the password, and sets the loggedIn state to true upon successful authentication.

##### **Rendering Admin Dashboard**

If the user is logged in (loggedIn is true), the AdminDashboard component is rendered, passing the username as a prop.

##### **JSX Structure**

The JSX structure includes elements for displaying a welcome message, a form for entering username and password, and a login button. Error messages are displayed if authentication fails.

#### **AdminDashboard Component**

##### Overview

The AdminDashboard component oversees administrative tasks concerning menu items and reservations. It facilitates adding new menu items, viewing reservations, and managing reservations for private events.

##### State Variables

* **menuItem**: Holds the title of the menu item being added.
* **itemDescription**: Stores the description of the menu item being added.
* **itemPrice**: Captures the price of the menu item being added.
* **itemType**: Records the type of menu item (e.g., dishes, drinks, alcohol).
* **imgSrc**: Stores the URL of the image representing the menu item.
* **reservations**: Stores customer reservations.
* **privateEventReservations**: Manages reservations for private events.
* **alertMessage**: Contains success or error messages for user feedback.
* **showAddMenuItem**, **showReservations**, **showPrivateEventReservations**: Boolean variables controlling the visibility of different dashboard sections.

##### **Functionality**

* **Adding Menu Items**: Users can add new menu items by completing a form with title, description, price, and image URL.
* **Viewing Reservations**: Users can view reservations made by customers.
* **Viewing Private Event Reservations**: Users can view reservations made for private events.

##### **JSX Structure**

The JSX structure includes elements for displaying various sections of the admin dashboard based on user actions.

**HeroSection Component Documentation**

**Overview**

The HeroSection component is responsible for rendering the hero section of the website, showcasing a welcome message and rotating images from a collection stored in Firebase Firestore.

**State Variables**

* **currentImage**: Stores the index of the currently displayed image.
* **imageUrls**: Stores an array of URLs for the images fetched from the Firestore database.

**Firebase Configuration**

The Firebase configuration object firebaseConfig is used to initialize Firebase with the appropriate credentials and settings.

**Fetching Image URLs**

The useEffect hook is utilized to fetch image URLs from the Firestore collection named 'images'. Upon fetching, the URLs are stored in the imageUrls state variable.

**Image Rotation**

Another useEffect hook is employed to rotate the images displayed in the hero section. The currentImage state variable is updated at regular intervals to cycle through the available images.

**Text Content**

The hero section also contains text content welcoming users to the website and inviting them to experience a culinary journey like no other.

**JSX Structure**

The JSX structure consists of two main sections:

1. Text Content: Displays the welcome message and description.
2. Gallery: Renders the images fetched from Firestore, with a transition effect to switch between images.

**Location Component Documentation**

**Overview**

The LocationSection component displays the location information of the restaurant, including a map with the restaurant's location marker, operating hours, parking information, and address.

**LocationMap Component**

**Overview**

The LocationMap component is responsible for rendering the Google Map with the restaurant's location marker.

**Props**

* **location**: Object containing the latitude and longitude coordinates of the restaurant.
* **className**: Additional CSS class names to apply to the map container.

**Google Maps Integration**

The GoogleMapReact library is used to integrate Google Maps into the component. The component sets the default center and zoom level of the map, as well as disables map dragging, zoom control, and scrollwheel zoom.

**Marker Component**

A custom Marker component is defined to represent the restaurant's location on the map.

**LocationSection Component**

**Overview**

The LocationSection component is the parent component that encapsulates the location information display.

**Content**

* Map Display: The component includes the LocationMap component to render the map with the restaurant's location marker.
* Operating Hours: Displays the operating hours of the restaurant.
* Parking Information: Provides information about parking availability.
* Address: Shows the address of the restaurant.

**Location Data**

The LocationSection component defines an example location object with latitude and longitude coordinates.

**Menu Items Component Documentation**

**Overview**

The Menu Items component consists of three separate components: Alcohol, Dishes, and Drinks. Each component is responsible for fetching and displaying a specific category of menu items.

**Alcohol Component**

**Overview**

The Alcohol component fetches and displays alcoholic beverages from the Firestore database.

**Features**

* Fetches alcohol data from the 'menu\_alcohol' collection in Firestore.
* Displays alcohol items including title, description, price, and image.

**Dishes Component**

**Overview**

The Dishes component fetches and displays food dishes from the Firestore database.

**Features**

* Fetches dish data from the 'menu\_dishes' collection in Firestore.
* Displays dish items including title, description, price, and image.

**Drinks Component**

**Overview**

The Drinks component fetches and displays non-alcoholic drinks from the Firestore database.

**Features**

* Fetches drink data from the 'menu\_drinks' collection in Firestore.
* Displays drink items including title, description, price, and image.

**Usage**

Each component is designed to be easily integrated into the restaurant's website or application. Simply import the desired component and include it in the appropriate section of the UI.

**Navbar with Mega Menu Component Documentation**

**Overview**

The Navbar with Mega Menu component provides a navigation menu for the restaurant's website or application. It includes dropdown functionality for menu items with subcategories.

**Features**

* Responsive design for both desktop and mobile devices.
* Dropdown menu for menu items with subcategories.
* Easily customizable navigation items and dropdown content.

**Usage**

Simply import the NavbarWithMegaMenu component into your project and include it in the desired location within your application. You can customize the navigation items by updating the **navItems** array.

**Props**

The NavbarWithMegaMenu component does not accept any props.

**Example**

import React from 'react';

import NavbarWithMegaMenu from './NavbarWithMegaMenu';

const App = () => {

const handleNavItemClick = (section) => {

// Handle navigation to the selected section

console.log('Navigating to section:', section);

};

return (

<div>

<NavbarWithMegaMenu onNavItemClick={handleNavItemClick} />

{/\* Other components and content \*/}

</div>

);

};

export default App;

**PrivateEventsForm Component**

**Description:** The PrivateEventsForm component is responsible for handling reservations for private events. It includes a form where users can input their name, email, phone number, event type, and the number of people attending. Upon submission, the form data is sent to a Firestore database for storage. If the restaurant's capacity is full for the specified time, an alert is displayed to the user.

**Props:** None

**State:**

* formData: An object containing the form field values.
* showAlert: A boolean value indicating whether to show the alert modal.
* alertMessage: A string containing the message to be displayed in the alert modal.

**Methods:**

* handleInputChange: Handles changes in form input fields and updates the formData state accordingly.
* handleSubmit: Handles form submission. It validates the form data, checks the restaurant's capacity, and submits the data to Firestore.
* handleAlertClose: Closes the alert modal.

**ReservationFormFields Component**

**Description:** The ReservationFormFields component renders input fields for the reservation form, including name, email, and phone number.

**Props:**

* formData: An object containing the form field values.
* handleInputChange: A function to handle changes in form input fields.

**State:** None

**Methods:** None

**ReservationsForm Component**

**Description:** The ReservationsForm component is similar to the PrivateEventsForm but is specifically designed for general reservations. It includes a form where users can input their name, email, phone number, and the number of people attending. Upon submission, the form data is sent to a Firestore database for storage. If the restaurant's capacity is full for the specified time, an alert is displayed to the user.

**Props:** None

**State:**

* formData: An object containing the form field values.
* showAlert: A boolean value indicating whether to show the alert modal.
* alertMessage: A string containing the message to be displayed in the alert modal.

**Methods:**

* handleInputChange: Handles changes in form input fields and updates the formData state accordingly.
* handleSubmit: Handles form submission. It validates the form data, checks the restaurant's capacity, and submits the data to Firestore.
* handleAlertClose: Closes the alert modal.

**Additional Notes:**

* These components require Firebase configuration to connect to Firestore for data storage.
* They utilize CustomAlertModal component for displaying alerts to the user.
* Form input validation can be added for better user experience and data integrity.
* Styling classes are used for layout and design. Adjustments can be made according to the project's styling requirements.

**App Component**

**Description:** The App component is the root component of the application. It renders the main navigation bar (NavbarWithMegaMenu) and the active section based on user interaction.

**Component Structure:**

* NavbarWithMegaMenu: Renders the main navigation bar with a mega menu for navigating different sections of the application.
* Active Section: Renders the currently active section of the application based on user interaction. Initially set to the HeroSection.

**State:**

* activeSection: A state variable used to manage the currently active section of the application. It is initialized with the HeroSection component.

**Handlers:**

* handleNavItemClick: A function passed to the NavbarWithMegaMenu component as a prop. It updates the activeSection state with the selected section when a navigation item is clicked.

**Example Usage:**

import React, { useState } from 'react';

import NavbarWithMegaMenu from "./components/navbar/Navbar";

import HeroSection from './components/hero/HeroSection';

export default function App() {

const [activeSection, setActiveSection] = useState(<HeroSection/>);

const handleNavItemClick = (section) => {

setActiveSection(section);

};

return (

<div>

<NavbarWithMegaMenu onNavItemClick={handleNavItemClick} />

{activeSection}

</div>

);

}

**CustomAlertModal Component**

**Description:** The CustomAlertModal component is a reusable modal component for displaying alert messages to the user. It appears as a dialog box with a message and an "OK" button. When the button is clicked, the modal closes.

**Props:**

* message: A string containing the message to be displayed in the alert modal.
* onClose: A function to handle the closing of the modal when the "OK" button is clicked.

**Example Usage:**

import React, { useState } from 'react';

import CustomAlertModal from './CustomAlertModal';

const ExampleComponent = () => {

const [showAlert, setShowAlert] = useState(true);

const handleCloseAlert = () => {

setShowAlert(false);

};

return (

<div>

<h1>Example Component</h1>

{showAlert && (

<CustomAlertModal

message="This is an example alert message."

onClose={handleCloseAlert}

/>

)}

</div>

);

};

export default ExampleComponent;

**Database Structure and Usage**

**Introduction:** This document outlines the database structure for the restaurant project built with React. It describes the collections, documents, and fields used in the Firestore database, as well as how these data are utilized in the project.

**Firestore Database Structure**

**Collections:**

1. menu\_dishes
   * Description: Collection to store menu items categorized as dishes.
   * Fields:
     + title: Title of the dish.
     + description: Description of the dish.
     + price: Price of the dish.
     + imgSrc: URL of the image representing the dish.
2. menu\_drinks
   * Description: Collection to store menu items categorized as drinks.
   * Fields:
     + title: Title of the drink.
     + description: Description of the drink.
     + price: Price of the drink.
     + imgSrc: URL of the image representing the drink.
3. menu\_alcohol
   * Description: Collection to store menu items categorized as alcoholic beverages.
   * Fields:
     + title: Title of the alcoholic beverage.
     + description: Description of the alcoholic beverage.
     + price: Price of the alcoholic beverage.
     + imgSrc: URL of the image representing the alcoholic beverage.
4. reservations
   * Description: Collection to store reservations made by customers.
   * Fields:
     + name: Name of the customer.
     + email: Email of the customer.
     + phoneNumber: Phone number of the customer.
     + numberOfPeople: Number of people in the reservation.
     + timestamp: Timestamp indicating the date and time of the reservation.
5. privateEventsReservations
   * Description: Collection to store reservations made for private events.
   * Fields:
     + name: Name of the person organizing the event.
     + email: Email of the person organizing the event.
     + phoneNumber: Phone number of the person organizing the event.
     + date: Date of the event.
     + timestamp: Timestamp indicating the date and time of the reservation.
6. users
   * Description: Collection to store the users usernames and passwords for the restaurant admins.
   * Fields:
     + username: username of the user.
     + password: password of the user.

**Usage in Project:**

* Menu Items: The menu items are fetched from the respective collections (menu\_dishes, menu\_drinks, menu\_alcohol) and displayed on the website's menu page.
* Reservations: Reservation data from the reservations collection are fetched and displayed in the admin dashboard, also the informations about the user reservations saved in this table.
* Private Event Reservations: Data from the privateEventsReservations collection are fetched and displayed in the admin dashboard, also the informations about the user reservations saved in this table.
* Admin Login: The admin username and passwords are saved in users collection, and if they are exist then the admin has the possibility to add menu items and to show the existing reservations.

[Database Link](need to add the link here )

**Index File**

**Description:** The main.js file serves as the entry point of the React application. It renders the root component (App) into the DOM.

**Dependencies:**

* react: A JavaScript library for building user interfaces.
* react-dom: Provides DOM-specific methods used to interact with the DOM.
* react-dom/client: The client-side entry point of the react-dom package.
* react.StrictMode: A tool for highlighting potential problems in an application during development.
* App.jsx: The root component of the application.
* index.css: CSS file for styling the application.

**Rendering:**

* ReactDOM.createRoot: Creates a root for the React application. This method is part of the new React concurrent mode APIs.
* document.getElementById('root'): Retrieves the DOM element with the ID 'root', where the application will be rendered.
* render: Renders the App component into the root DOM element.

**Example Usage:**

import React from 'react';

import ReactDOM from 'react-dom/client';

import App from './App.jsx';

import './index.css';

ReactDOM.createRoot(document.getElementById('root')).render(

<React.StrictMode>

<App />

</React.StrictMode>

);

**User’s Guide**

**User Guide for Website Navigation**

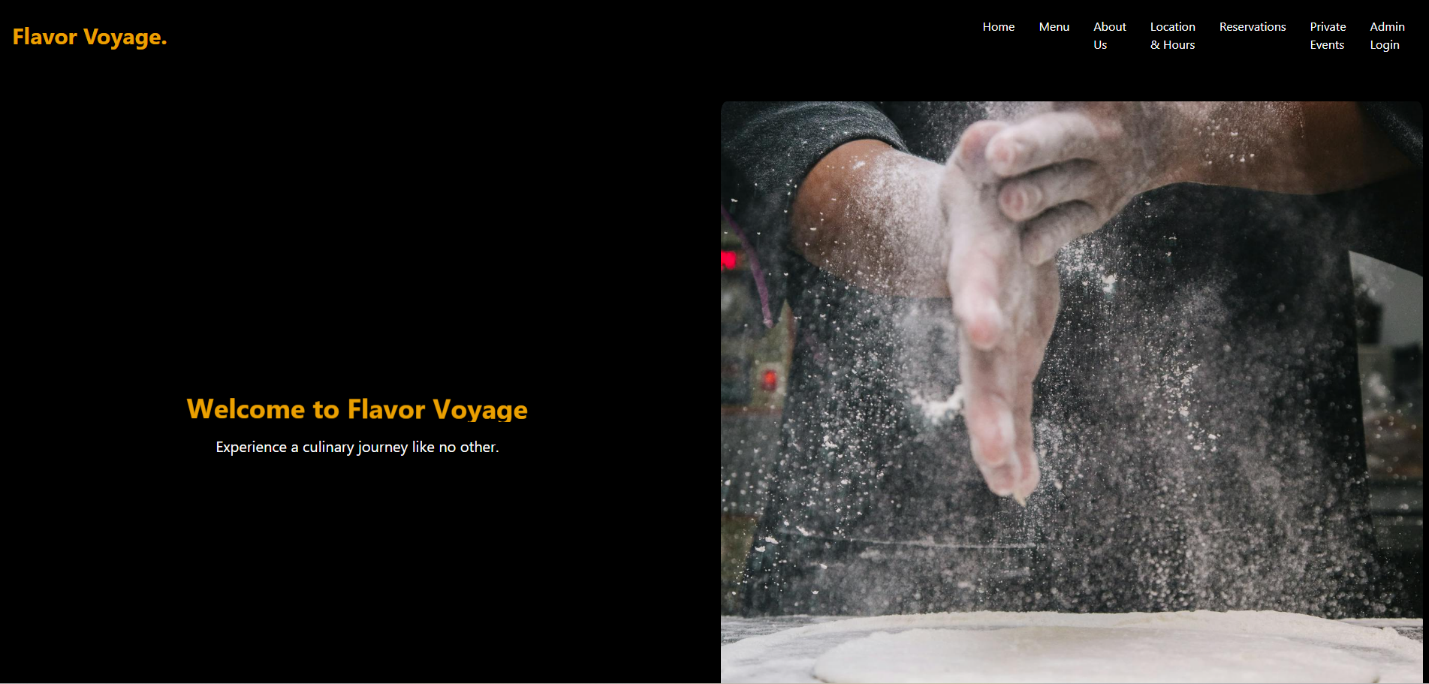
**Introduction** Welcome to our website! This user guide will help you navigate through the different sections and features of our website.

**Table of Contents**

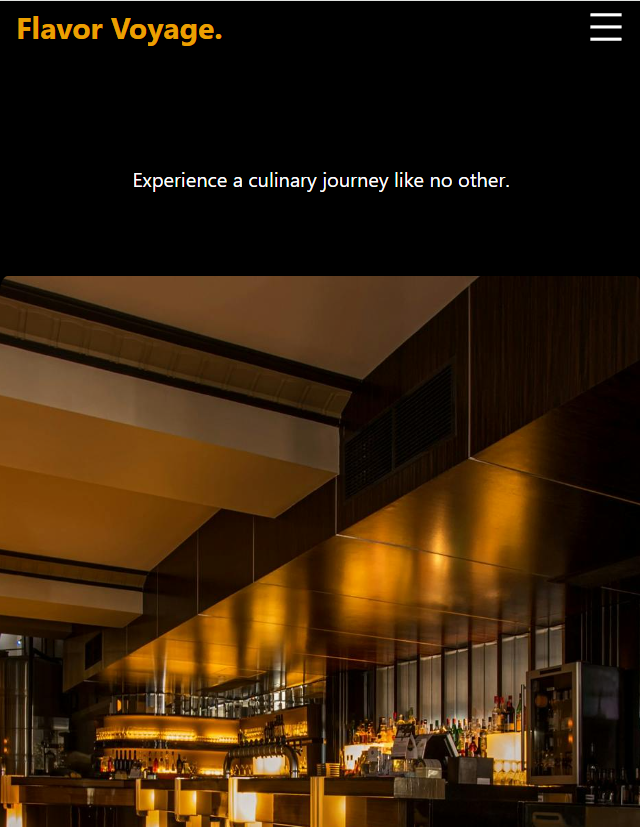
1. Homepage
2. Navigation Menu
3. Reservations
4. Private Events
5. About Us
6. Admin Login

**Homepage** Upon visiting our website, you will land on the homepage. A title and animated gallery will be displayed, and at the top will be shown navigation bar.

**Desktop View** Here's how our homepage looks like on desktop devices:



**Mobile View** For mobile users, the homepage is optimized for a seamless browsing experience:



**Navigation Menu** Our website features a navigation menu located at the top of the page. You can use this menu to access different sections of the website, such as the menu, reservations, private events, about us, and contact us.

**Desktop View** Here's how our navigation bar looks like on desktop devices:



**Mobile View** For mobile users, the navbar is hidden, and there is a mega menu button:



after clicking on it, the navbar will be displayed:

תמונה שמכילה טקסט, צילום מסך, גופן, עיצוב

התיאור נוצר באופן אוטומטי

**Reservations** To make a reservation at our restaurant, navigate to the "Reservations" section in the navigation menu. Fill out the reservation form with your name, email, phone number, and the number of people in your party. Then, click the "Submit" button to confirm your reservation.

**Desktop View** Here's how our reservations page looks like on desktop devices:

תמונה שמכילה טקסט, צילום מסך, עיצוב

התיאור נוצר באופן אוטומטי

**Mobile View** For mobile users:

תמונה שמכילה טקסט, צילום מסך, גופן, עיצוב

התיאור נוצר באופן אוטומטי

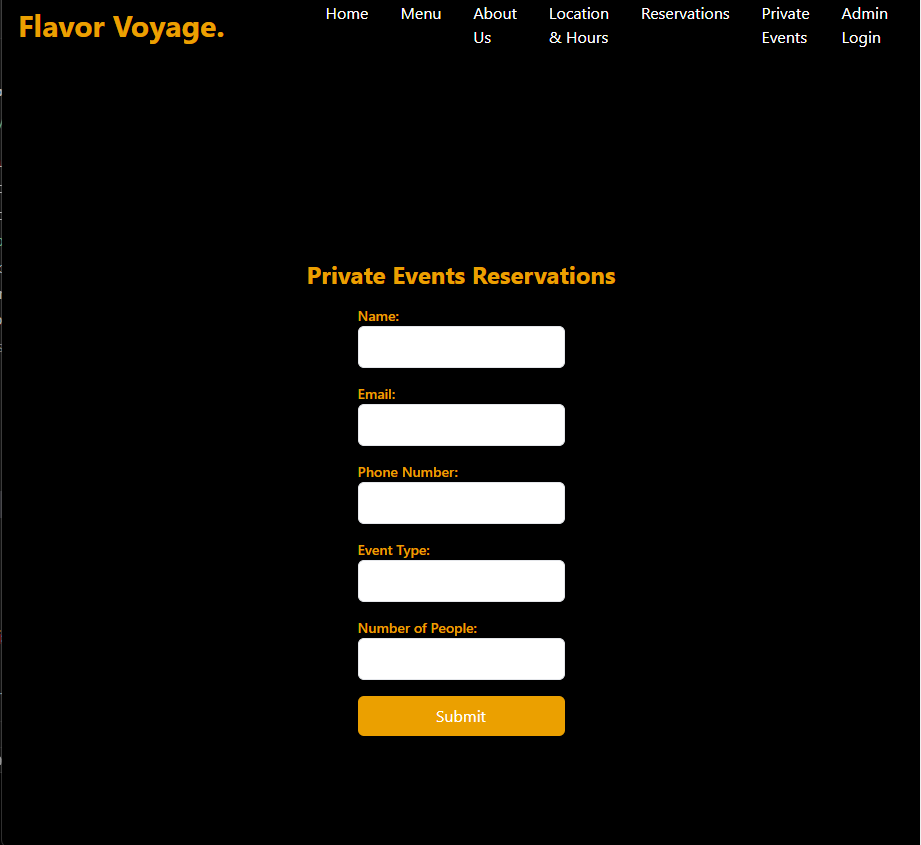
after successful reservation you will get :

תמונה שמכילה טקסט, צילום מסך, תוכנה, עיצוב

התיאור נוצר באופן אוטומטי

**Private Events** If you're interested in hosting a private event at our restaurant, you can do so by visiting the "Private Events" section. Fill out the private events reservation form with your event details, including your name, email, phone number, event type, and the number of attendees.

**Desktop View** Here's how our private events reservations page looks like on desktop devices:



**Mobile View** For mobile users:

תמונה שמכילה טקסט, צילום מסך, עיצוב

התיאור נוצר באופן אוטומטי

after successful reservation you will get :

תמונה שמכילה טקסט, צילום מסך, תוכנה, תכונות מולטימדיה

התיאור נוצר באופן אוטומטי

**About Us** Learn more about our restaurant by visiting the "About Us" section. Here, you will find information about our history, mission, values.

**Desktop View**

תמונה שמכילה טקסט, צילום מסך, גופן, מסמך

התיאור נוצר באופן אוטומטי

**Mobile view:**

תמונה שמכילה טקסט, צילום מסך, תוכנה, גופן

התיאור נוצר באופן אוטומטי

**\*זאת אנימציה לכן לא כל המלות מופיעות בתמונה.**

**Admin Login** This page is made for the restaurant managers for enabling them to add menu items and display the reservations. At first, the admin is asked for entering username and password (username: rami, password: amasha). And then the admin dashboard is being displayed.

תמונה שמכילה טקסט, צילום מסך, גופן

התיאור נוצר באופן אוטומטי

**Conclusion** We hope this user guide helps you navigate our website effectively. If you have any further questions or need assistance, feel free to reach out to us via email: ramiamasha84@gmail.com.