Event Management System

1. Introduction

The **Event Management System** is a web-based application built using **Gin (Golang)** that allows users to create, manage, and participate in events. This system provides functionalities for:

- Creating, updating, and deleting events
- User registration and authentication
- Event registration and cancellation
- Admin functionalities for managing users

It is designed to be **scalable, efficient, and secure**, making use of **Gin**, a high-performance HTTP web framework for Golang. The system uses **MySQL** as its database for storing event and user-related data.

2. Objective

The **Event Management System** aims to provide a structured and efficient way to manage events by offering key functionalities such as event creation, registration, and user authentication. The main objectives are:

- To **simplify event management** by allowing users to create, update, and delete events.
- To **ensure security** by implementing user authentication (Signup/Login) and role-based access control.
- To **allow admin control**, enabling administrators to manage users and events efficiently.

This system is built using **Gin (Golang)** for high performance and uses **MySQL** for data storage to ensure reliability and consistency.

3. Scope of Event Management System

The Event Management System provides a backend API for managing events and user interactions. The system includes:

- **Event Management:** Create, update, delete, and view events.
- **User Authentication:** Signup, login, and secure access using JWT.

- **Admin Controls:** View and manage all users and events.
- **RESTful API:** Enables frontend integration for better usability.

4. Functional Requirements

- **User Authentication: Signup/Login with secure JWT authentication.**
- **Event Management:** Users can create, update, and delete events.
- **Event Registration:** Users can register for events.
- **Admin Controls:** Admins can manage users and delete events.
- **Database:** MySQL for storing users and events data
- **Secure Access Control:** Role-based authentication for users and admins.

5. Non-Functional Requirements

- **Performance:** System should handle multiple users without lag.
- **Scalability:** Support for growing numbers of users and events.
- **Security:** Secure authentication with JWT tokens.
- **Availability:** System should be available 24/7 with minimal downtime.

6. Use Cases

1. **User Use Cases:**
    - Sign up and log in to the system.
    - View available events.
    - Register or unregister for an event.

2. **Event Organizer Use Cases:**
    - Create, update, and delete events.
    - View registered users for their events.

7. Front-end

Currently, the system is backend-focused using Gin (Golang). However, the frontend can be developed using React.js, Angular, or Vue.js to provide a user-friendly interface.

8. Back-end

The backend is built using Golang with :

- **MySQL** for data storage
- **JWT Authentication** for security
- **REST API** architecture for easy integration

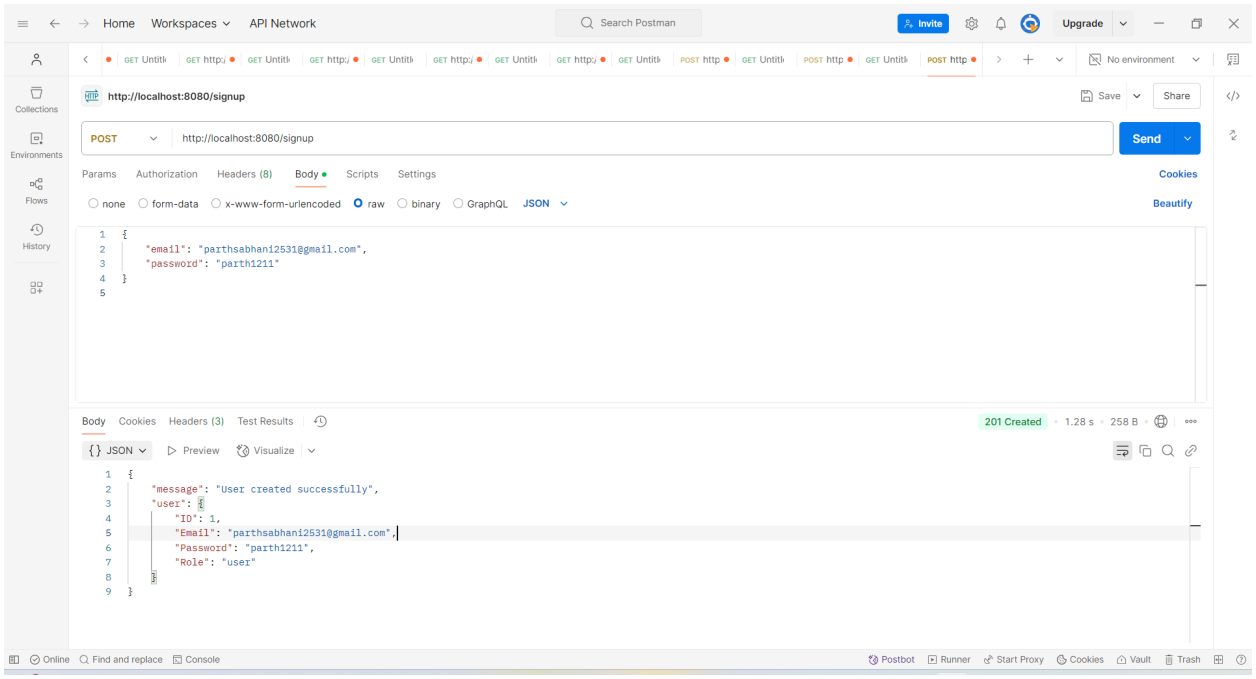
9. Advantages

- **Automation:** Reduces manual effort in event management.
- **Efficiency :** Streamlines event creation and user registration.
- **Security:** Provides authentication and role-based access control.
- **Scalability:** Can handle a large number of users and events.

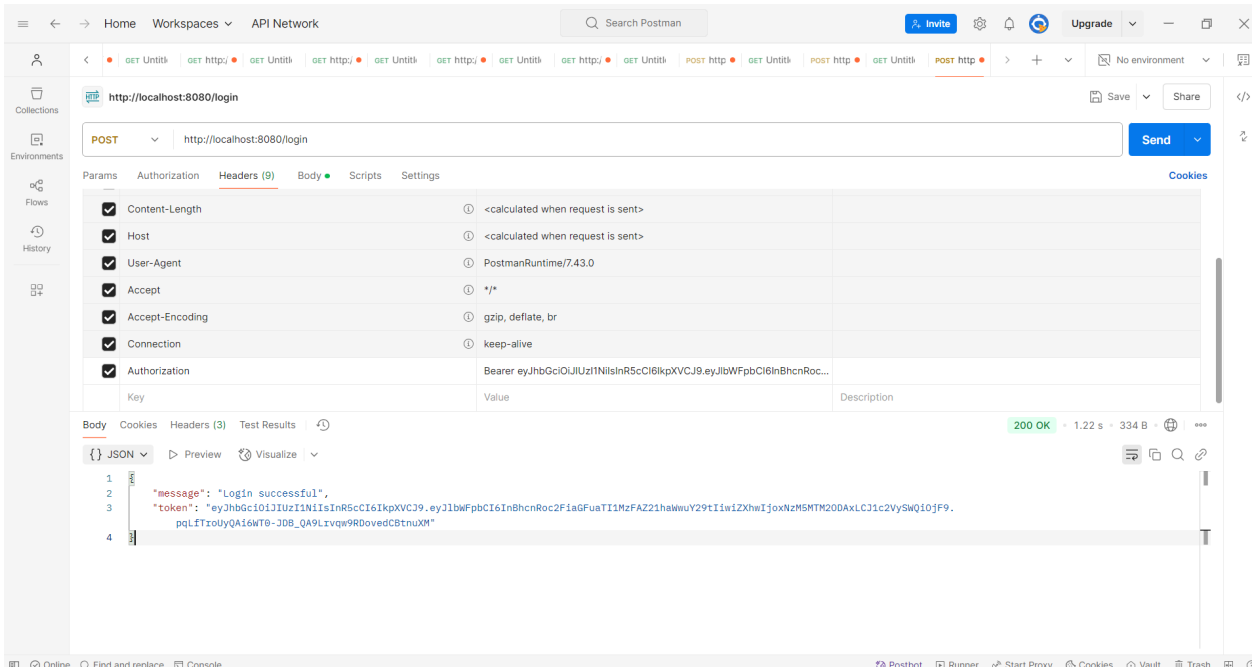
10. Disadvantages

- **Initial Setup Complexity:** Requires setting up a backend and database.
- **No Frontend UI Yet:** Currently, only API-based interactions are available.
- **Database Management Required:** Needs maintenance of MySQL database.

1.signup
202312024



2. Login



When the login is successful, the response includes a token.

This token is a JWT (JSON Web Token) and is used for authentication in protected API requests.