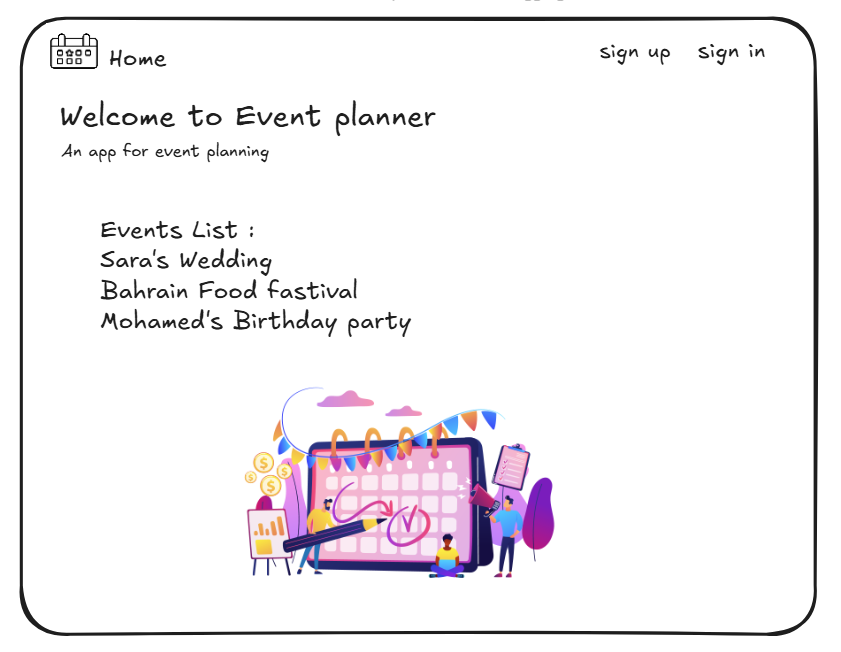
Event planning App : GA Project 3

# Mock-ups :



# Description :

The event planning application streamlines the process of scheduling gatherings, weddings, and business events. Users may easily create, change, and remove events using secure JWT-based user authentication, including crucial characteristics like as names, descriptions, dates, locations, and attendance lists . This app makes event planning simple, with a user-friendly interface that allows for seamless event management.

# ERD :

A screenshot of a computer

Description automatically generated

# User Stories :

* As a guest, I should be able to create an account.
* As a guest with an account, I should be able to log in to my account.
* As a user, I should be able to create an event .
* As a user, I should be able to read a list of all events .
* As a user, I should be able to view information about a single event along with the list of this event attendees .
* As a user, I should be able to add attendees to a specific event.
* As the planner of the event, I should be able to update that event.
* As the planner of the event, I should be able to delete that event.

# Pseudo code :

* Backend Pseudocode

User Authentication:

POST /register(username, password):

Store user in the database.

Return success message.

POST /login(username, password):

Validate user credentials.

Generate and return JWT token if valid.

Event Management:

POST /events (token, eventDetails):

Verify JWT token.

Create and store new event in the database.

Return success message.

GET /events (token):

Verify JWT token.

Retrieve and return list of all events.

GET /events/{eventID} (token):

Verify JWT token.

Fetch and return event details.

POST /events/{eventID}/attendees (token, attendeesList):

Verify JWT token.

If the user is the event planner, add attendees to event.

PUT /events/{eventID} (token, updatedDetails):

Verify JWT token.

If the user is the event planner, update event details.

DELETE /events/{eventID} (token):

Verify JWT token.

If the user is the event planner, delete the event.

Utility:

verifyToken(token) – Validate JWT and extract user info.

checkPlanner(token, eventID) – Check if the user is the planner of the event.

* Frontend Pseudocode

User Authentication:

register(username, password):

Call POST /register API to create account.

Display success or error message.

login(username, password):

Call POST /login API.

Store JWT token locally (e.g., in localStorage).

Redirect to dashboard.

Event Management:

createEvent(eventDetails):

Send POST /events request with JWT token and event details.

Display success or error message.

getAllEvents():

Send GET /events request with JWT token.

Display a list of events on the dashboard.

viewEvent(eventID):

Send GET /events/{eventID} request with JWT token.

Display event details and attendees.

addAttendees(eventID, attendeesList):

Send POST /events/{eventID}/attendees request with JWT token and attendees.

Update the UI with new attendees.

updateEvent(eventID, updatedDetails):

Send PUT /events/{eventID} request with JWT token and updated event details.

Update the UI with the new event details.

deleteEvent(eventID):

Send DELETE /events/{eventID} request with JWT token.

Remove the event from the UI.

Utility:

getAuthToken() – Retrieve JWT token from localStorage.

isLoggedIn() – Check if the user is logged in by checking for a valid JWT token.

## Back-End :

* Controller Folder :

events.js : CRUD ( Create , Read , Update and delete Events ) using routers .

profile.js : To verify token .

test-jwt.js : To get and verify token .

users.js : to post the sign in and sign up .

* Middleware Folder :

verify-token.js : for verify token function .

* Models folder :

event.js : for event and list schemas .

user.js : for userSchems .

* Server.js : for requiring and using libraries and controllers .

## Front-End :

* Src folder :

main.jsx : to import and show the App.jsx content .

App.jsx : for handle add , update and delete events .

App.css & index.css : add styling .

* Service folder :

authService.js : service for sign in and sign up .

eventService.js : service to show , create , update , delete events and attendees list .

* AttendeesForm : Adding attendees .
* Dashboard : show dashboard .
* EventDetails : show event details .
* EventForm : Add event .
* EventList : show events list .
* Landing : Landing page .
* Loading : to avoid app crush .
* NavBar : add nav bar .
* SigninForm : sign in the user .
* SignupForm : sign up the user .