DBMS WebApp Report

Team Heap Undercut

Atul Jayesh 21CS10012
Barun Parua 21CS10014
Owais Ahmad Lone 21CS10048
Ranjim Prabal Das 21CS10054
Navaneeth Shaji 21CS30032

March 2024

Contents

1 Introduction

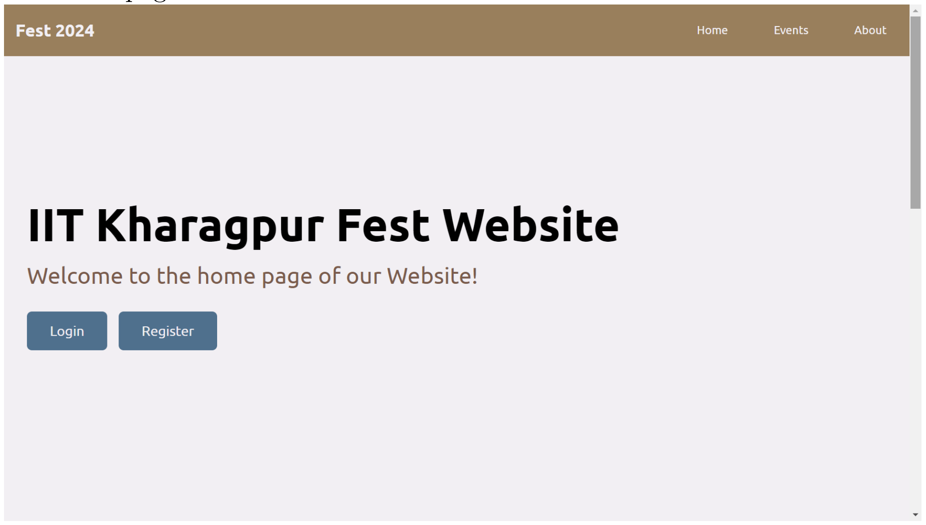
This is a report describing the Web Application that we have created for accessing a University Cultural and Technical Fest, organised by students at IIT Kharagpur.

The database associated with the application is hosted on a PostgreSQL server and we are using the Python programming language as an intermediary to fetch/update/add data to the database. In order to provide a friendly and appealing interface for the users of the application, we have used the Flask library and designed some web pages using HTML/CSS to provide a Web interface for the same.

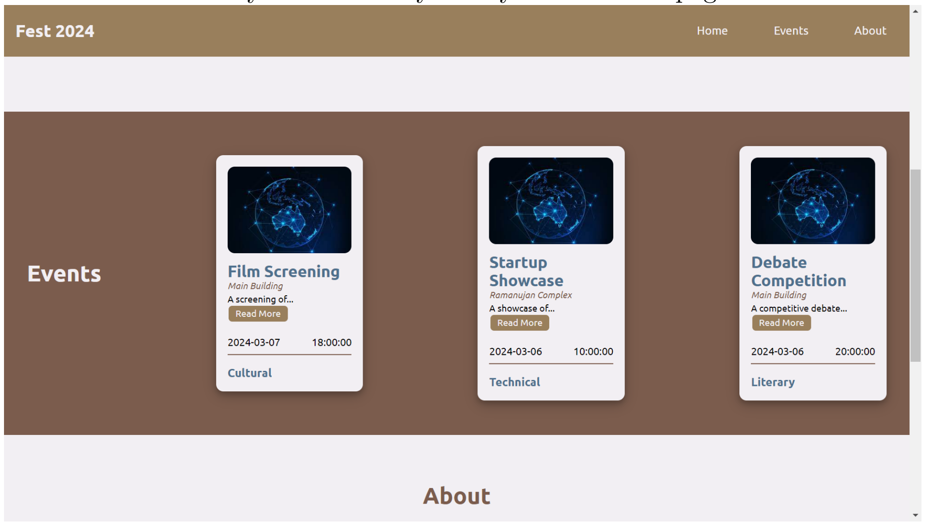
First of all, the home page contains some general information regarding the fest and events such as events names, descriptions, timings, venues etc and also provides the contact details of the admins whom the participants can contact if there are any issues during the fest. To access the remaining part of the database, the users have to login with their credentials provided during registration.

The next page contains some pictures of the home page.

The main page.


Fest 2024
Home
Events
About
IIT Kharagpur Fest Website
Welcome to the home page of our Website!
Login
Register

3 events are randomly selected every time you reload the page.



2 Schemas and the ER diagram

2.1 Schemas

Here we mention the structure of the various tables in our database along with the associated attribute names and types.

- **student** - stores the student details
  roll VARCHAR(9) NOT NULL,
  fest_id NUMERIC(5) PRIMARY KEY,
  name VARCHAR(50) NOT NULL,
  dept VARCHAR(50) NOT NULL,
  pass VARCHAR(50) NOT NULL

- **event** - stores the event details
  event_id NUMERIC(5) NOT NULL PRIMARY KEY,
  event_name VARCHAR(50) NOT NULL,
  event_date DATE NOT NULL,
  event_time TIME NOT NULL,
  event_venue VARCHAR(50) NOT NULL,
  event_type VARCHAR(50) NOT NULL,
  event_description VARCHAR(200),
  event_winner NUMERIC(5)

- **accommodation** - stores accommodation details
  acc_id NUMERIC(5) NOT NULL PRIMARY KEY,
  name VARCHAR(50) NOT NULL,
  capacity INT NOT NULL

- **ext_participant** - stores details of external participants
  fest_id NUMERIC(5) NOT NULL PRIMARY KEY,
  name VARCHAR(50) NOT NULL,
  college VARCHAR(50) NOT NULL,
  acc_id NUMERIC(5) NOT NULL ,
  FOREIGN KEY (acc_id) REFERENCES accomodation(acc_id)
  ON delete CASCADE ON update CASCADE,
  pass VARCHAR(50) NOT NULL

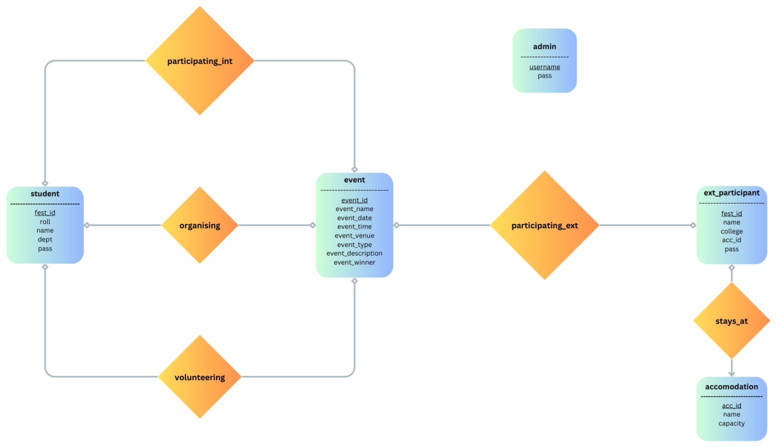
- **admin** - admin login details
  username VARCHAR(50) NOT NULL PRIMARY KEY,
  pass VARCHAR(50) NOT NULL

- **organising** - relates students to events with 'organise' role
  fest_id NUMERIC(5) NOT NULL,
  event_id NUMERIC(5) NOT NULL ,
  FOREIGN KEY (fest_id) REFERENCES student(fest_id)
  ON delete CASCADE ON update CASCADE,
  FOREIGN KEY (event_id) REFERENCES event(event_id)

ON delete CASCADE ON update CASCADE,
PRIMARY KEY(fest_id,event_id)

- **volunteering** - relates students to events with 'volunteer' role
fest_id NUMERIC(5) NOT NULL,
event_id NUMERIC(5) NOT NULL ,
FOREIGN KEY (fest_id) REFERENCES student(fest_id)
ON delete CASCADE ON update CASCADE,
FOREIGN KEY (event_id) REFERENCES event(event_id)
ON delete CASCADE ON update CASCADE,
PRIMARY KEY(fest_id,event_id)
- **participating_int** - relates students to events with 'participate' role
fest_id NUMERIC(5) NOT NULL,
event_id NUMERIC(5) NOT NULL ,
FOREIGN KEY (fest_id) REFERENCES student(fest_id)
ON delete CASCADE ON update CASCADE,
FOREIGN KEY (event_id) REFERENCES event(event_id)
ON delete CASCADE ON update CASCADE,
PRIMARY KEY(fest_id,event_id)
- **participating_ext** - relates external participants to events with 'participate' role
fest_id NUMERIC(5) NOT NULL,
event_id NUMERIC(5) NOT NULL ,
FOREIGN KEY (fest_id) REFERENCES ext_participant(fest_id)
ON delete CASCADE ON update CASCADE,
FOREIGN KEY (event_id) REFERENCES event(event_id)
ON delete CASCADE ON update CASCADE,
PRIMARY KEY(fest_id,event_id)

2.2 ER Diagram



Here is the ER diagram associated with our database and schema design.

3 Features

There are a list of features that are available to an user upon login. Here is a list of features as per the user type (note that the exact features available depends on user type).

3.1 External Participant

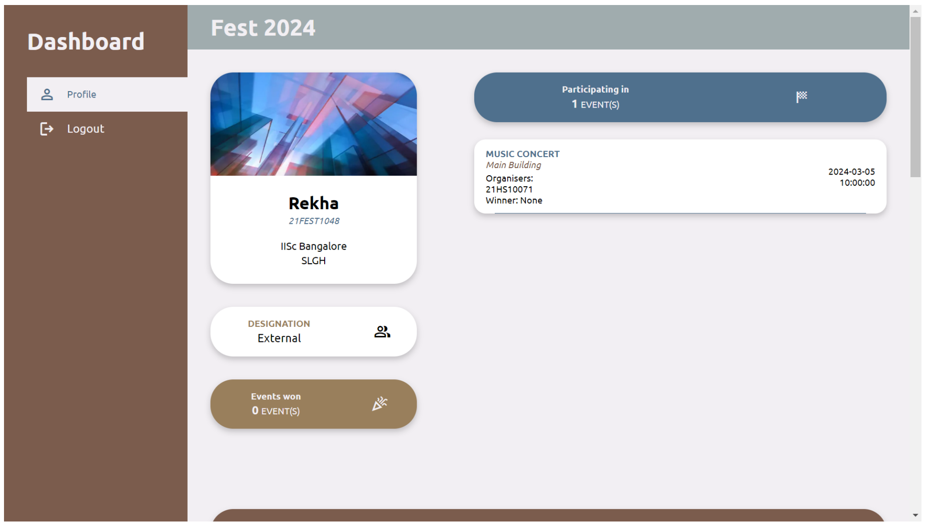
3.1.1 Registration

External participants have to create their account before accessing the fest database as their details are not available to us yet. So they have to go to Register page and enter their details such as Name, Password, College Name, Contact Number and Email Address in the respective text boxes and press Submit. If the registration is successful, then they get their Fest ID on the same page and get an option to go the Login Page. Otherwise, if the registration fails for any reason (either the accommodations are all full or the re-entered password did not match) then the corresponding error message is shown. The user may choose to go back to the Home Page at any point of time by choosing Go Back option.

3.1.2 Login

External participants can now login after their account has been created by specifying their Fest ID and password. Upon login, they can access the list of events they are currently participating in along with the necessary event details. Along with this, they can also access the list of events they are currently not participating in and get a choice to participate in them by clicking the appropriate button in the drop-down list. Some other necessary details such as Fest ID, name, college and accommodation are also mentioned in their dashboard. They can also see the the list of events they won!

The dashboard with necessary information.


Dashboard

Profile

Logout

Fest 2024

Rekha

21FEST1048

IISc Bangalore
SLGH

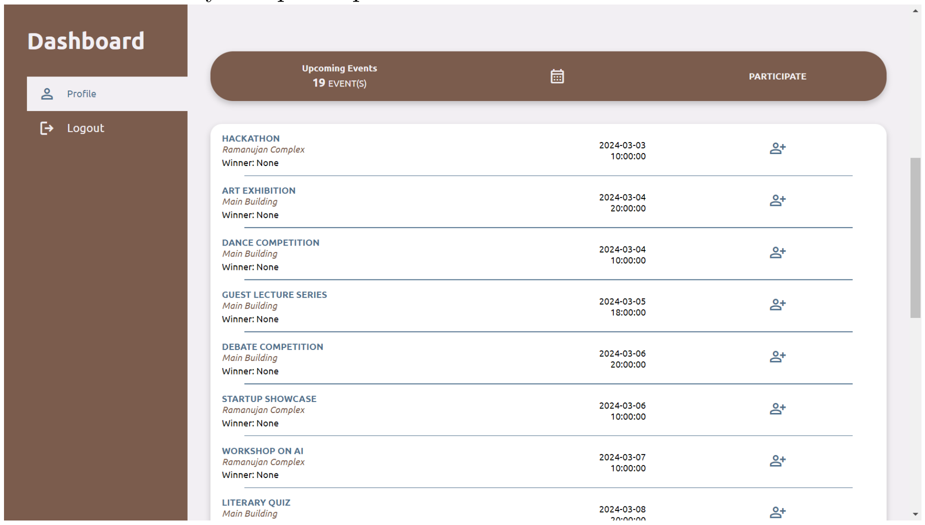
DESIGNATION
External

Events won
0 EVENT(S)

Participating in
1 EVENT(S)

MUSIC CONCERT
Main Building
Organisers:
21HS10071
Winner: None
2024-03-05 10:00:00

List of events they can participate in.


Dashboard

Profile

Logout

Upcoming Events
19 EVENT(S)

PARTICIPATE

HACKATHON *Ramanujan Complex* Winner: None	2024-03-03 10:00:00	
ART EXHIBITION *Main Building* Winner: None	2024-03-04 20:00:00	
DANCE COMPETITION *Main Building* Winner: None	2024-03-04 10:00:00	
GUEST LECTURE SERIES *Main Building* Winner: None	2024-03-05 18:00:00	
DEBATE COMPETITION *Main Building* Winner: None	2024-03-06 20:00:00	
STARTUP SHOWCASE *Ramanujan Complex* Winner: None	2024-03-06 10:00:00	
WORKSHOP ON AI *Ramanujan Complex* Winner: None	2024-03-07 10:00:00	
LITERARY QUIZ *Main Building*	2024-03-08	

3.2 Student

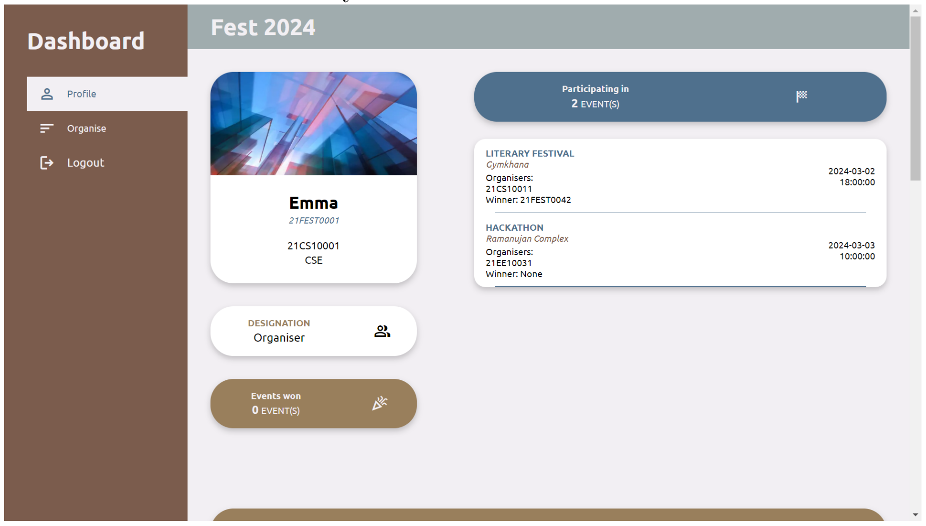
3.2.1 Registration

The students do not have to register for the fest according to our design pattern and their Login Details are already available to them (basically initial password is 'bt21' for students with roll numbers of the form 21XX10YYY and 'dd21' for students with roll numbers of the form 21XX30YYY where XX denotes department code, while YYY is the associated serial number. They do have an option to change their password later upon Login.

3.2.2 Login

Upon login, the content a student is able to see depends on the type of role they have. A student can have three roles namely, participant, volunteer and organiser. So firstly a student can see the lists of events they are participating in (along with the necessary details). Similarly the list of events they are volunteering for is also shown. Note that either or both of the list might be empty in the beginning. A list of all other events is also shown. A student can choose to either participate or volunteer for the other events as per their choice. They need to choose the appropriate button in the drop-down list. They can also see the the list of events they won!

The dashboard with necessary information.



Emma
21FEST0001
21CS10001
CSE

DESIGNATION
Organiser

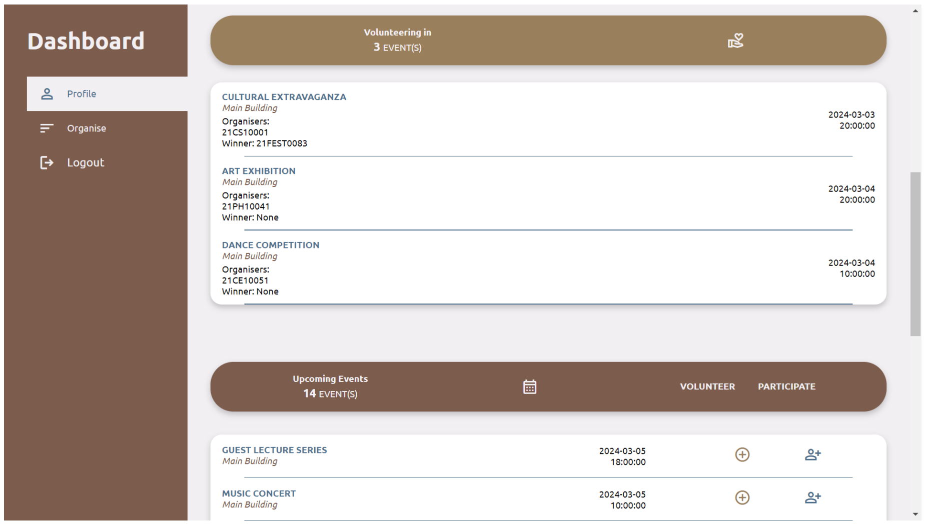
Events won
0 EVENT(S)

Participating in
2 EVENT(S)

LITERARY FESTIVAL
Gymkhana
Organisers:
21CS10011
Winner: 21FEST0042
2024-03-02
18:00:00

HACKATHON
Ramanujan Complex
Organisers:
21EE10031
Winner: None
2024-03-03
10:00:00

List of events they are volunteering in. Can choose to do more!



CULTURAL EXTRAVAGANZA
Main Building
Organisers:
21CS10001
Winner: 21FEST0083
2024-03-03
20:00:00

ART EXHIBITION
Main Building
Organisers:
21PH10041
Winner: None
2024-03-04
20:00:00

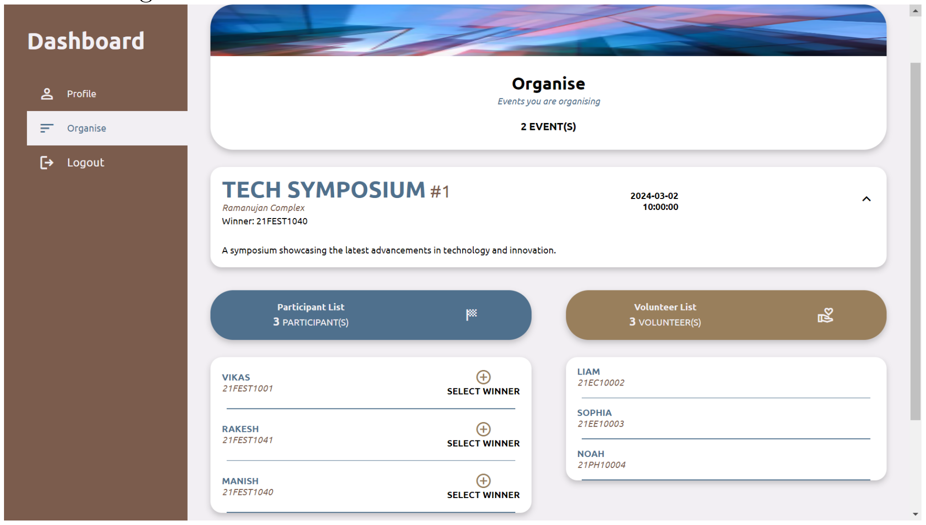
DANCE COMPETITION
Main Building
Organisers:
21CE10051
Winner: None
2024-03-04
10:00:00

GUEST LECTURE SERIES
Main Building
2024-03-05
18:00:00

MUSIC CONCERT
Main Building
2024-03-05
10:00:00

Apart from these two roles a student also might have the organising role. This means they are in-charge of the event and can see the complete list of participants and volunteers associated with that event and have the option to choose a winner from the participant list. For ease of access, we have provided the necessary details on a separate tab which is visible only to organisers.

What an organiser can see.



TECH SYMPOSIUM #1
Ramanujan Complex
Winner: 21FEST1040
2024-03-02
10:00:00

A symposium showcasing the latest advancements in technology and innovation.

VIKAS
21FEST1001
SELECT WINNER

RAKESH
21FEST1041
SELECT WINNER

MANISH
21FEST1040
SELECT WINNER

LIAM
21EC10002

SOPHIA
21EE10003

NOAH
21PH10004

Note that while a student may have all 3 roles, they will have atmost one role for any particular event. This means that all the three lists associated with a student are mutually exclusive but might not be exhaustive, i.e. there maybe events that a student is not at all involved with. This is to ensure that students are not able to affect their own results. Also, a student cannot choose to get the organise role on their own and they must contact the Administrators who will then assign the roles accordingly. More on this part will be covered in the admin section.

3.3 Administrators

3.3.1 Registration

There is no option to register as an admin as there are already allotted in our database. We have provided some username password combinations such as 'admin'/'admin' and 'root'/'root' for this purpose.

3.3.2 Login

On login, an admin can see the list of events along with necessary details. Now as previously mentioned, an admin may choose to either add or remove organisers to any of events. The organisers must be students that are not yet volunteering or participating in that specific event. Our system takes care of this and throws an error message if it is violated. Note that this is done by entering the roll number of the student on a separate form.

Apart from this, admins can choose to add more events as well by filling up a form with all details such as date, time, description, venue etc. They may then add organisers as per their choice as described above.

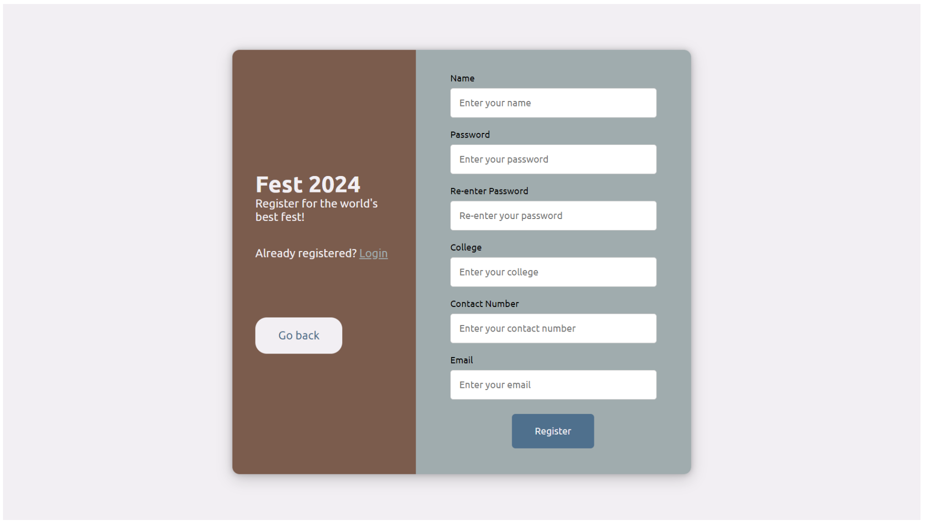
An option to remove participants is also provided to the admins. Say if an external participant wants to drop from the fest for some reason then they can be dropped by the admin using this functionality. Note that a cascade operation ensures the drop happens across the event tables as well.

4 Forms

As we need to keep updating the database by taking the suitable inputs from the users, we need forms. Here is a list of all forms that we have used to get data from the users.

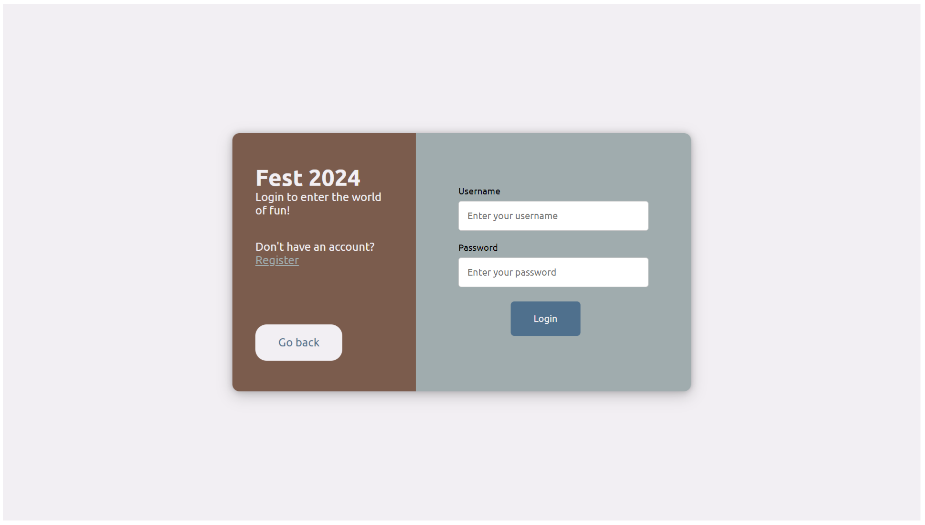
4.1 Registration Form

This form is provided for the external participants to enter their details such as Name, Password, College Name, Contact Number and Email Address in the respective text boxes and press Submit. If the registration is valid then they receive their Login ID else an error message is shown. Note that the fields must be supplied with appropriate values like email address should be of the form X@Y, contact number must be a number etc otherwise registration will fail.



4.2 Login Form

This form is provided to the user so that they can enter their username and password in the respective text boxes provided. On clicking the Login button, it is then processed to check whether it is valid or not. If it is valid, the user is directed to their dashboard as per their roles else the user has to re enter the correct username and password.



4.3 Event Form

This form is provided to the admin so that they can add more events to the fest. All the necessary event details such as venue, description, time etc must be specified in the correct format. After that when the admin selects enter, it is added to the database.

4.4 Add/Remove Organisers

This form is also provided to the admin so that they can add/remove organisers from events. On clicking the event, they get a form to enter the Roll Number of the student they want to add/remove to the events. If there is an error then the respective error message is printed else the corresponding message is printed. Note that initially there is an organiser for each event but there is no rule implemented by us that an event must have at least one organiser at any point of time. Also, a student may organise multiple events and an event may have any number of organisers. Only restriction is, a student must not be a participant or a volunteer in the event he is being added to.

4.5 On click Forms

These were the forms in which we had to enter data. There are numerous other forms in the application where users can give data to the database by simply clicking on the respective buttons. Some examples are the participate button for external participants, participate/volunteer button for students, select winner button for organisers etc.

5 Queries

As we are operating using data from a PostgreSQL server, we need to make the appropriate queries in order to fetch data, make suitable decisions based on them and finally update/delete or add new data.

Hence here is a list of all the queries used in our application with which task they are helping with.

1. Selecting 3 events randomly from the event list to display on the home page.

```
select event_id,event_name,event_venue,
    event_description,event_date,event_time,
    event_type from event order by random()
    limit 3;
```

2. Checking if the username-password combination matches for an admin.

```
select * from admin where username = '{
    username}' and pass = '{password}';
```

3. Checking if the username-password combination matches for an external participant.

```
select fest_id from ext_participant where
    fest_id = '{username}' and pass = '{
    password}';
```

4. Checking if the username-password combination matches for a student.

```
select fest_id from student where fest_id = '
    {username}' and pass = '{password}';
```

5. Checking if the student is an organiser for some event.

```
select fest_id from organising where fest_id
    = '{username}';
```

6. Getting complete event details for an admin to access.

```
select event_id,event_name, event_venue,
    event_date, event_time from event;
```

7. Getting roll numbers of organisers for an event.

```
select roll from organising natural join
    student where event_id = {event_id};
```

8. Getting last event ID from the event table so that we can add more events.

```
select event_id from event order by event_id
    desc limit 1;
```

9. Insert a new event in the list.

```
insert into event (event_id,event_name,
    event_venue, event_date, event_time,
    event_type, event_description) VALUES ('{
    event_id}','{event_name}','{event_venue}'
    ,'{event_date}','{event_time}','{
    event_type}','{event_description}');
```

10. Get list of participants to subsequently remove an external participant.

```
select fest_id, name from ext_participant;
```

11. Select the accommodation ID of the external participant, add one to it (as we are removing a participant), then remove it from the list.

```
select acc_id from ext_participant where
    fest_id = {fest_id};
update accomodation SET capacity = capacity +
     1 where acc_id = {acc_id[0]};
delete from ext_participant where fest_id = {
    fest_id};
```

12. Getting fest ID of the student we have to add using the roll number, getting the event ID and then inserting it to organising table

```
select fest_id from student where roll = '{
    roll}';
select * from participating_int where fest_id
     = {fest_id[0]} and event_id = {event_id
    };
insert into organising VALUES ({fest_id[0]},{
    event_id});
```

13. Getting fest ID of the student we have to add using the roll number and then removing the student from the organising table.

```
select fest_id from student where roll = '{
    roll}';
delete from organising where fest_id = {
    fest_id[0]} and event_id = {event_id};
```

14. Getting the details of the user (external participant) to show on their dashboard.

```
select fest_id , ext_participant.name, college
    , accomodation.name from ext_participant ,
     accomodation where fest_id = {fest_id}
    and accomodation.acc_id = ext_participant
    .acc_id;
```

15. Getting list of events they are participating in and the list of events they are not participating in.

```
select event_id ,event_name ,event_date ,
    event_time ,event_venue ,event_winner from
    event natural join participating_ext
    where fest_id = {fest_id};
select roll , name from organising natural
    join student where event_id = {event_id};
     -- to get organisers
select event_id ,event_name ,event_date ,
    event_time ,event_venue from event where
    event_id not in (select event_id from
    participating_ext where fest_id = {
    fest_id});
```

16. Getting list of events won by external participants.

```
select event_id ,event_name ,event_date ,
    event_time ,event_venue from event where
    event_id in (select event_id from
    participating_ext where fest_id = {
    fest_id} and event_winner = {fest_id});
```

17. Getting details of student users.

```
select fest_id , name, roll , dept from student
     where fest_id = {fest_id};
```

18. Getting list of events in which student is participating.

```
select event_id ,event_name ,event_date ,
    event_time ,event_venue ,event_winner from
    event natural join participating_int
    where fest_id = {fest_id};
select roll , name from organising natural
    join student where event_id = {event_id};
-- to get organiser list
```

19. Getting list of events in which student is volunteering.

```
select event_id,event_name,event_date,
   event_time,event_venue,event_winner from
   event natural join volunteering where
   fest_id = {fest_id};
select roll, name from organising natural
   join student where event_id = {event_id};
-- to get organiser list
```

20. Getting list of events in which student is organising.

```
select * from event where event_id in (select
    event_id from organising where fest_id =
    {fest_id});
```

21. Getting participant and volunteer lists in which a student is organising.

```
select fest_id, name from participating_ext
   natural join ext_participant where
   event_id = {event_id} union select
   fest_id, name from participating_int
   natural join student where event_id = {
   event_id};

select roll, name from volunteering natural
   join student where event_id = {event_id};
```

22. Getting list of events in which student is not involved yet to give option to participate or volunteer.

```
select event_id,event_name,event_date,
   event_time,event_venue from event where
   event_id not in (select event_id from
   participating_int where fest_id = {
   fest_id}) and event_id not in (select
   event_id from volunteering where fest_id
   = {fest_id}) and event_id not in (select
   event_id from organising where fest_id =
   {fest_id});
```

23. Getting list of events won by student.

```
select event_id,event_name,event_date,
   event_time,event_venue from event where
   event_id in (select event_id from
   participating_int where fest_id = {
   fest_id} and event_winner = {fest_id});
```

24. Setting of a winner by the organiser.

```
update event SET event_winner = '{winner_name
    }' where event_id = {event_id};
```

25. Adding either external participants or students to the participating list.

```
insert INTO participating_ext VALUES ({
    fest_id},{event_id});
insert INTO participating_int VALUES ({
    fest_id},{event_id});
```

26. Adding interested students to the volunteering list.

```
insert INTO volunteering VALUES ({fest_id},{
    event_id});
```

27. Registration of an external participant. Getting the largest fest ID from the table yet, to create next fest id and then insert accordingly to table all the values entered by the user.

```
select fest_id from ext_participant order by
    fest_id desc limit 1;
select acc_id from accomodation order by
    capacity desc limit 1;
insert into ext_participant VALUES ({fest_id
    },'{username}','{college}',{acc_id},'{
    password}');
update accomodation SET capacity = capacity -
     1 where acc_id = {acc_id};
```

Note that the list of queries used to create the tables and to feed the initial data to the database is not mentioned here as it is already done on our server and the application is not exactly interacting with them.

6 Triggers

We have not implemented any separate triggers as we are already doing the necessary validations using a combination of SQL and Python checks at all required steps. We get the data stored using SQL, and then use the Python login to decide upon the next steps.

7 Libraries Used

We have aimed to keep this implementation simple and hence we have used very few libraries and done a lot of the work from scratch rather than using existing templates. Here is a list of the libraries used with their basic functions.

- **psycopg2** This is the primary library that allows us to connect to the PostgreSQL database and then execute all the required SQL queries.
- **Flask** Flask is the micro web framework that we have used to implement the web interface of this application. The Jinja template allows us to use Python flow control inside HTML pages to give a better structure and logic to the pages. It is minimalistic as in we had to create all the necessary templates/authentication/stylings etc from scratch.
- **hashlib** This library implements a common interface to many different secure hash algorithms. We have used one of them (specifically the *sha256* algorithm) to ensure proper authentication during login.

These libraries can be installed easily using the Python package manager (PIP).

8 Conclusion

In summary, we've created a robust web application for managing a University Cultural and Technical Fest at IIT Kharagpur. Utilizing Flask for web development and PostgreSQL for database management, our application offers features tailored to different user roles: external participants, students, and administrators.

External participants can register, view event details, participate, and get to know which events they won. Students enjoy similar functionalities based on their roles as participants, volunteers, or organizers. Administrators have privileged access to manage events, add/remove organizers, add new events, and handle participant registrations.

With a carefully designed database schema, ER diagram, forms, queries, and validation mechanisms, our application ensures smooth functionality and data integrity. It offers an intuitive and user-friendly interface, enhancing the fest experience for all users.

Overall, our DBMS WebApp provides an efficient solution for organizing and managing university events, contributing to a seamless fest experience.