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

**הגשת פרויקט**

**<Virtual Event Platform><B28><17>**

****

|  |  |
| --- | --- |
| **שם חבר.ת הצוות** | **תז** |
| מירנה אבו חליל | 322662081 |
| דיאנא חוג'יראת | 324297308 |
| פבלו דובצ'ינסקי | 327248720 |
| רוני ברין | 318295516 |

<תקציר הפרויקט - עד חצי עמוד- פונקציונליות מרכזיות לפי משתמשים>

**תקציר הפרויקט :**

האתר שלנו הוא פתרון אינטרנטי ידידותי למשתמש שנועד לייעל את תהליך גילוי אירועים אקדמיים,

הפלטפורמה כוללת רשימת אירועים מסודרת לפי קטגוריות עם כל הפרטים החשובים, עמוד ייעודי לכל אירוע עם ביקורות ודירוגים, ולוח שנה למעקב אחר אירועים, מה שמבטיח ארגון קל ונוח.

בנוסף, הפלטפורמה תומכת במצבי תאורה בהירים וכהים לנגישות מיטבית .

ומאפשרת לסטודנטים, חוקרים ואקדמאים להשתתף בקלות בפעילויות אקדמיות ברחבי העולם המתקיימות בZOOM .

**פונקציונליות מרכזיות לפי משתמשים:**

* + חיפוש וגילוי אירועים לפי קטגוריות וסוגים .
  + עיון בפרטי אירועים, כולל תיאור, תאריך, שעה, וקישור להשתתפות.
  + אפשרות לעקוב אחר האירועים דרך לוח שנה.
  + תמיכה במצב תאורה בהיר וכהה לשיפור חוויית המשתמש.
  + דף ייעודי לכל אירוע הכולל ביקורות ודירוגים לשקיפות ונוחות.
  + לוח שנה אינטראקטיבי לצפייה באירועים בצורה מסודרת יותר.

< מימוש-שמות הטכנולוגיות המרכזיות בכל אחד מהחלקים -/styling/db/ Backend/frontend>

|  |  |  |
| --- | --- | --- |
| **חלק** | **טכנולוגיות מרכזיות** | **שימוש** |
| Styling | Tailwind CSS | עיצוב דינמי ומהיר עם מחלקות מוכנות מראש. |
| Fronted | React.js + Next.js | בניית ממשק משתמש ודינמיות עם SEO טוב יותר. |
| Backend | Next.js API Routes | ניהול בקשות API ותקשורת עם ה- Database. |
| Database | MongoDB Atlas | אחסון נתונים גמיש וניהול אירועים ומשתמשים. |

<קישור לתיקיית גיט ציבורי>

<https://github.com/pablo1611/Events-Website.git>

<קישור לאתר>

<https://events-website-w3yk.vercel.app/login>

1. עליכם להמשיך את בניית האתר לפי האלמנטים המתקדמים שלמדתם
2. יש למנות מהנדס מערכת בכל צוות, אשר יהיה אחראי על הגדרת והקצאת המשימות בתרגיל זה.  
   נא לרשום את שם הסטודנט בתרגיל זה. על מהנדס המערכת לכתוב כיצד נעשתה חלוקת העבודה מול הצוות, מה היו המשימות של כל חבר צוות, האם היה ממשק בין חברי הצוות, והאם המשימות מולאו:

מהנדס המערכת : פבלו דובצ'ינסקי

|  |  |  |
| --- | --- | --- |
| **שם חבר הצוות** | **משימות שהוקצו** | **משימות שהושלמו** |
| מירנה אבו חליל | בניית תיק משתמש + תיקון דרישות ו USECASE | סיום תיק משתמש |
| דיאנא חוג'יראת | בניית תיק מתכנת + תיקון דיאגרמת ארכיטקטורה | סיום תיק מתכת |
| פבלו דובצ'ינסקי | עדכון מסד נתונים + תקציר פרויקט ומבנה קוד | עדכון מסד נתונים ומבנה קוד |
| רוני ברין | ץיקון ושיפור קוד | הוספת קישורים |

1. רשימת דרישות פונקציונליות ולא פונקציונליות (בנפרד, יש לסווג דרישות לא פונקציונליות לפי wikipedia NFR).

**Functionality Requirements :**

* + - 1. The system allows registration.
      2. The system allows users to log in using an email address and password.
      3. The system allows identification.
      4. The system displays events by categories.
      5. The system allows users to select a catgory.
      6. The system allows users to choose between dark and light mode.
      7. The system allows entry to events through a link for each event.
      8. The system allows more than one user to connect to the same event.
      9. The system allows the user to view the site from different places(computer/ mobile/ ipad/ etc.) by pressing the F12 button.
      10. The system allows users to select event and view details.
      11. The system allows users to view a calendar displaying events on their respective dates.

**Non-Functionality Requirements :**

**1.Availability:**

The system will be available 24/7 with minimal downtime, allowing users to access the platform from anywhere and at any time via a stable internet connection.

**2.Privacy:**

The system will store passwords in an encrypted format to protect user data**.**

**3.Performance:**

The system will provide fast page loading (less than 2 seconds) to ensure a smooth user experience.

Filtering and searching for events will be performed quickly without noticeable delays.

**4. Maintainability:**

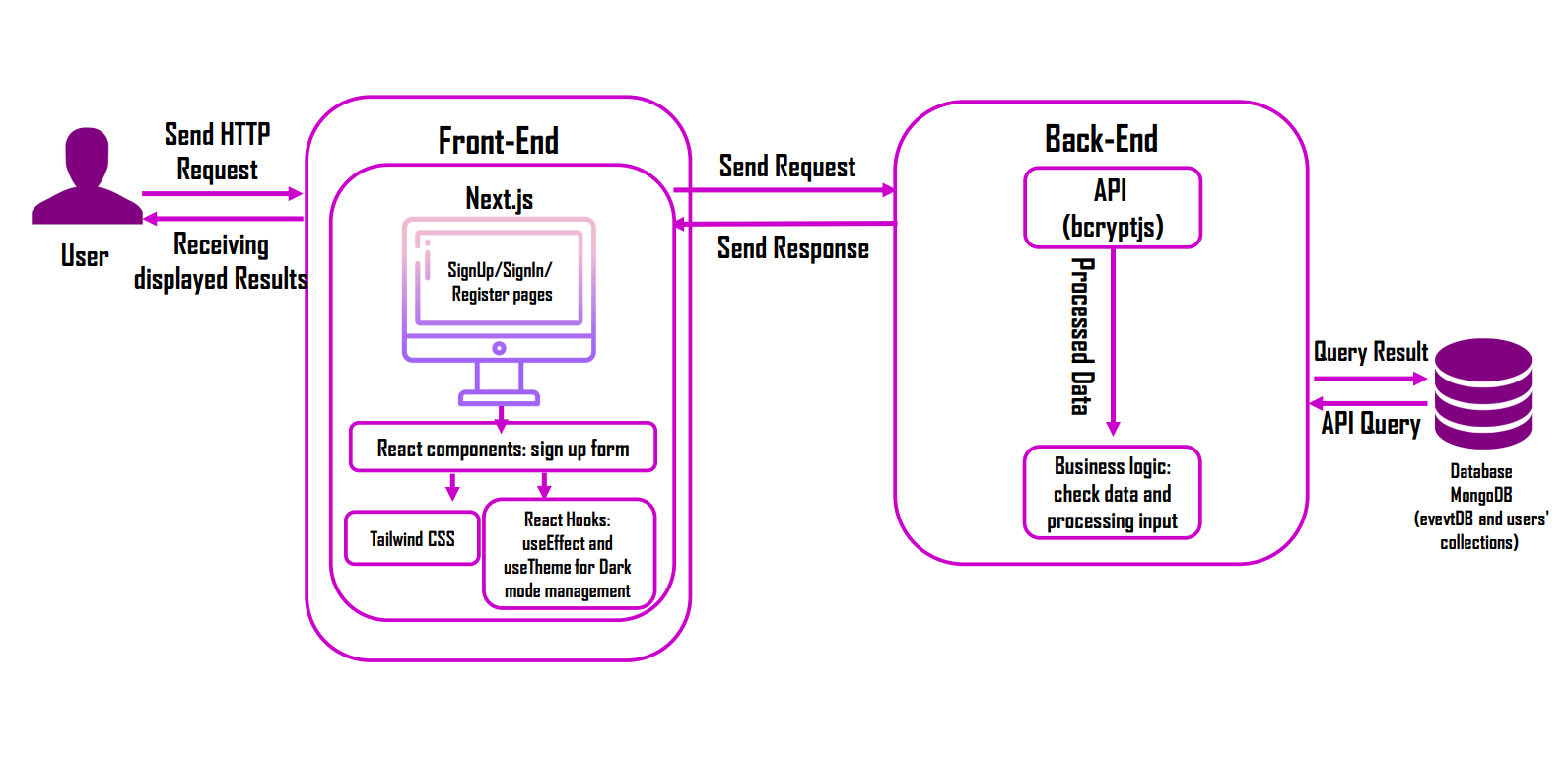
The system will be designed to support future updates and modifications without requiring significant code changes**.**

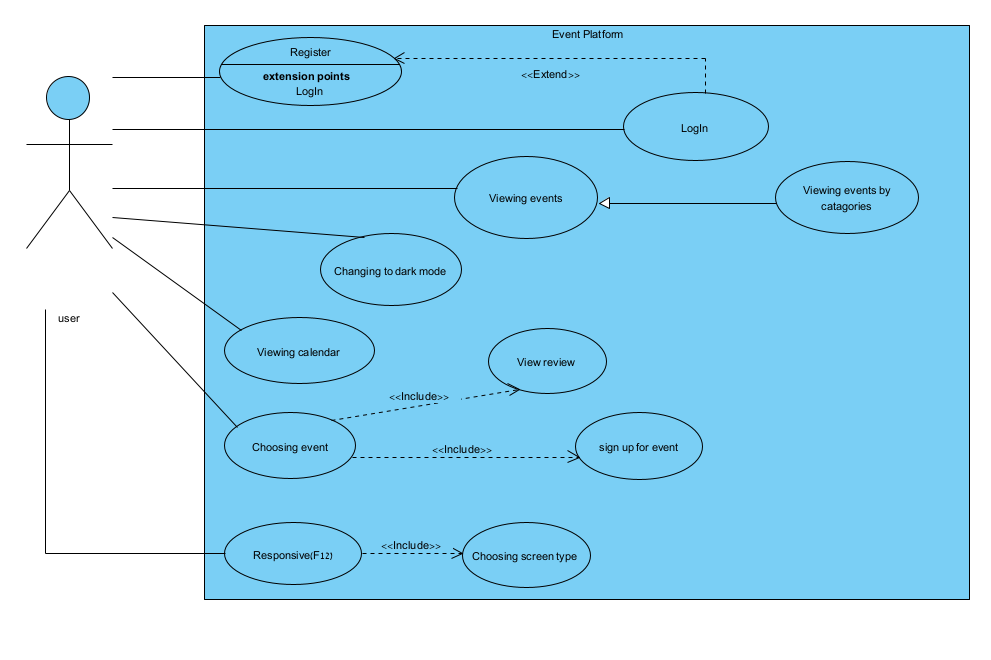
**5. Usability:**

The system will be easy to navigate, even for users with basic technical knowledge**.**

Error messages will be clear and include instructions to resolve issues (like : "Missing required field").

3. מבנה ואינטראקציה:  
 א. ארכיטקטורה מעודכנת של האתר (תרשים הכולל את האלמנטים המרכזיים)



ב.דיאגרמת use case המתארת את השימוש באתר.

4. מבנה סופי של האתר שלכם:

**נדרשת בכל פרויקט פריסה מלאה (deploment) של הפרויקט!   
לא יתקבלו הגשות של קבצים או אתר ב -localhost .**

**טכנולוגיות**:    להלן המרכיבים הטכנולוגיים המומלצים לשימוש בפרויקט:

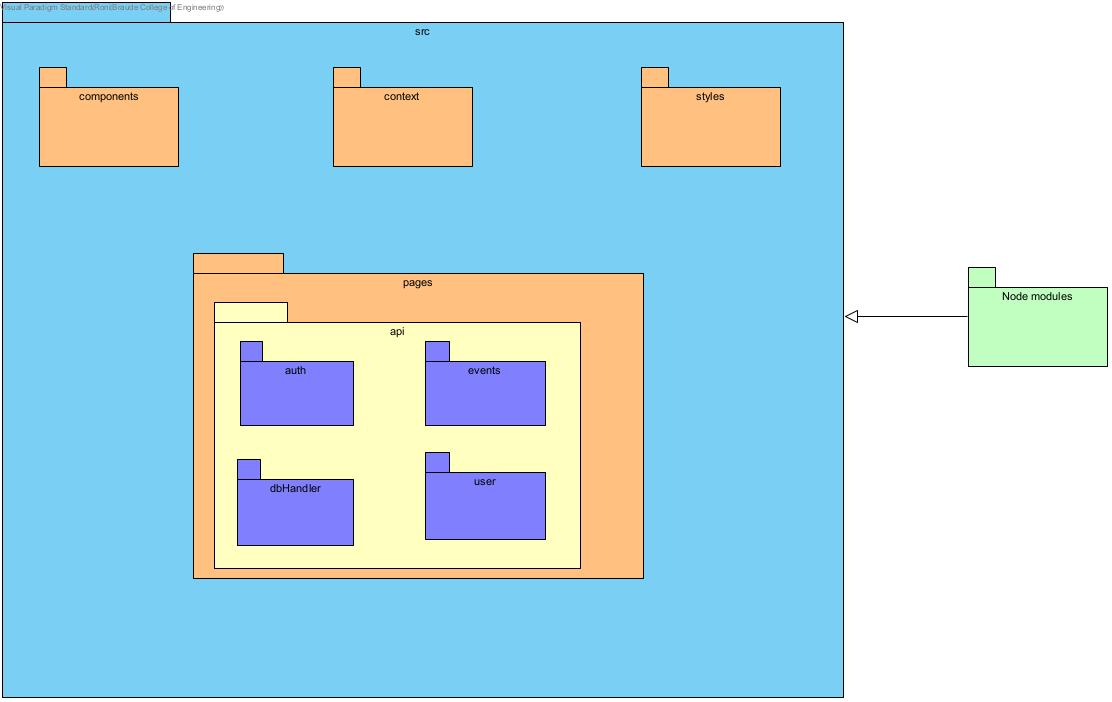
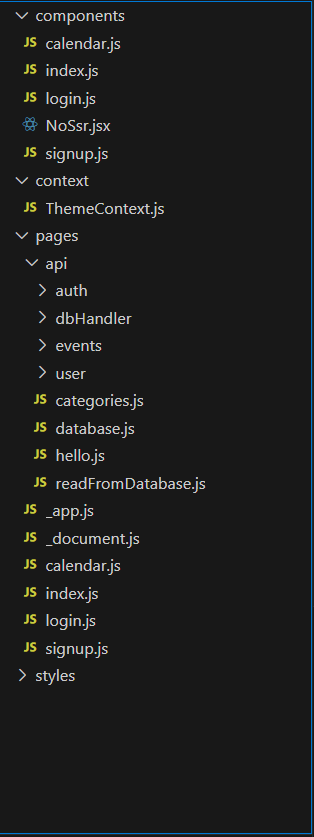
·   - front-end: React or Preact or Next with Tailwind

    - back-end:

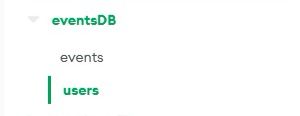
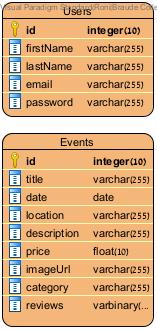
    - option 1 - remote services/APIs

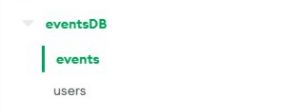
    - option 2 - node js and express deployed on VERCEL

1. האתר ימומש ב -react/preact/Next, וכן שימוש ב Tailwind - נא להציג דיאגרמה המתארת את התיקיות והקבצים השונים. יש לפרט את הקומפוננטות השונות.



1. פירוט את פריטי המידע - דיאגרמת מבנה DB.

****



1. יש להגיש תיק למתכנת:

# Introduction

## Overview

The Academic Events Platform is a web-based application designed to streamline the process of discovering, registering for, and managing academic events. It provides an interface where users can browse upcoming events, register to them, and submit reviews. The platform includes features such as authentication, personalized event recommendations, and a user-friendly calendar view.

## Purpose

The platform is built with Next.js for a dynamic and efficient frontend, while Node.js and MongoDB power the backend and database. Authentication is handled using bcrypt for password hashing, ensuring security. The system also uses Tailwind CSS for styling and Context API for state management.

# Main Files

## Installation

Installation dependencies, to ensure that all dependencies are correctly installed before proceeding.

npm install next react react-dom

npm install mongodb

npm install bcryptjs

npm install tailwindcss postcss autoprefixer -D

## 

## Imports

The imports allow authentication, context management, and MongoDB connectivity, UI components, and styling.

import dynamic from 'next/dynamic';

import React, { useState, useEffect, createContext, useContext } from 'react';

import bcrypt from 'bcryptjs';

import clientPromise from '../dbHandler/index';

import { MongoClient, ObjectId } from "mongodb";

import { Html, Head, Main, NextScript } from "next/document";

import { useRouter } from 'next/router';

import { useTheme } from '../context/ThemeContext';

import Image from 'next/image';

import Link from 'next/link';

import { Geist } from 'next/font/google';

## Data Fetch

Fetching data from the database is done using MongoDB queries, to retrieve all stored events from the database.

const client = await clientPromise;

const database = client.db("eventsDB");

const events = database.collection("events");

const result = await events.find({}).toArray();

# CSS Use

CSS is used extensively throughout the application to ensure a modern, responsive, and visually appealing user interface. The platform usesTailwind CSS to manage styles efficiently and maintain consistency across components.

CSS Implementation:

Global Styles: Configured in tailwind.config.js and imported into \_app.js.

Component-Specific Styles: Tailwind utility classes are applied directly within JSX elements.

Theme-Based Styling: CSS is dynamically adjusted based on the selected dark or light mode.

#### **Global Styles in globals.css**

@tailwind base;

@tailwind components;

@tailwind utilities;

:root {

--foreground-rgb: 0, 0, 0;

--background-start-rgb: 214, 219, 220;

--background-end-rgb: 255, 255, 255;

}

body {

color: rgb(var(--foreground-rgb));

background: rgb(var(--background-end-rgb));

}

#### **Dark Mode Implementation**

The function toggles between dark and light mode by adding or removing the dark class from the documentElement.

const toggleDarkMode = () => {

setDarkMode(!darkMode);

localStorage.setItem('darkMode', !darkMode);

document.documentElement.classList.toggle('dark');

};

# Key Functions

## User Authentication

Signup API: Handles new user registrations. It receives user details (first name, last name, email, password). Hashes the password using bcrypt. Stores the user in MongoDB. Returns a success message upon completion.

export default async function handler(req, res) {

if (req.method !== 'POST') {

return res.status(405).json({ error: 'Method not allowed' });

}

try {

const { firstName, lastName, email, password } = req.body;

const salt = await bcrypt.genSalt(10);

const hashedPassword = await bcrypt.hash(password, salt);

const client = await clientPromise;

const database = client.db("eventsDB");

const users = database.collection("users");

await users.insertOne({ firstName, lastName, email, password: hashedPassword });

res.status(201).json({ message: 'User created successfully' });

} catch (error) {

res.status(500).json({ error: 'Internal server error' });

}

}

## Event Fetching

It retrieves all available events from the database. Connects to MongoDB. Fetches all event documents. Returns the list of events.

export default async function handler(req, res) {

try {

const client = await clientPromise;

const database = client.db("eventsDB");

const events = database.collection("events");

const result = await events.find({}).toArray();

res.status(200).json({ events: result });

} catch (error) {

res.status(500).json({ error: 'Error fetching events' });

}

}

## Event Registration

Allows users to register for an event. Receives event ID and user ID. Updates the event document to add the user to the registered list. Returns a success message if registration is completed.

export default async function handler(req, res) {

if (req.method !== 'POST') {

return res.status(405).json({ error: 'Method not allowed' });

}

const { id, userId } = req.body;

try {

const client = await clientPromise;

const database = client.db("eventsDB");

const events = database.collection("events");

const result = await events.updateOne(

{ \_id: new ObjectId(id) },

{ $addToSet: { registeredUsers: userId } }

);

res.status(200).json({ message: 'Successfully registered for event' });

} catch (error) {

res.status(500).json({ error: 'Internal server error' });

}

}

# Design Patterns & Code Templates

## Context API for State Management

The Context API is used to manage global states such as dark mode and authentication, ensuring cleaner and more maintainable code.

const ThemeContext = createContext();

export function ThemeProvider({ children }) {

const [darkMode, setDarkMode] = useState(false);

useEffect(() => {

const isDark = localStorage.getItem('darkMode') === 'true' ||

(!('darkMode' in localStorage) && window.matchMedia('(prefers-color-scheme: dark)').matches);

setDarkMode(isDark);

}, []);

return (

<ThemeContext.Provider value={{ darkMode, toggleDarkMode }}>

{children}

</ThemeContext.Provider>

);

}

## 

## 

## 

## 

## 

## API Routes for Modular Backend Handling

Next.js API routes provide a clean separation between frontend and backend, making maintenance easier.

export default async function handler(req, res) {

if (req.method !== 'POST') {

return res.status(405).json({ error: 'Method not allowed' });}

try {

const { email, password } = req.body;

const client = await clientPromise;

const database = client.db("eventsDB");

const users = database.collection("users");

const user = await users.findOne({ email });

if (!user) {

return res.status(401).json({ error: 'Invalid credentials' }); }

const isValid = await bcrypt.compare(password, user.password);

if (!isValid) {

return res.status(401).json({ error: 'Invalid credentials' }); }

return res.status(200).json({

id: user.\_id,

email: user.email,

firstName: user.firstName,

lastName: user.lastName });

} catch (error) {

return res.status(500).json({ error: 'Internal server error' });

}}

## Component-Based UI Design

Ensuring reusability and better separation of concerns. This ensures UI consistency and simplifies updates across the project.

const ReviewCard = ({ review, darkMode }) => {

return (

<div className={`${darkMode ? 'bg-gray-800' : 'bg-white'} rounded-lg p-6 shadow-md mb-4`}>

<h4 className={darkMode ? 'text-white' : 'text-gray-900'}>{review.reviewer}</h4>

<p className={darkMode ? 'text-gray-400' : 'text-gray-500'}>{review.lectureDate}</p>

<p className={darkMode ? 'text-gray-300' : 'text-gray-600'}>{review.review}</p>

</div>

);

};

## Home Page

Serves as the landing page displaying a list of upcoming events. Fetches data from the /api/events endpoint to dynamically render events. Provides navigation links to login/signup and event details.

import { useState, useEffect } from 'react';

import Link from 'next/link';

export default function Home() {

const [events, setEvents] = useState([]);

useEffect(() => {

fetch('/api/events')

.then(res => res.json())

.then(data => setEvents(data.events));

}, []);

return (

<div>

<h1>Upcoming Events</h1>

{events.map(event => (

<Link key={event.\_id} href={`/events/${event.\_id}`}>

<a>{event.title}</a>

</Link>

))}

</div>

);

}

## Event Details Page

Displays event details and allows users to register.

import { useRouter } from 'next/router';

import { useEffect, useState } from 'react';

export default function EventDetails() {

const router = useRouter();

const { id } = router.query;

const [event, setEvent] = useState(null);

useEffect(() => {

if (id) {

fetch(`/api/events/${id}`)

.then(res => res.json())

.then(data => setEvent(data));

}

}, [id]);

if (!event) return <p>Loading...</p>;

return (

<div>

<h1>{event.title}</h1>

<p>{event.description}</p>

</div>

);

}

## User Event Calendar

Displays a calendar view of all events a user is registered for.

import { useState, useEffect } from 'react';

export default function Calendar() {

const [events, setEvents] = useState([]);

useEffect(() => {

fetch('/api/user/events')

.then(res => res.json())

.then(data => setEvents(data.events));

}, []);

return (

<div>

<h1>My Event Calendar</h1>

{events.map(event => (

<p key={event.\_id}>{event.title}</p>

))}

</div>

);

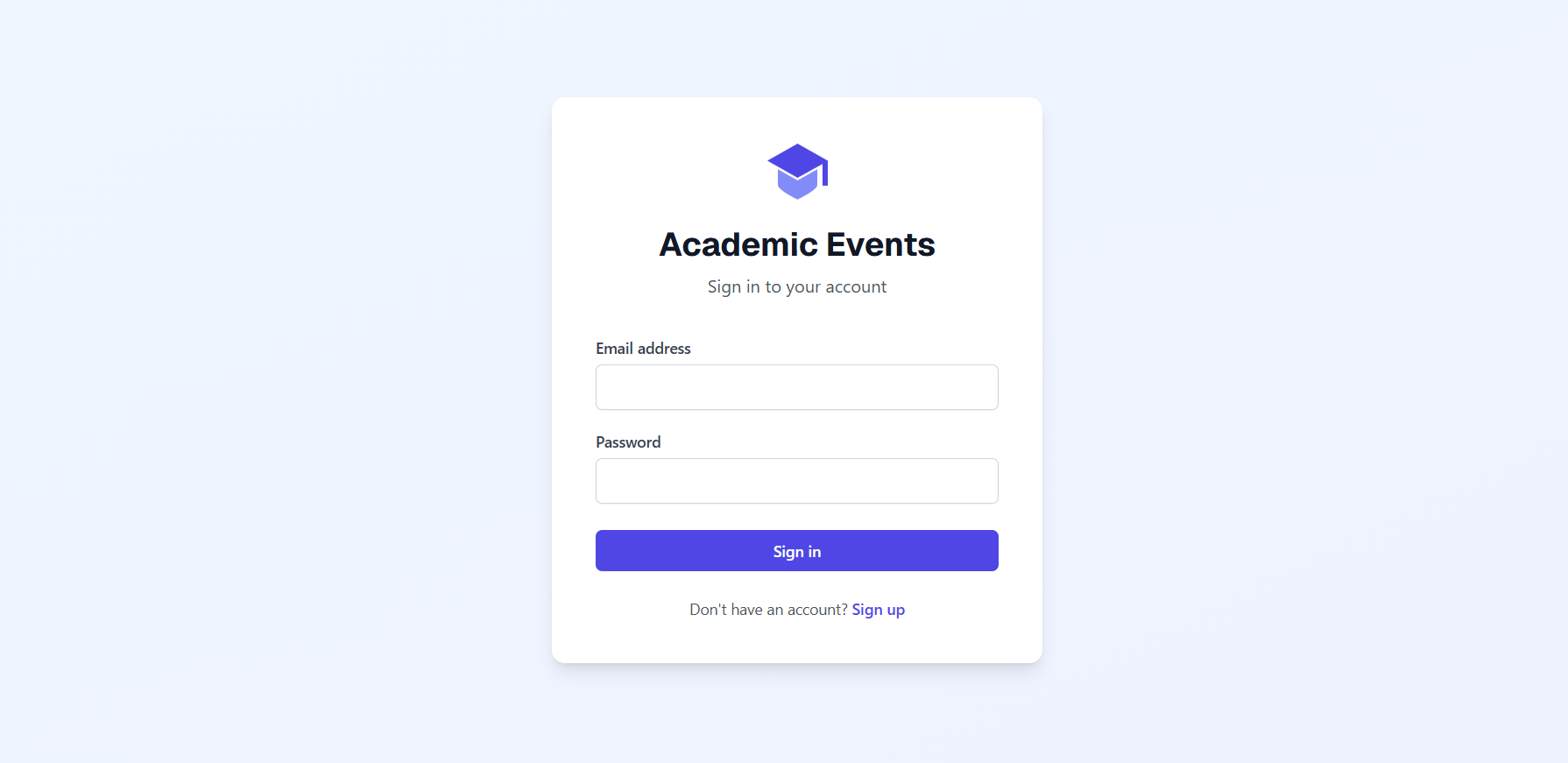
}

6. יש להגיש תיק משתמש הכולל צילומי מסך והסברים כיצד להשתמש באתר שלכם.

**תיק משתמש :**

**מסך ראשי:**

**בעת כניסת המשתמש לאתר, יופיע מסך של התחברות / הרשמה לאתר**

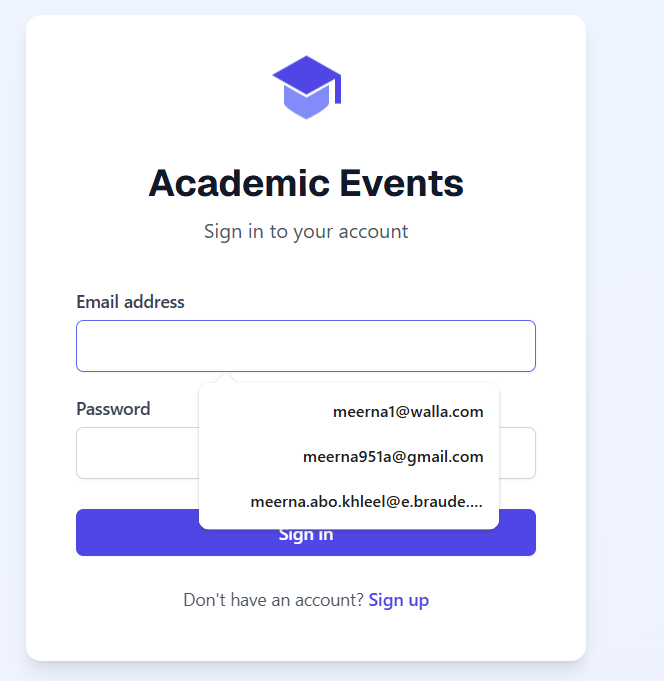
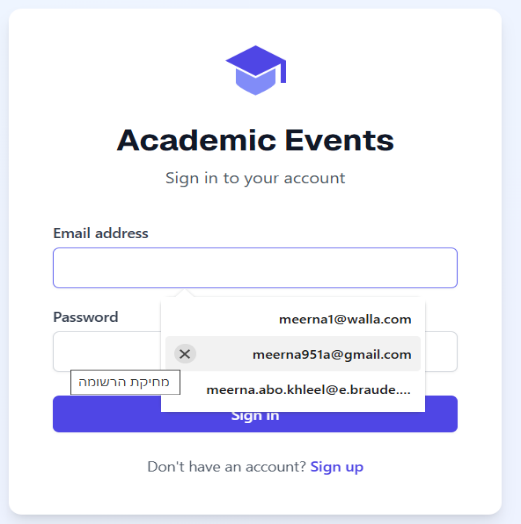
****

זה דף ההתחברות לפלטפורמת Academic Events הוא משמש כשער גישה לחשבון שלך לצורך חיפוש והצטרפות לאירועים אקדמיים.

1. שדות:
   * כתובת דוא"ל: הזן את כתובת הדוא"ל המשויכת לחשבונך.
   * סיסמה: הזן את הסיסמה שלך בצורה מאובטחת.
2. פעולות:
   * התחברות: לאחר שהזנת את הפרטים שלך, לחץ על כפתור התחברות

"Sign in " כדי לגשת לחשבון.

* + הרשמה: "Sign up" אם עדיין אין לך חשבון, לחץ על קישור הרשמה כדי ליצור אחד.

****

**הצעות אוטומטיות לכתובת דוא"ל :**

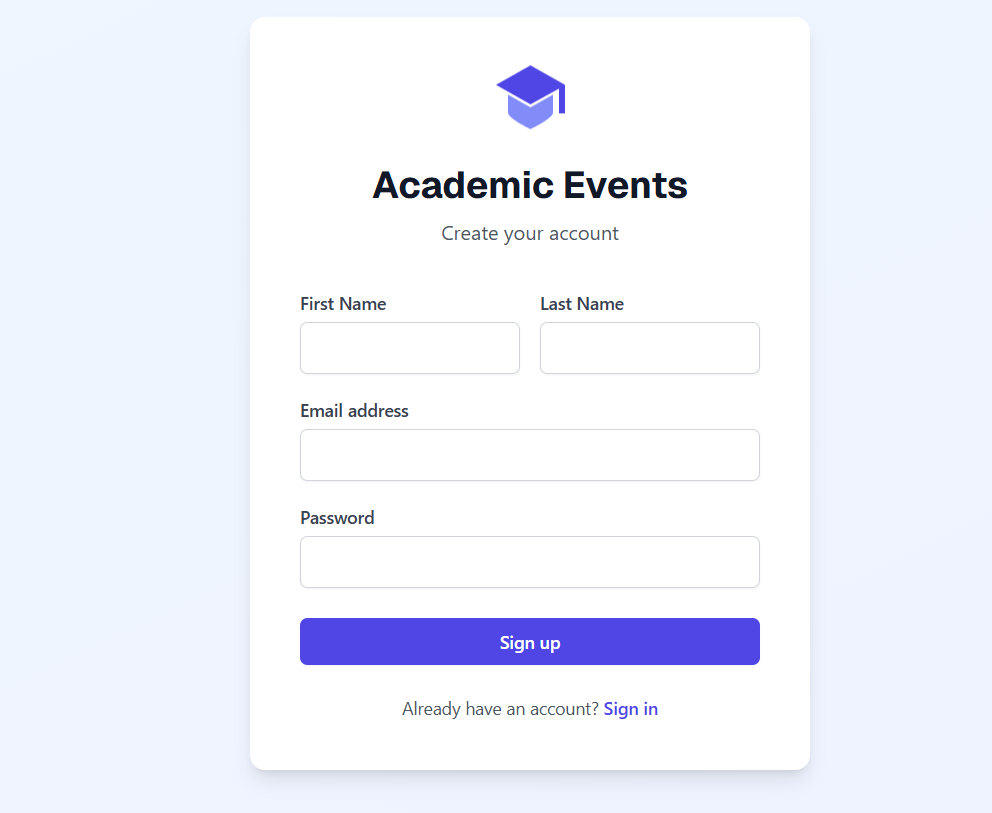
בעת הזנת כתובת דוא"ל בשדה ההתחברות, ייתכן שתראה רשימה של כתובות דוא"ל שהוזנו בעבר כהצעות. תכונה זו מסופקת על ידי הדפדפן שלך כדי לשפר את הנוחות ולאפשר בחירה מהירה של כתובת דוא"ל שהוזנה קודם לכן.

**מה כדאי לדעת:**

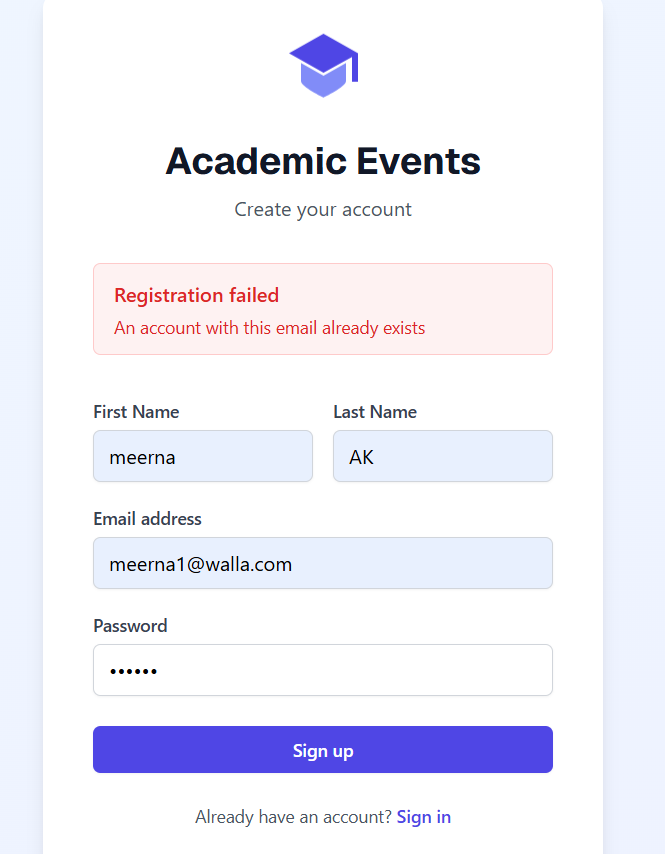
1. פרטיות: ההצעות נשמרות באופן מקומי בדפדפן שלך ואינן נשמרות בפלטפורמה. הן נראות רק במכשיר שלך.
2. בחירת כתובת דוא"ל: לחץ על כתובת הדוא"ל הנכונה מתוך הרשימה כדי לחסוך זמן.
3. ניהול ההצעות:

אם כתובת דוא"ל אינה נכונה או אינה מעודכנת, תוכל לנהל או למחוק את הכתובות השמורות דרך הגדרות הדפדפן שלך.

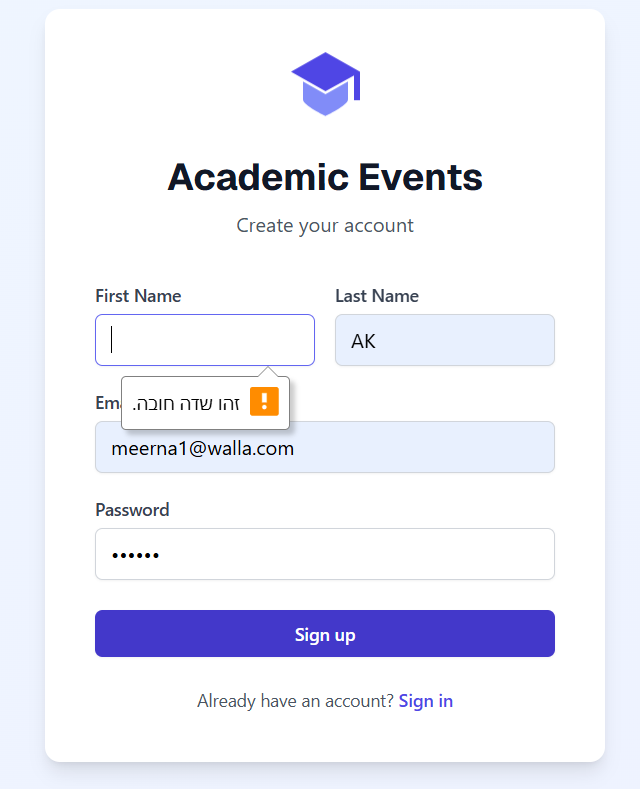
לאחר כניסה למסך ראשי, אם אין לך עדיין חשבון, תקיש על המילה "Sign up", יופיע לך את הדף הזה:

**דף ההרשמה**:

1. **מטרה**: דף זה מיועד ליצירת חשבון חדש בפלטפורמה.
2. **שדות למילוי**:
   * **First Name :** הזן את שמך הפרטי.
   * **Last Name** : הזן את שם המשפחה שלך.
   * **Email Address :** הזן כתובת דוא"ל חוקית שתחובר לחשבונך.
   * **Password :**  בחר סיסמה מאובטחת שתשמש אותך להתחברות.
   * **שם פרטי**: הזן את שמך הפרטי. ודא שהשדה לא נשאר ריק, כיוון שזהו שדה חובה.
3. **פעולות**:
   * **Sign Up :** לאחר מילוי כל השדות, לחץ על כפתור **"Sign Up"** כדי ליצור את חשבונך.
   * **Sign In :** אם כבר יש לך חשבון לחץ על המילה **"Sign In"** בתחתית הדף כדי לעבור לדף ההתחברות (לחזור לדף הראשון).
4. **הודעות שגיאה**:



* + **הרשמה נכשלה**: אם כתובת הדוא"ל כבר נמצאת בשימוש, תוצג הודעת שגיאה באדום: *"An account with this email already exists"* (חשבון עם כתובת דוא"ל זו כבר קיים). במקרה כזה, השתמש בכתובת דוא"ל אחרת או התחבר אם כבר יש לך חשבון.

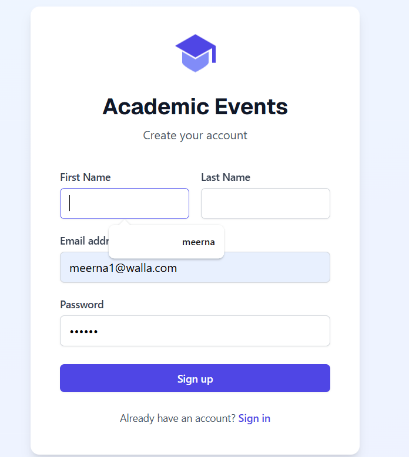


* + **שדה חובה**: אם שדה חובה (כמו שם פרטי) נשאר ריק, תוצג הודעה :

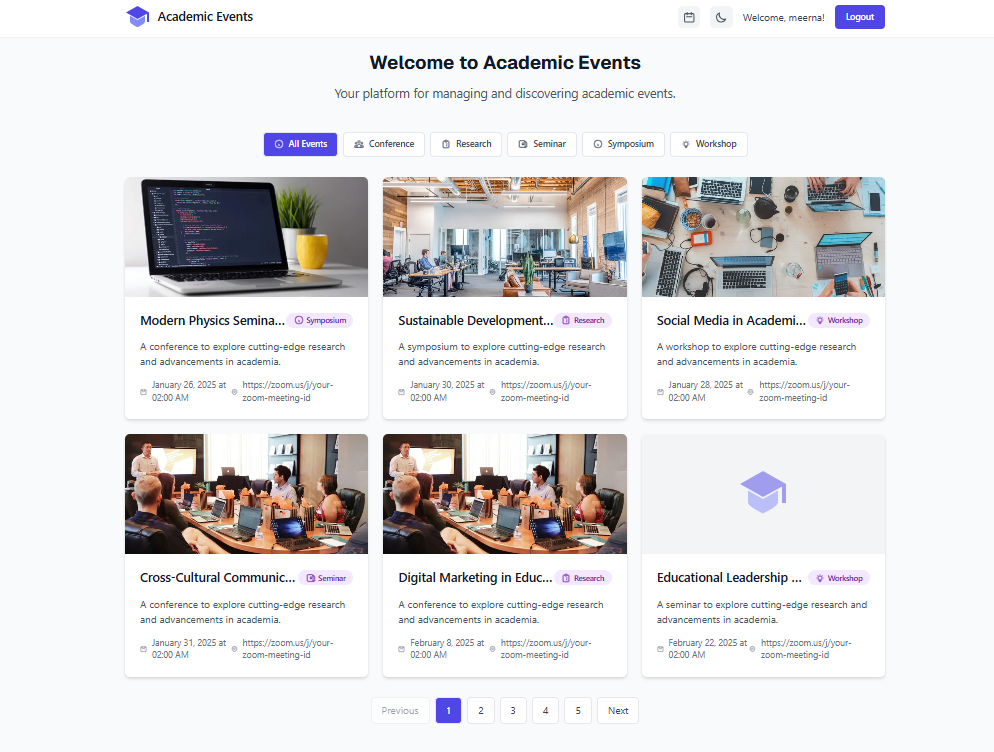
"*זהו שדה חובה."*

יש למלא את הפרטים החסרים כדי להמשיך.

1. **הערות :** 
   * בדוק שכל השדות מולאו באופן מדויק.
   * בחר סיסמה מאובטחת הכוללת שילוב של אותיות גדולות, אותיות קטנות, מספרים ותווים מיוחדים.
   * ייתכן שתראה רשימת השלמה אוטומטית בשדות שהוזנו בעבר כהצעות , כדי לשפר את הנוחות ולאפשר בחירה מהירה, לדוגמה:



אחרי הקשה על כפתור "Sign in" / "Sign up" (בכך שהכל תקין), יעבור לדף הראשי אחרי התחברות .



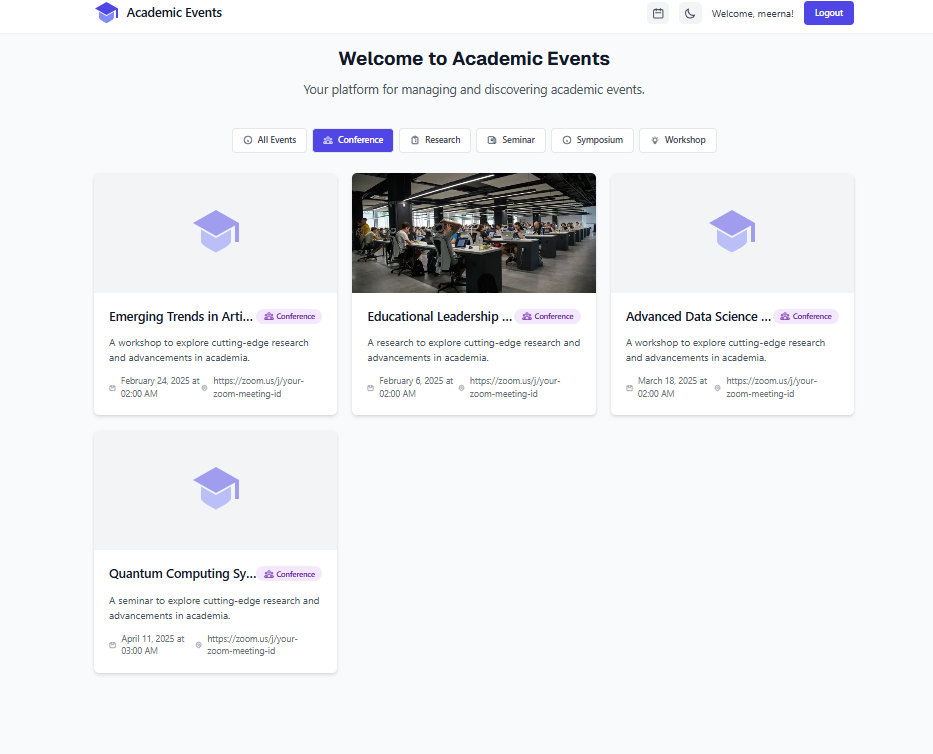
**דף רשימת האירועים(המסך הראשי של המשתמש)**

דף זה מהווה את העמוד המרכזי בפלטפורמה, שבו המשתמשים יכולים לבחור אירועים אקדמיים בהתאם להעדפותיהם, לסנן, לשנות לdark mode , לכנס ללוח שנה .

**תכונות מרכזיות:**

1. **ברוכים הבאים לפלטפורמת Academic Events**:  
   בחלק העליון של העמוד, ישנה כותרת שמסבירה בקצרה את מטרת הפלטפורמה: ניהול וגילוי אירועים אקדמיים.
2. **סרגל סינון אירועים**:
   * סרגל זה מציע אפשרויות לסינון האירועים על פי קטגוריות, והם:
     + **All Events** , **Conference**, **Research** , **Seminar , Symposium** , **Workshop**.
   * לחיצה על אחת הקטגוריות תציג רק את האירועים שמתאימים לאותה קטגוריה.

לדוגמה:



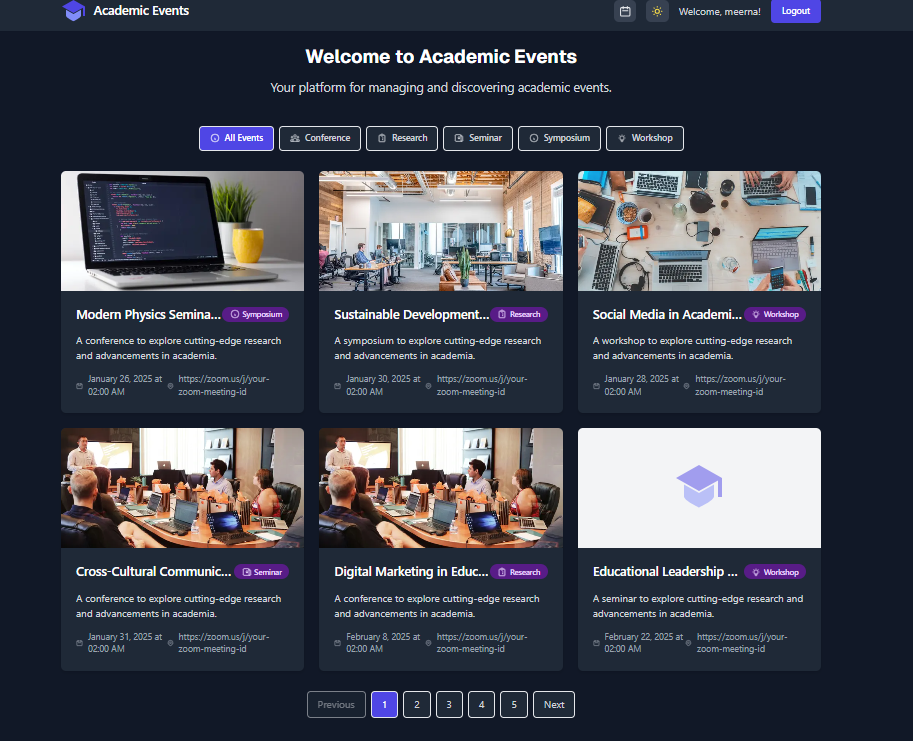
1. **רשימת האירועים (כרטיסי אירועים)**:
   * כל אירוע מוצג בכרטיס, המכיל:
     + **שם האירוע וסוגו**.
     + **תיאור קצר**: מתאר את נושא האירוע ומטרתו.
     + **תאריך ושעה**: מועד האירוע מוצג בצורה ברורה.
     + **קישור להשתתפות**: מדובר באירועים מקוונים.
2. **דפדוף עמודים**:
   * בתחתית העמוד יש פס ניווט בין עמודים, המאפשר למשתמשים לעבור בין עמודי האירועים במידה וישנם אירועים רבים.
3. **התאמת תצוגת מכשיר (F12)**:

* משתמשים יכולים ללחוץ על כפתור**F12** כדי לראות כיצד הפלטפורמה נראית בגדלי מסכים שונים. שם ניתן לדמות את תצוגת האתר על מכשירים שונים .
* תכונה זו מועילה במיוחד למשתמשים או למפתחים שרוצים לבדוק כיצד הפלטפורמה מותאמת למכשירים שונים.

**כפתורי ניווט בראש הדף:**

1. **כפתור מעבר dark mode**:
   * ממוקם בראש הדף, עם סמל של שמש או ירח.
   * לחיצה על כפתור זה מאפשרת מעבר בין מצב תאורה בהיר למצב כהה (ולהפך), בהתאם להעדפת המשתמש.

לדוגמה, מעבר למצב ככה :

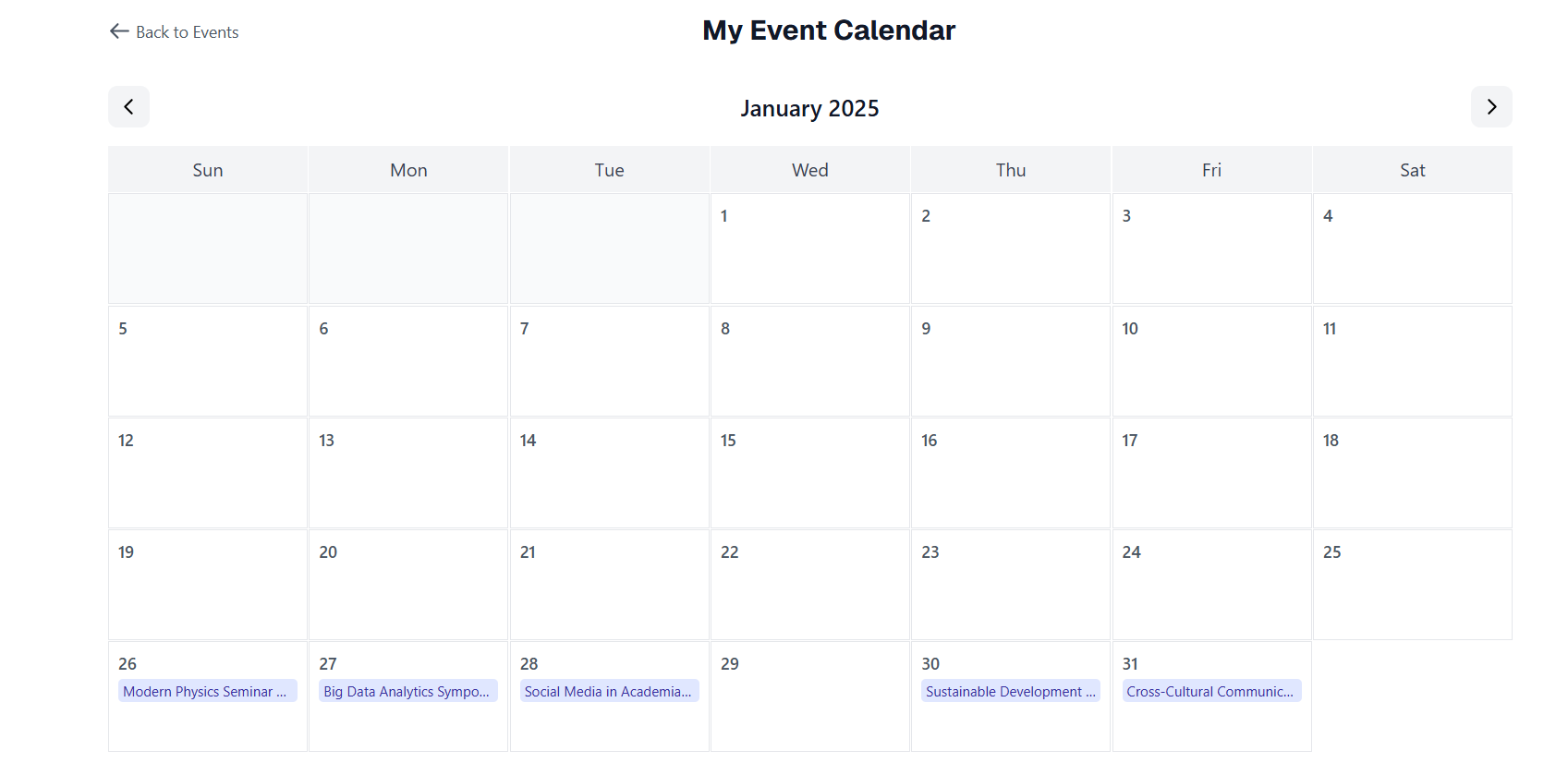


1. **כפתור לוח השנה (Calendar)**:
   * ממוקם ליד כפתור dark mode .
   * לחיצה עליו תוביל את המשתמש לדף **לוח השנה האישי**, שבו ניתן לראות את האירועים לפי תאריכים.
2. **כפתור התנתקות (Logout)**:
   * ממוקם בצד ימין העליון של הדף.
   * לחיצה עליו מאפשרת למשתמש להתנתק מהמערכת לאחר השימוש.

**הוראות שימוש:**

1. בחר את הקטגוריה המתאימה בעזרת כפתורי הסינון בראש הדף.
2. גלול בין כרטיסי האירועים כדי למצוא את האירוע שמעניין אותך.
3. לחץ על הקישור להשתתפות כדי להצטרף לאירוע המקוון בזמן הנקוב.
4. השתמש בפס הדפדוף בתחתית הדף כדי לעבור בין עמודים במידה וישנם אירועים נוספים.

לאחר לחיצה על כפתור לוח השנה, הדף יעבור ללוח השנה:

****

**דף לוח האירועים :**

**מטרה:**

דף לוח השנה נועד לספק למשתמשים תצוגה ברורה של כל האירועים לפי תאריכים.

**תכונות מרכזיות:**

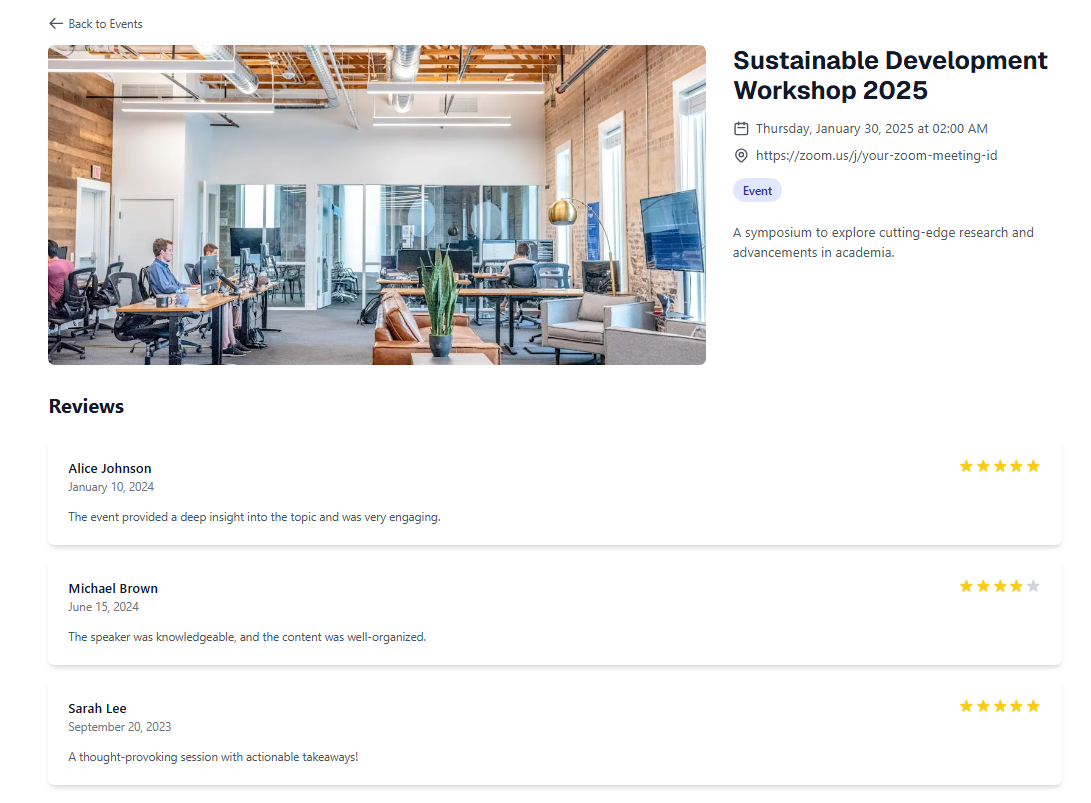
1. **לוח השנה**:
   * לוח השנה מציג את החודש הנוכחי (למשל, ינואר 2025) עם תאריכים ומידע על האירועים.
   * האירועים מופיעים כתיבות בתאריכים הרלוונטיים, וניתן לכנס לכל אירוע ואירע דרך לחיצה על האירוע עצמו.
2. **תצוגת אירועים**:
   * כל תיבה בלוח השנה מכילה:
     + **שם האירוע**.
     + **קישור מהיר**: לחץ על שם האירוע בתיבה כדי לעבור לעמוד הפרטים שלו.
3. **ניווט בין חודשים**:
   * ניתן לעבור בין חודשים באמצעות כפתורי החיצים ימין/שמאל בראש לוח השנה.

**הוראות שימוש:**

1. **מעקב אחר אירועים**:
   * בדוק את לוח השנה כדי לצפות באירועים שיש בחודשים הנוכחי/הקרוב.
   * לחץ על תיבת האירוע כדי לצפות בפרטי האירוע.
2. **ניווט בין החודשים**:
   * עבור בין חודשים בלוח השנה בעזרת כפתורי הניווט.
3. **כפתור חזרה לאירועים (Back to Events)**:

* ממוקם בפינה השמאלית העליונה של העמוד.
* לחץ על כפתור זה כדי לחזור לעמוד הראשי של **רשימת האירועים**, שם תוכל לגלות אירועים חדשים או לעיין בפרטים נוספים.

לחיצה על תיבת האירוע בלוח השנה/ בדף הראשי יעביר אותנו לדף חדש עם מידע על האירוע עצמו :



**עמוד פרטי האירוע הנבחר :**

**מטרה:**

עמוד זה מספק פרטים מלאים על אירוע נבחר, כולל מידע מעמיק על האירוע, תאריך ושעה, קישור להשתתפות, וביקורות של משתתפים קודמים.

**תכונות מרכזיות:**

1. **כותרת ותיאור האירוע**:
   * בראש העמוד בצד ימין מופיעים שם האירוע, תיאור קצר ומטרת האירוע.
2. **פרטי האירוע**:
   * **תאריך ושעה**: תאריך ושעת התחלה של האירוע מוצגים בצורה ברורה.
   * **קישור להשתתפות**: קישור המוביל לפלטפורמה מקוונת (ZOOM) שבאמצעותו ניתן להשתתף באירוע.
   * **סוג האירוע**: מצוין כסמל, כדי לעזור למשתמשים לזהות במהירות את הפורמט.
3. **תמונה חזותית**:
   * התמונה המרכזית מציגה אווירה או תוכן רלוונטי לאירוע, ומספקת הקשר ויזואלי.
4. **סקירת המשתמשים (Reviews)**:
   * רשימה של ביקורות ממשתתפים קודמים, כולל:
     + **שם המשתמש ותאריך הביקורת**.
     + **הערות המשתמש**: תובנות או חוויות הקשורות לאירוע.
     + **דירוג כוכבים**: מדרג את איכות האירוע על בסיס חוויות המשתתפים.
5. **כפתור חזרה לעמוד האירועים (Back to Events)**:
   * ממוקם בפינה העליונה השמאלית.
   * לחיצה עליו תחזיר את המשתמש לעמוד הראשי של רשימת האירועים.

**הוראות שימוש:**

1. **סקירת פרטי האירוע**:
   * קרא את התיאור והפרטים החשובים כדי להבין את נושא האירוע והזמן שבו יתקיים.
2. **השתתפות באירוע**:
   * לחץ על הקישור המופיע בעמוד כדי להצטרף לאירוע במועד המתאים.
3. **סקירת ביקורות**:
   * עיין בביקורות הקיימות כדי לקבל מידע נוסף על חוויות משתתפים קודמים.
4. **חזרה לעמוד הראשי**:
   * לחץ על כפתור **Back to Events** כדי לשוב לעמוד רשימת האירועים.