Event Management System

End User Analysis

- Organiser/Manager
- Attendee/Worker under Manager

Functional Requirements

- Create Events (For organisers only)
- Delete Events (For organiser only)
- Display if Events (All the participating events as organiser or participant)
- Edit Event (Date, Location, time, Status, Participants (Add or remove))
- View the KPI's (Total Participated Events, Ongoing Events, Pending Tasks)
- Profile View
- Dashboard view
- View the available tasks for the user
- Update the status of the assigned task
- Create new tasks
- View past events
- Create Volunteer Tasks (for managers only)
- Accept Volunteer tasks (for everyone including managers)
- Progress of the tasks (Progress bar)
- Edit tasks (By event manager only)

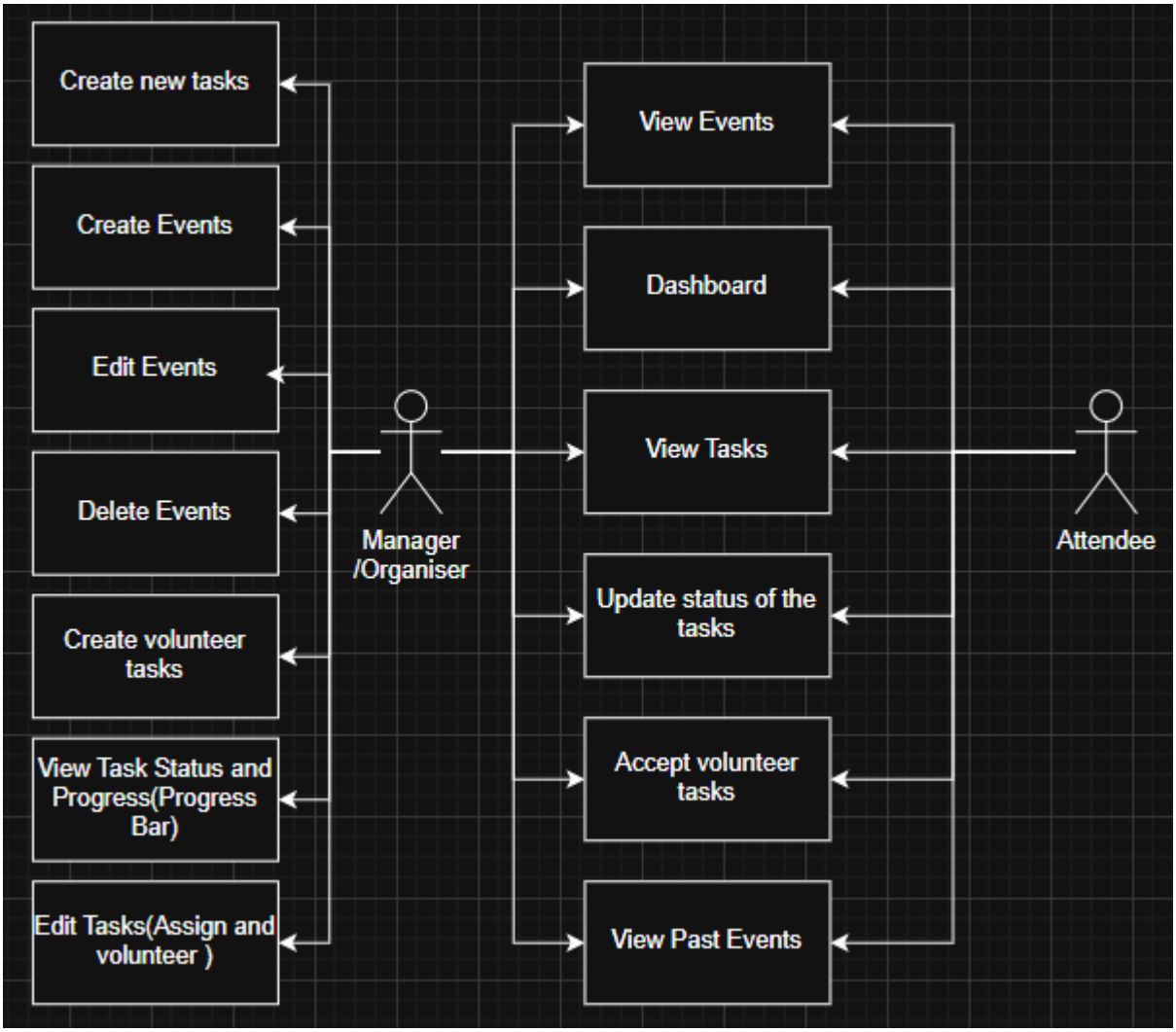
Non-Functional Requirements

- Authentication
- Authorisation
- Dynamic screen sizing (Compactable)
- Caching
- Input validations
- User Friendly Professional Interface
- Analytics and Reporting

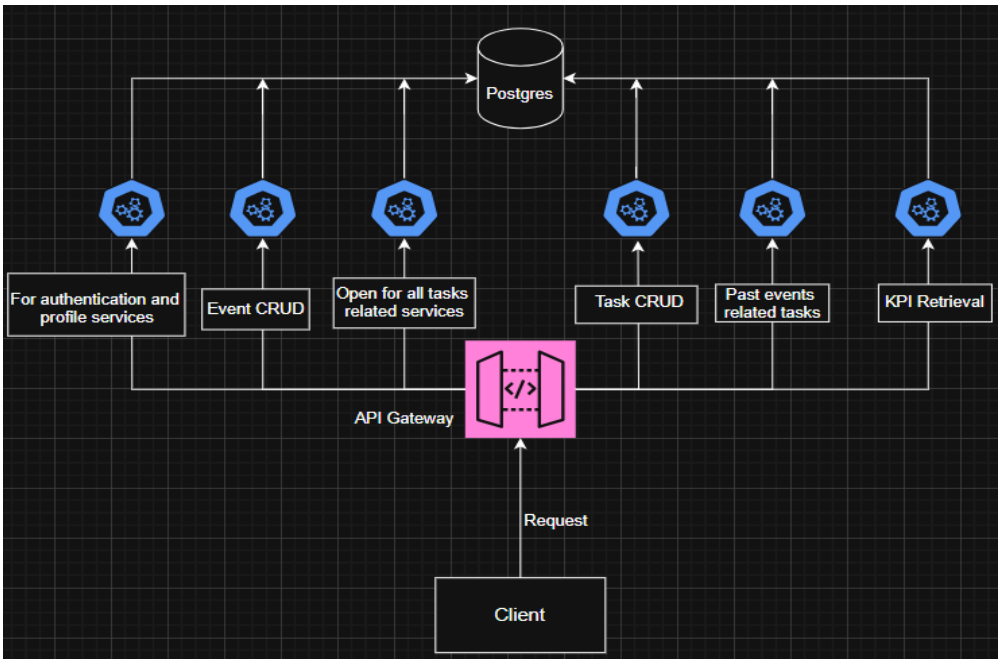
Software Requirements

- NextJS (Front End)
- FastAPI (Backend)
- Postgres (Database)
- Redis (Caching)
- PGadmin 4(For database visualisation)
- Postman/Swagger (API Testing)

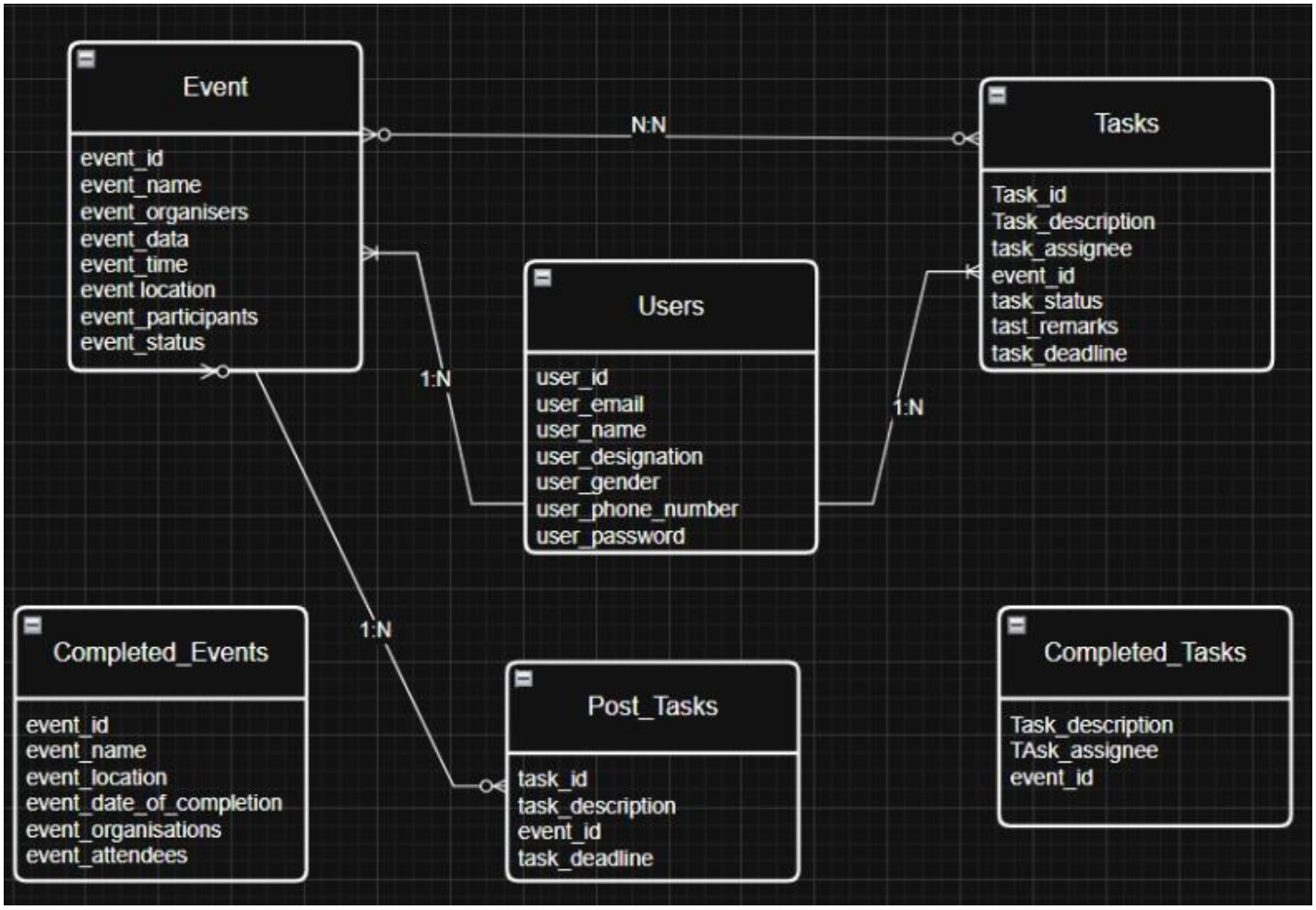
Use Case Diagram



Microservice Architecture



Database Schema



API Data

- KPI Microservice
  - /kpi/get_events – This is a GET method for retrieving all the data from the events_table and completed_ events and returns the Total_events_participated and ongoing_events count which the logged-in user is a part of.
- Accounts_Service Microservice
  - /accounts/authenticate – The is a POST method for authentication of the credentials of the user credentials i.e. email and password
  - /accounts/profile – This is a GET method which is used to retrieve all the details of the user except the password from the users table in a JSON format
- Events Microservice
  - /events/create_events – This is a POST method which takes in the data from the client end through the request and creates a new event in the database
  - /events/user_events – This is a GET method that takes in the parameter user_id and returns all the events in which the user is either oraganiser or participant
  - /events/event/{event_id} – This is a DELETE method which takes in event_id and delets the data from the table events

- Open_Events Microservice
  - /open/post_tasks – This is a POST method which takes in task_description,event_id,task_deadline and creats a volunteer task and posts in the notification section
  - /open/post_tasks – This is a GET method which retrieves all the volunteer tasks present in the post_tasks table and returns to the client
  - /open/accept_tasks – This is a POST method which takes in the user_id and assigns the volunteer task to the user id and transfers the volunteer task from post_task table to the tasks_table
- Past Microservice
  - /past/events-completed – This is a GET method which takes returns all the deleted/completed/past registered events from the database and returns to the client
- Tasks Microservice
  - /tasks/tasks – This is a POST method which creates a new task and this is only accessible for the organiser
  - /tasks/{task_id} – This is a PUT method which is used to update the table tasks after the changes are done from the client end
  - /tasks/tasks/{user_id} or {event_id} – This is a GET method that is used to retrieve all the data from the tasks table based on the passed parameter i.e. user_id or event_id