**THE UNIVERSITY OF DODOMA**



**COLLEGE OF INFORMATICS AND VIRTUAL EDUCATION**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**GROUP ASSIGNMENT REPORT**

PROJECT TITLE: **WiPlan THE EVENT MANAGEMENT SYSTEM**

COURSE NAME: **DATABASE MANAGEMENT SYSTEMS [CP 224]**

DEGREE PROGRAM: **CYBER SECURITY AND DIGITAL FORENSICS ENGINEERING [CSDFE 2]**

**GROUP MEMBERS**

|  |  |  |
| --- | --- | --- |
| **S/N** | **FULL NAME** | **REGISTRATION NUMBER** |
| 1 | MESHACK EDWARD CHANAI | T22-03-03833 |
| 2 | TABU WASHA CHANANGWA | T22-03-00017 |
| 3 | BARAKA KIBELA | T21-03-03732 |
| 4 | WALTER A. MASSAWE | T22-03-09382 |
| 5 | QUEEN MSEMO ELISAFI | T22-03-05883 |
| 6 | MUSLIM ISSA SALUM | T22-03-04476 |
| 7 | DERICK KILIMA | T22-03-04005 |
| 8 | JOEL RICHARD TIMOTH | T22-03-07239 |
| 9 | JOSEPH MAKOBA MASINGIRI | T22-03-02072 |
| 10 | FAIDA KHAMIS MOH'D | T22-03-04564 |
| 11 | KASHILIMU PHABIAN MADUKA | T22-03-06782 |
| 12 | NOAH JONAH NYARI | T22-03-04065 |
| 13 | ALEX KUNDANDUMI KAAYA | T22-03-01731 |
| 14 | EMMANUEL GEORGE SHIRIMA | T22-03-05937 |
| 15 | AGNESS NYAKWAKA | T22-03-08360 |

Table of Contents

[EVENT MANAGEMENT SYSTEM FUNCTIONALITIES 5](#_Toc169038739)

[Before Event we have features like 5](#_Toc169038740)

[During Event we have features like 6](#_Toc169038741)

[After Event 6](#_Toc169038742)

[Front-end 8](#_Toc169038743)

[Back-end 8](#_Toc169038744)

[Component Interaction 8](#_Toc169038745)

[Entities and Relationships 9](#_Toc169038746)

[ENTITY RELATIONSHIP DIAGRAM 11](#_Toc169038747)

[Registering users 18](#_Toc169038748)

[Create event 19](#_Toc169038749)

[View Upcoming Events and Register to event 20](#_Toc169038750)

[View Event 20](#_Toc169038751)

[Managing events 21](#_Toc169038752)

[View all the created events 21](#_Toc169038753)

[Edit Created Event 22](#_Toc169038754)

[View Edited event 23](#_Toc169038755)

[View Registered users 23](#_Toc169038756)

# **INTRODUCTION**

In today's world, there are many events that run our lives be it a company conferences, weddings, music festivals, academic gatherings or community gatherings they all demand some planning and professional organization for better and seamless execution.

An Event Management System we call it **WiPlan** emerges as an important tool, streamlining the main processes involved in organizing, managing, and evaluating events of all scales and types.

The primary objective of an WiPlan is to facilitate the end-to-end management of events, encompassing aspects such as scheduling, registration, ticketing, venue management, marketing, and post-event analytics.

By integrating various functionalities into this platform, WiPlan significantly reduces the administrative burden on event planners, enhances attendee experience, and ensures that every event component is synchronized and executed flawlessly.

This documentation shows the comprehensive features and benefits of an Event Management System. It provides a detailed overview of the core modules, main database and user interface, as well as practical insights into its implementation and usage.

Whether you are an event manager seeking to optimize your workflow, a developer aiming to understand the system's technical intricacies, or a stakeholder interested in the system's value proposition, this report serves as an essential guide to understanding and leveraging WiPlan effectively.

By exploring the functionalities and advantages of an Event Management System, we aim to illuminate how technology can transform the event management providing efficiency, creativity, and engagement in event planning and execution.

# **Purpose and Scope**

The Event Management System (WiPlan) is designed to provide a comprehensive solution for planning, organizing, and managing events of various scales and types.

The purpose of the WiPlan is to:

1. Make easy and fast the event planning process
2. Enhance coordination and communication among stakeholders
3. Improve the efficiency and accuracy of event-related tasks
4. Deliver a superior experience for attendees, sponsors, and organizers

## EVENT MANAGEMENT SYSTEM FUNCTIONALITIES

The event management system covers the entire event lifecycle, from planning and promotion to execution and follow-up.

This section cover the functionalities offered by the system, categorized by the different stages of your event:

* Before Event
* During Event
* After Event.

By looking at these features, we can efficiently manage every aspect of events and ensure a successful experience for both organizers and attendees.

## Before Event we have features like

1. **Event Creation functionalities**

Defining event details including **name**, **description**, **venue**, **date**, **time**, **duration**, and **speaker** **information**.

Specifying if the event is **free** or **paid**.

(Optional) Include a section for "What to Bring" to inform attendees of any necessary items.

1. **Agenda Management functionalities**

Creating and managing an agenda for the event, including **topics**, **timings**, and **speaker** **allocations**.

1. **Event Mode functionalities**

Setting the event mode as "Before Event" initially when the event has not yet started.

Implementing a functionality to switch the mode to "During Event" once the event commences and set features like agendas and event time table displaying.

1. **Registration and Barcodes functionalities**

Allowing attendees to register for the event through the system.

Generating unique barcodes for registered attendees.

Providing options to print or download barcodes for verification.

Ensuring barcode access is restricted to registered attendees only.

1. **Attendance Tracking functionality**

Enabling organizers to track attendee check-in using barcodes which is useful for events where attendance needs to be monitored.

1. **Food Management functionality**

Creating menus for the event, specifying whether food options are **free** or **paid**.

## During Event we have features like

1. **Event Mode functionality**

The system should automatically switch to "During Event" mode upon starting the event and display in-event features like time tables for whatever is going to happen, agendas.

1. **Agenda Management/adjustment functionality**

Allowing adjustments to the agenda in real-time during the event (such as adding topics, revising timings).

1. **Speaker Updates functionality**

Providing a platform to announce any last-minute changes to the speaker schedule.

1. **Audience Engagement functionality**

Implementing features for audience participation, including:

* 1. Submitting comments or questions on specific topics.
  2. Up-voting questions to prioritize them for discussion.
  3. Conducting polls or votes if required.

1. **Food Ordering functionalities (Optional)**

For compact venues, offer a feature for attendees to order food directly through the system during the event.

Ensure menus are readily accessible within the ordering functionality.

## After Event

1. **Attendance Analysis (Optional):**

Generating reports analyzing attendee demographics and check-in data (if attendance tracking was enabled).

1. **Feedback Collection:**

Facilitating the collection of feedback from attendees through forms or comments.

1. **Event Wrap-up:**

Allowing organizers to compose and send thank-you notes or updates or notification about future events to attendees

# **SYSTEM ARCHITECTURE**

The system architecture of the Event Management System, WiPlan, is designed to integrate various components seamlessly to provide comprehensive event management functionalities. The architecture can be divided into three main layers:

* User Interface (UI),
* Application Logic,
* Database.

## ****Front-end****

**HTML:** Provides the core structure and content of web pages.

**Bootstrap:** A CSS framework for creating responsive and mobile-friendly user interfaces.

## ****Back-end****

**PHP:** A server-side scripting language that processes user requests and interacts with the database.

**PostgreSQL:** An open-source relational database management system (RDBMS) that stores and manages event data.

## Component Interaction

**User Interface (UI):**

Developed using HTML and Bootstrap, the UI offers a user-friendly interface for users to interact with the system.

Features such as event creation, registration, and attendance tracking are accessible through the UI.

**PHP Scripts:**

These server-side scripts are triggered by user interactions with UI elements (e.g., clicking a button, submitting a form).

Tasks handled by PHP scripts include validating user input, communicating with the database to perform CRUD operations, and generating dynamic content for the UI.

**PostgreSQL Database:**

Acts as the central storage for all event-related data.

PHP scripts interact with the database to manage information on events, user accounts, and other relevant data.

# **ENTITY RELATIONSHIP DIAGRAM**

The Entity Relationship Diagram (ERD) is a graphical representation of the entities in the Event Management System and the relationships between them. This diagram is essential for understanding how data is structured and how different parts of the system interact with each other.

## Entities and Relationships

The primary entities in the system include Events, Users, Registrations, Agendas, Announcements, Attendance, Feedback, and Subscriptions. Below is a detailed explanation of each entity and its relationships.

**Events**

**Represents an event created within the system.**

**Attributes:** event\_id, name, description, venue, time, duration, mode, status, organizer\_id, image\_url

**Relationships**

One-to-Many with Users (an event is organized by one user, but one user can organize many events).

One-to-Many with Registrations (an event can have many registrations).

One-to-Many with Agendas (an event can have multiple agenda items).

One-to-Many with Announcements (an event can have multiple announcements).

One-to-Many with Attendance (an event can have multiple attendance records).

One-to-Many with Feedback (an event can have multiple feedback entries).

**Users**

**Represents a user of the system, either an attendee or an organizer.**

**Attributes:** user\_id, username, password, email, age, privileges

**Relationships**

One-to-Many with Events (a user can organize multiple events).

One-to-Many with Registrations (a user can register for multiple events).

One-to-Many with Feedback (a user can leave feedback for multiple events).

**Registrations**

**Represents the registration of a user for a specific event.**

**Attributes:** registration\_id, event\_id, user\_id, barcode

**Relationships:**

Many-to-One with Events (multiple registrations can be associated with a single event).

Many-to-One with Users (multiple registrations can be associated with a single user).

**Agendas**

**Represents the agenda items for a specific event.**

**Attributes:** agenda\_id, event\_id, topic, start\_time, end\_time, speaker

**Relationships:**

Many-to-One with Events (multiple agenda items can be associated with a single event).

**Announcements**

**Represents announcements made for a specific event.**

**Attributes:** announcement\_id, event\_id, title, message, timestamp

**Relationships:**

Many-to-One with Events (multiple announcements can be associated with a single event).

**Attendance**

**Tracks the attendance of users at events.**

**Attributes:** attendance\_id, event\_id, user\_id, check\_in\_time, check\_out\_time

**Relationships:**

Many-to-One with Events (multiple attendance records can be associated with a single event).

Many-to-One with Users (multiple attendance records can be associated with a single user).

**Feedback**

**Represents feedback given by users for events.**

**Attributes:** feedback\_id, event\_id, user\_id, rating, comment, timestamp

**Relationships:**

Many-to-One with Events (multiple feedback entries can be associated with a single event).

Many-to-One with Users (multiple feedback entries can be associated with a single user).

**Subscriptions**

**Represents a user's subscription to an event for updates and notifications.**

**Attributes:** subscription\_id, user\_id, event\_id, subscription\_date

**Relationships:**

Many-to-One with Users (multiple subscriptions can be associated with a single user).

Many-to-One with Events (multiple subscriptions can be associated with a single event).

## ENTITY RELATIONSHIP DIAGRAM

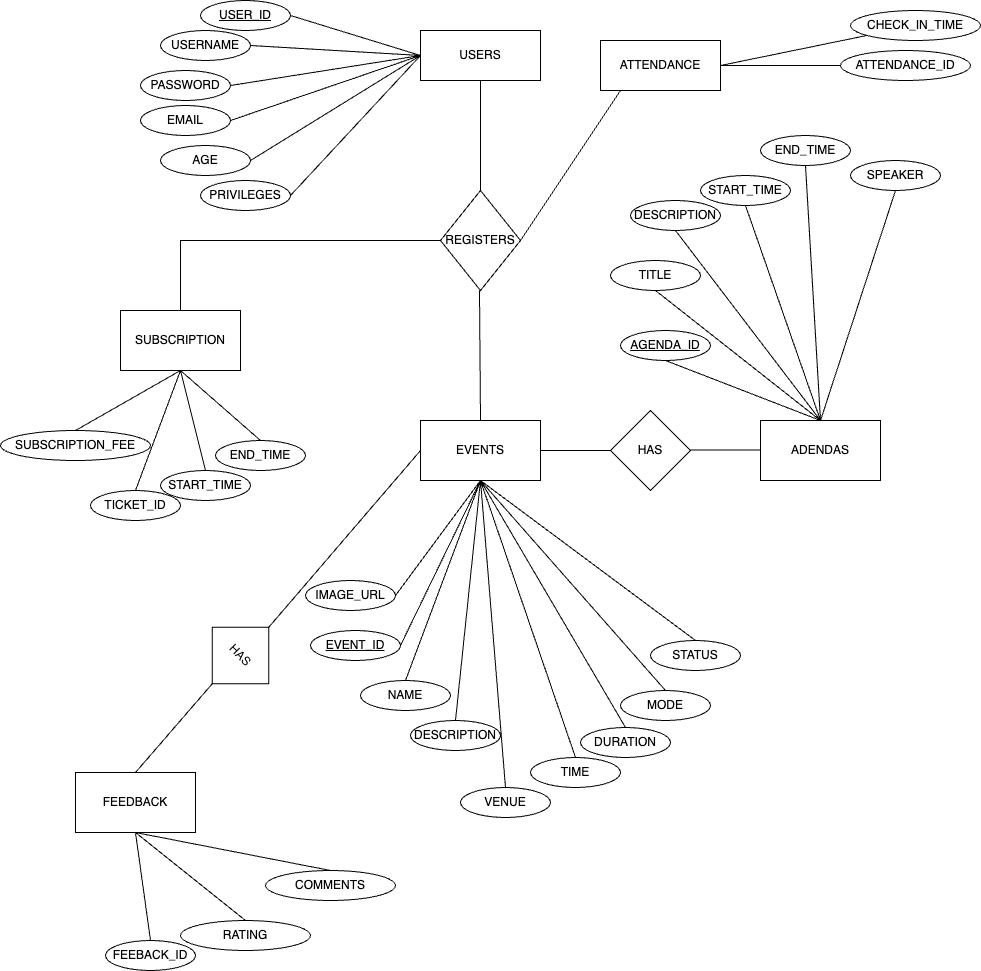


Figure 1: event\_management entity relationship diagram

# **RELATIONAL SCHEMA**

The relational schema of the Event Management System WiPlan outlines the organization of data into tables and defines the relationships and constraints among them. This structured format ensures data integrity and efficient data management.

**Relational Schema**

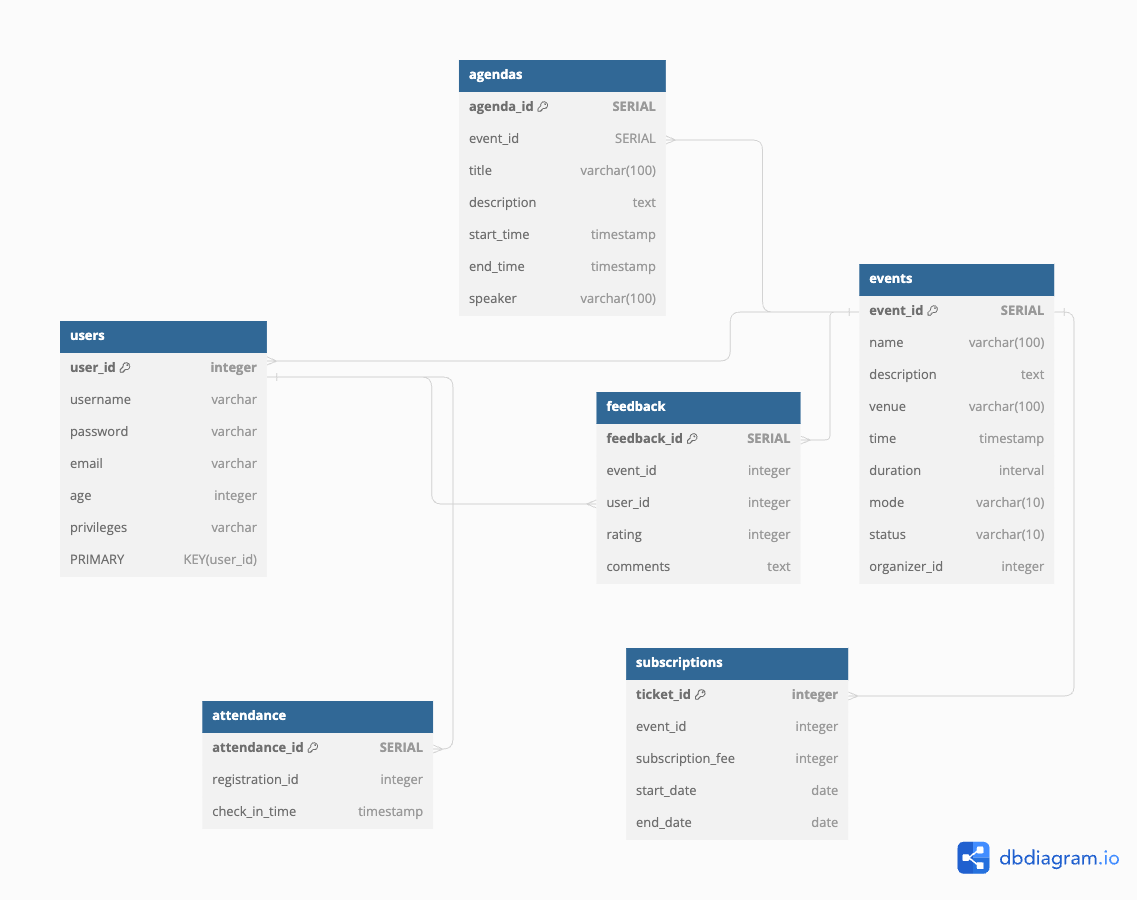


Figure 2: event\_management relation schema

# **NORMALIZATON STEPS**

As we are supposed to put the tables for our database in the third normal formal, here we are going to show that all the necessary rules of dependency have been sustained and hence to prove that our database is normalized to the third normal form.

**Unnormalized form**

We first created tables that were to take a lot of data at once. For example, one central table would take all users, events and organizers in one table. This data was a large cluttered table.

**The first normal form**

This is achieved as all the tuples in our tables show that they all contain atomic values.

Example; from the table users, we will examine it to see that it has achieved the first normal form.

CREATE TABLE users (

user\_id SERIAL NOT NULL,

username varchar (50) NOT NULL,

password varchar (255) NOT NULL,

email varchar (100) NOT NULL,

age integer NOT NULL,

privileges varchar (50) NOT NULL,

PRIMARY KEY (user\_id)

);

This shows that all tuples are atomic hence the first normal from has been achieved.

**The second normal form**

This is achieved in our database as all our tables show that there is no existence of partial dependency and thus ensuring that all non-key attributes are fully functional dependent on the primary key.

Example: from the table users, we will examine it to see that it has achieved the second normal form.

From the query, we notice that there is no partial dependency among any of the attributes in the table and that all non-key attributes (privileges, age) and all other attributes do depend on the primary key hence the second normal form is achieved.

**The third normal form**

This is achieved by ensuring that there aren’t any transitive dependencies and ensuring that all attributes in a table depend on the primary key.

Example: from the table users, we will examine it to see that it has achieved the third normal form.

CREATE TABLE users (

user\_id SERIAL NOT NULL,

username varchar (50) NOT NULL,

password varchar (255) NOT NULL,

email varchar (100) NOT NULL,

age integer NOT NULL,

privileges varchar (50) NOT NULL,

PRIMARY KEY (user\_id)

);

From the user table we find out that there is not transitive dependency and all the attributes do depend on the primary key thus ensuring that the table is in the third normal form.

All this has been evidenced through the investigation of the sql scripts written to create the various tables of our event\_management system.

# **SQL SCRIPTS**

CREATE TABLE agendas (

agenda\_id SERIAL NOT NULL,

event\_id SERIAL NOT NULL,

title varchar (100) NOT NULL,

description text NOT NULL,

start\_time timestamp without time zone NOT NULL,

end\_time timestamp without time zone NOT NULL,

speaker varchar (100) NOT NULL,

PRIMARY KEY (agenda\_id),

CONSTRAINT agendas\_event\_id\_fkey FOREIGN key(event\_id) REFERENCES events(event\_id) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE announcements (

announcement\_id SERIAL NOT NULL,

event\_id integer,

content text NOT NULL,

timestamp timestamp without time zone NOT NULL,

PRIMARY KEY (announcement\_id),

CONSTRAINT announcements\_event\_id\_fkey FOREIGN key(event\_id) REFERENCES events(event\_id) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE attendance (

attendance\_id SERIAL NOT NULL,

registration\_id integer,

check\_in\_time timestamp without time zone NOT NULL,

PRIMARY KEY (attendance\_id),

CONSTRAINT attendance\_registration\_id\_fkey FOREIGN key(registration\_id) REFERENCES registrations(registration\_id) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE events (

event\_id SERIAL NOT NULL,

name varchar (100) NOT NULL,

description text NOT NULL,

venue varchar (100) NOT NULL,

time timestamp without time zone NOT NULL,

duration interval NOT NULL,

mode varchar (10) NOT NULL,

status varchar (10) NOT NULL,

organizer\_id integer,

image\_url text,

PRIMARY KEY (event\_id),

CONSTRAINT events\_organizer\_id\_fkey FOREIGN key(organizer\_id) REFERENCES users(user\_id) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE feedback (

feedback\_id SERIAL NOT NULL,

event\_id integer,

user\_id integer,

rating integer NOT NULL,

comments text,

PRIMARY KEY (feedback\_id),

CONSTRAINT feedback\_event\_id\_fkey FOREIGN key(event\_id) REFERENCES events(event\_id) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT feedback\_user\_id\_fkey FOREIGN key(user\_id) REFERENCES users(user\_id) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE registrations (

registration\_id SERIAL NOT NULL,

event\_id integer,

user\_id integer,

barcode varchar (50) NOT NULL,

PRIMARY KEY (registration\_id),

CONSTRAINT registrations\_event\_id\_fkey FOREIGN key(event\_id) REFERENCES events(event\_id) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT registrations\_user\_id\_fkey FOREIGN key(user\_id) REFERENCES users(user\_id) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE subscriptions (

ticket\_id SERIAL NOT NULL,

event\_id SERIAL NOT NULL,

subscription\_fee numeric (10,2) NOT NULL,

start\_date date NOT NULL,

end\_date date NOT NULL,

PRIMARY KEY (ticket\_id),

CONSTRAINT subscriptions\_organizer\_id\_fkey FOREIGN key(event\_id) REFERENCES users(user\_id) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE users (

user\_id SERIAL NOT NULL,

username varchar (50) NOT NULL,

password varchar (255) NOT NULL,

email varchar (100) NOT NULL,

age integer NOT NULL,

privileges varchar (50) NOT NULL,

PRIMARY KEY (user\_id)

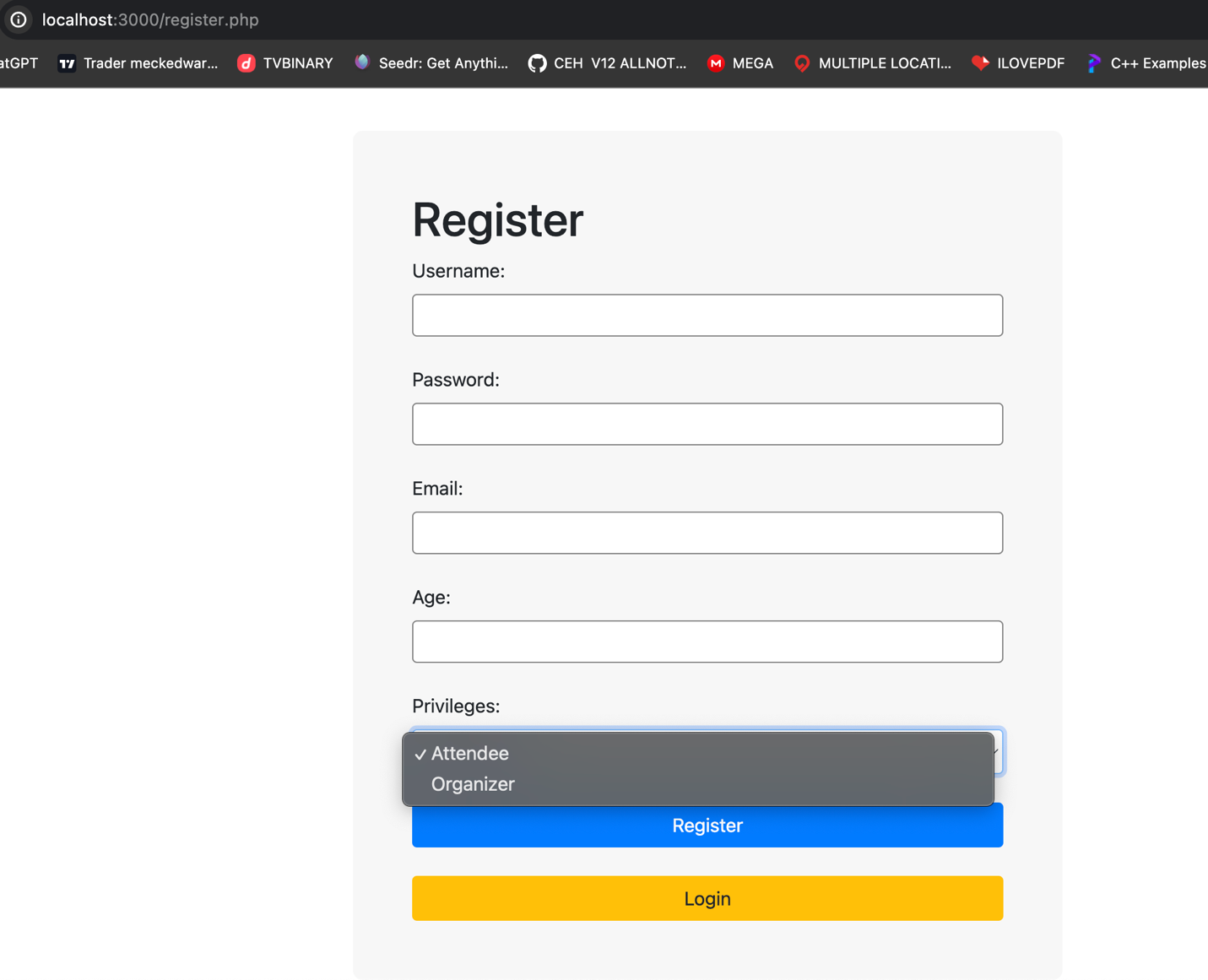
);

# **REQUIREMETS MET**

We have managed to meet a great number of requirements for our system to interact with the user in various way to meet their end goal and directives, the following are the met requirements in details:

## Registering users

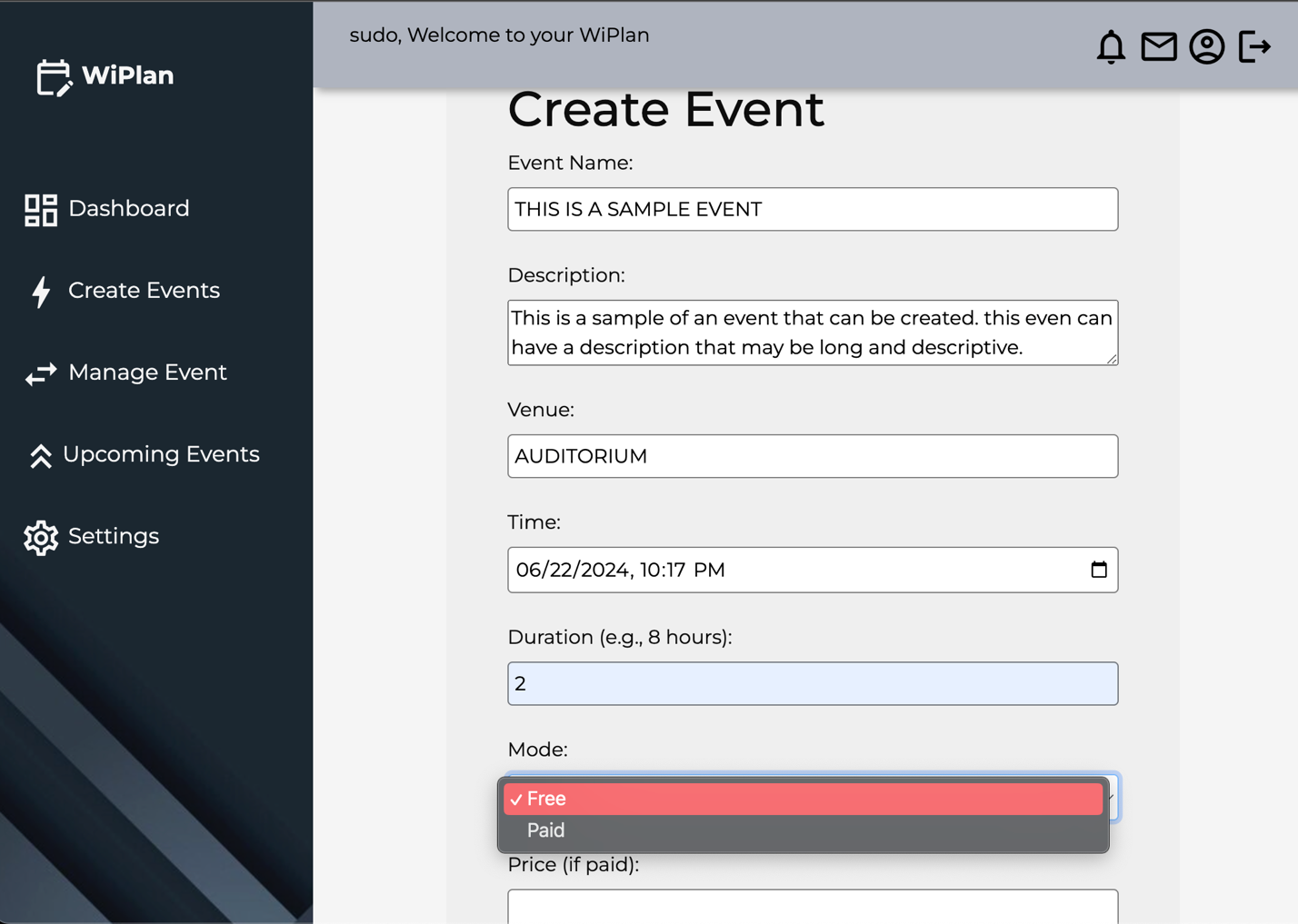
Within our system we have managed to register users through the use of emails and passwords which are then saved safely within the database using hashed functions, to help them gain access into the system of which they have two options of users mainly as a normal user or organizer who plan events from the main dashboard.



## Create event

This is a service accessible only to organizing users as they have the privilege to organize an event and post it for others to see it by giving detail information about the given event such as:

1. Name of event
2. Venue
3. Time
4. Duration
5. Description and
6. Image of associated event

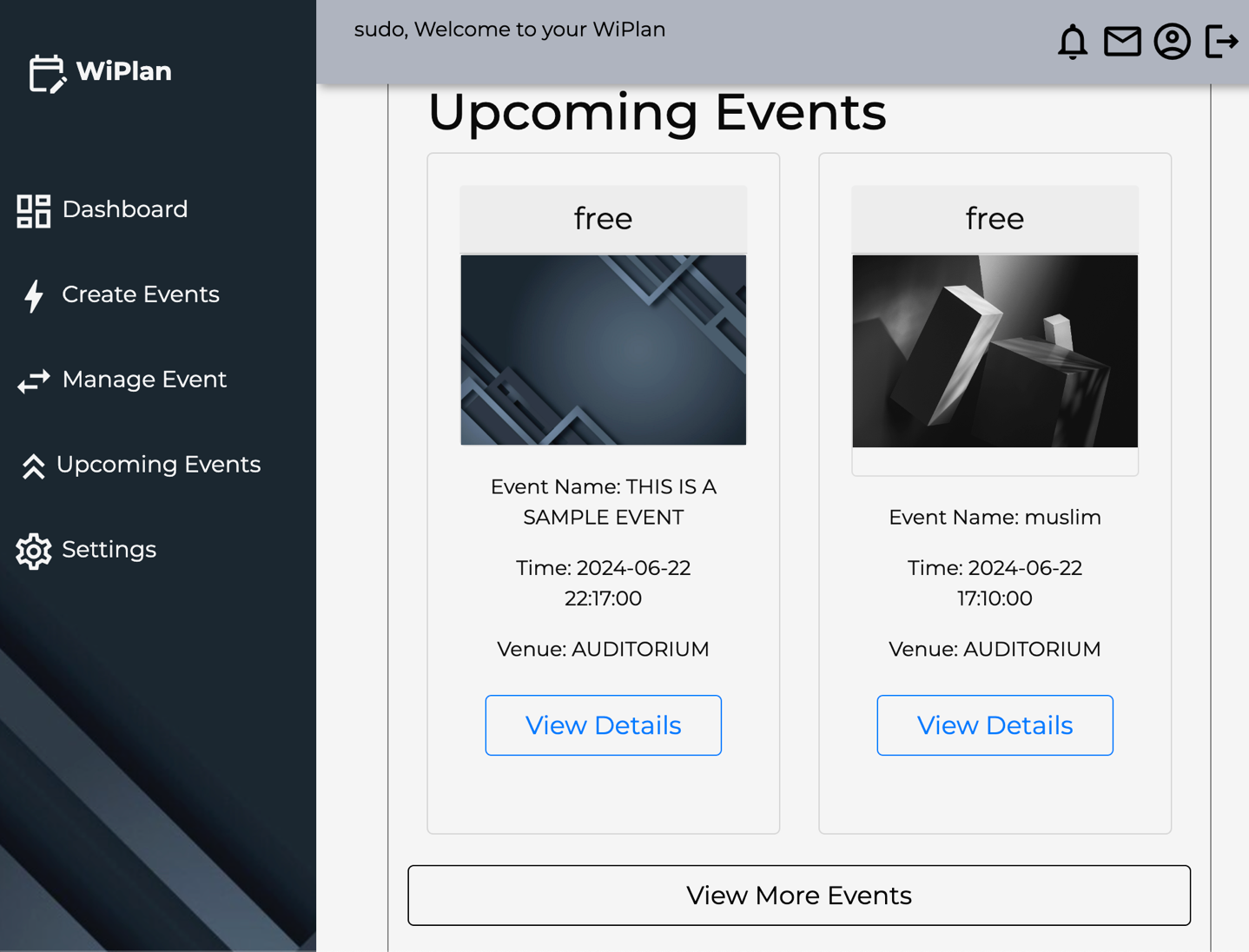


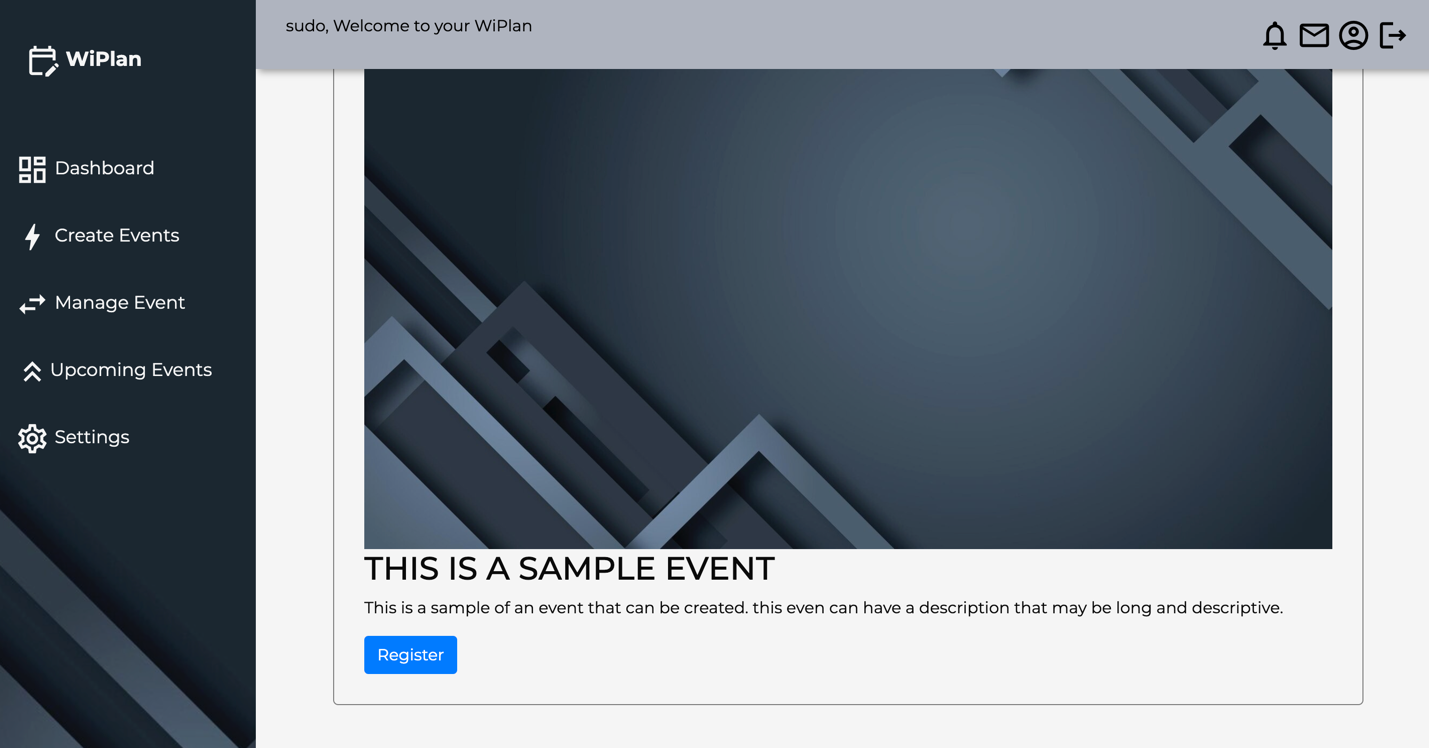
## View Upcoming Events and Register to event

This is a requirement given to both users as each can choose to attend a certain event thus, from the events menu a user can opt to attend an event by signing up for it and making the registration.

The event may be free or paid. For the free event the user will not be required to enter amount and amount may be entered if the event is a paid event.

## View Event



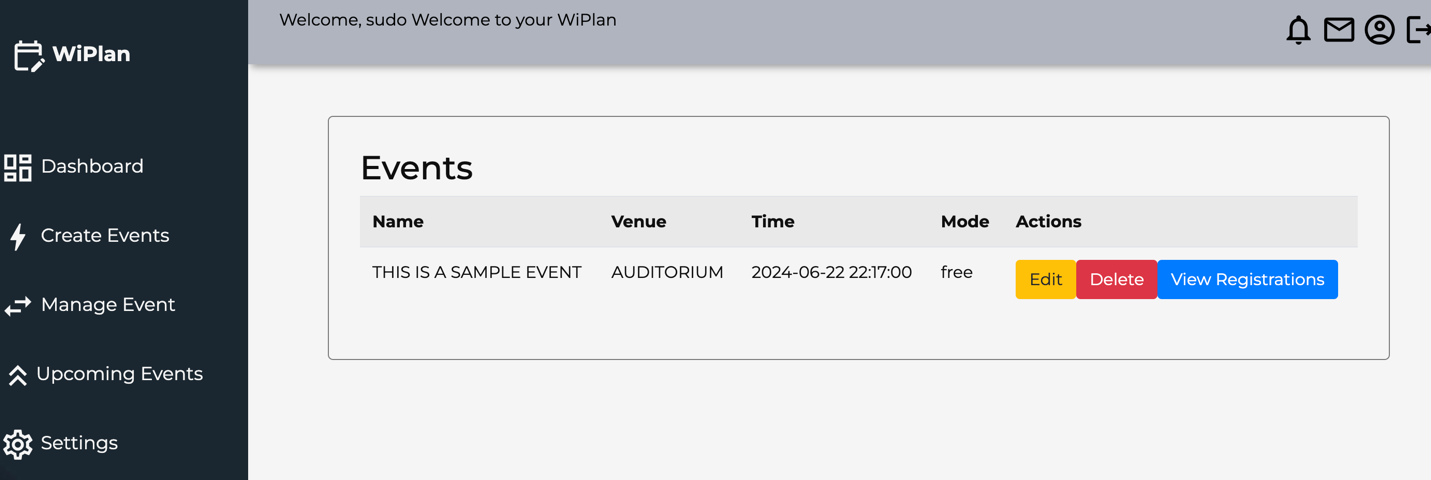
****

## Managing events

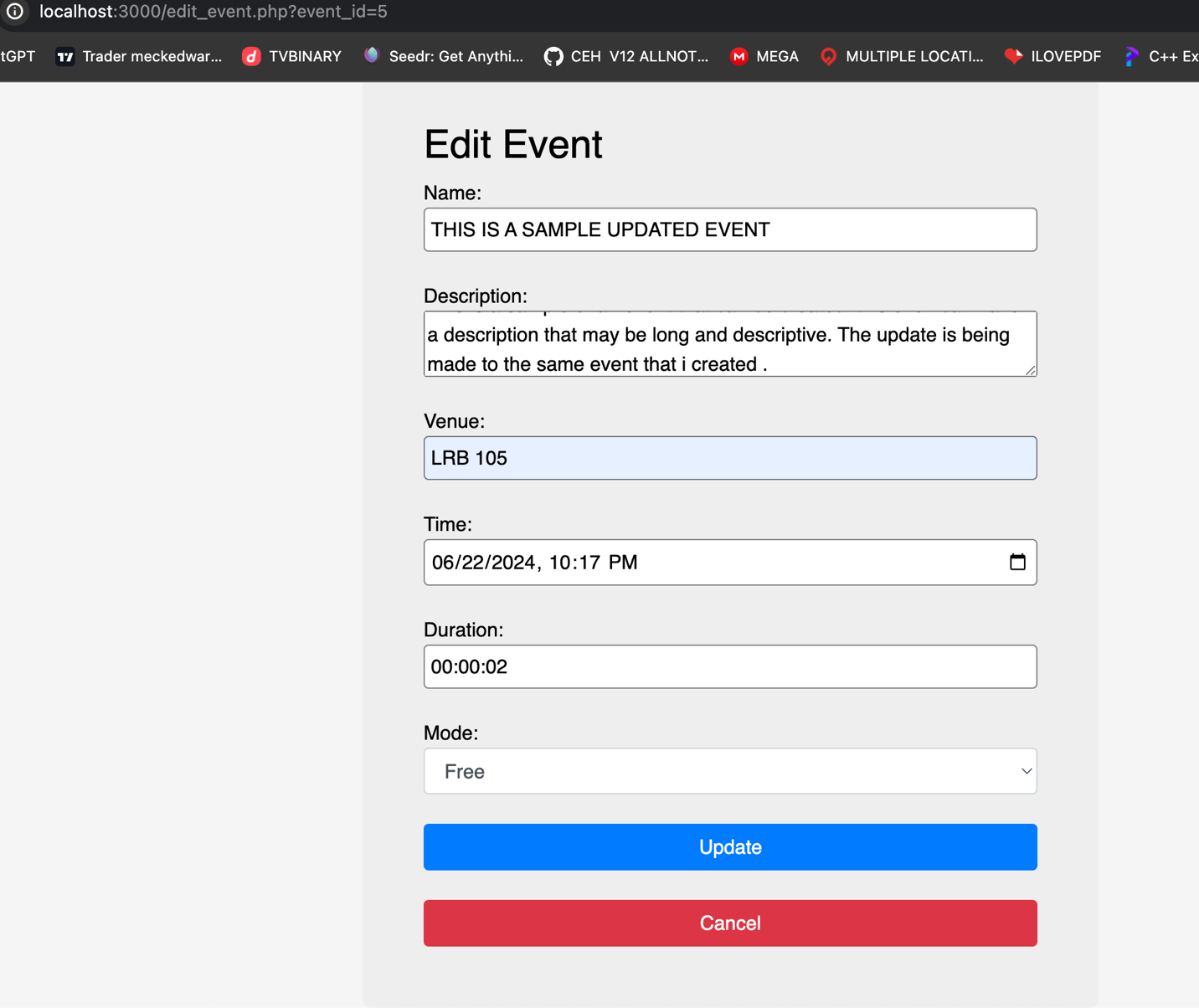
This is a requirement given to the organizing user as the user has the ability to manage everything concerning an event such as deleting an old event or cancelled event, creating agendas for the events, and editing any information regarding a specific event. But also, the organizer has the ability to view all registered people to a specific event of his/her creation.

The user can edit, delete and view users who have registered to that event

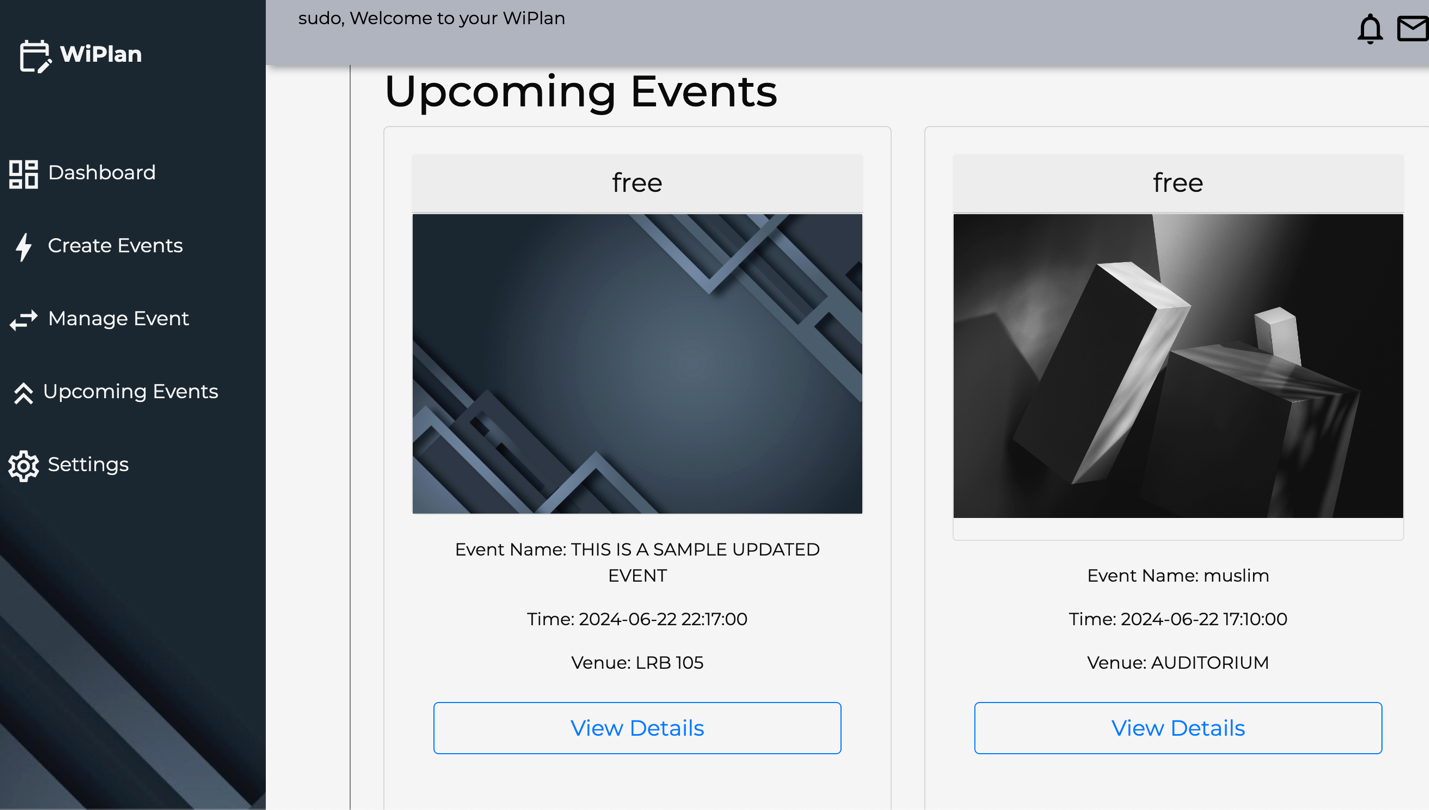
## View all the created events

****

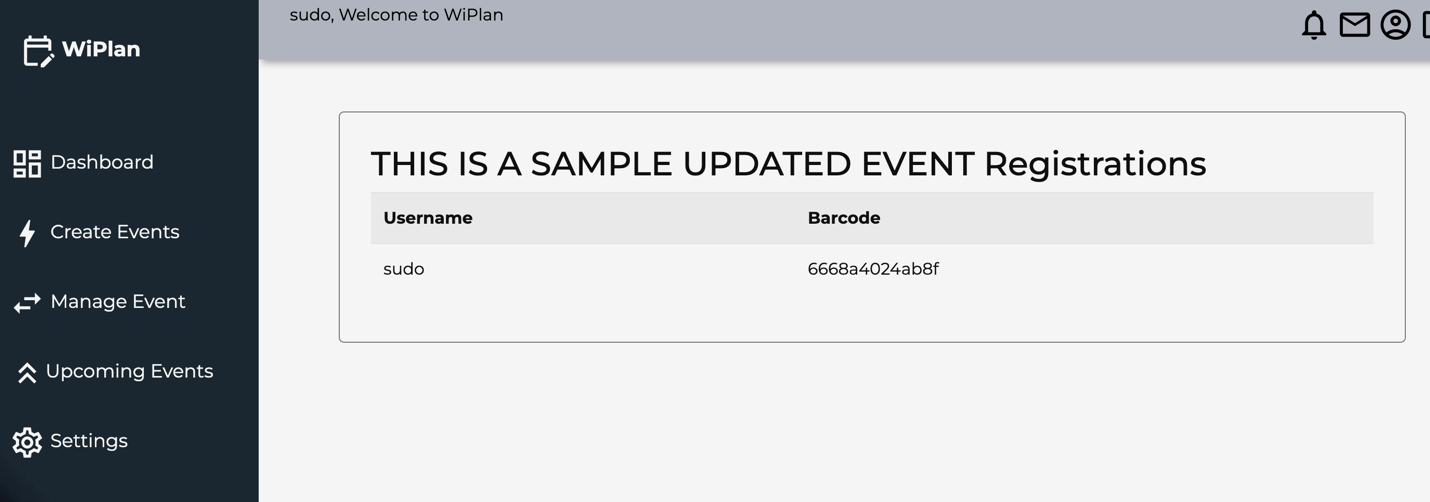
## Edit Created Event

****

## View Edited event

****

## View Registered users

****

# **Conclusion**

The Event Management System (WiPlan) successfully fulfills the core requirements of organizing and managing events efficiently. The system features, including user registration, event creation, agenda management, and feedback collection, provide a robust platform for both organizers and participants.

Throughout the development process, rigorous testing ensured the reliability and performance of the system. User documentation was created to facilitate ease of use, and technical documentation supports future maintenance and enhancements.

Security considerations were paramount, with measures implemented to protect user data and ensure secure communications. Looking forward, potential enhancements such as mobile application integration and advanced analytics can further extend the system's capabilities.

This project not only achieved its primary objectives but also laid a strong foundation for future development, contributing to streamlined event management and enhanced user experiences.