

**TECHEVENT**

Alisha Singh (1847206)

Alwin Joseph (1847207)

Sachin Thakur (1847250)

Under the guidance of Prof. Smitha Vinod

DBMS project report submitted in partial fulfillment of the requirements for the award of degree of

Master of Computer Applications of CHRIST (Deemed to be University)

October - 2019



CERTIFICATE

*This is to certify that the report titled* ***TechEvent*** *is a bona fide record of work done by* ***Alisha Singh (1847206), Alwin Joseph (1847207) and Sachin Thakur (1847250)*** *of CHRIST (Deemed to be University), Bangalore, in partial fulfillment of the requirements of III Semester MCA during the year 2019.*

**Head of the Department Project Guide**

Valued-by:

|  |  |  |
| --- | --- | --- |
|  | Examination Centre | : CHRIST  (Deemed to be University) |
| 1. |  |
| 2. | Date of Exam | : 30/08/2019 |

# ACKNOWLEDGEMENTS

### First of all, we thank the lord almighty for his immense grace and blessings showered on me at every stage of this work. we are greatly indebted to our Head of the Department of Computer Science, Prof. Joy Paulose, CHRIST (Deemed to be University) for providing the opportunity to take up this project as part of my curriculum.

We are greatly indebted to our MCA Course Coordinator, Prof. Tulsi, for providing the opportunity to take up this project and for helping with her valuable suggestions.

we are deeply indebted to our project guide Prof. Smitha Vinod, for her assistance and valuable suggestions as a guide. She made this project a reality.

we express our sincere thanks to Dr. Chandra J, Prof. Arul Kumar and Prof. Nismon Raio, lecturers of the Department of Computer Science, CHRIST (Deemed to be University), for their valuable suggestions during the course of this project. Their critical suggestions helped us to improve the project work.

Acknowledging the efforts of everyone, their chivalrous help in the course of the project preparation and their willingness to corroborate with the work, their magnanimity through lucid technical details lead to the successful completion of my project.

we would like to express our sincere thanks to all our friends, colleagues, parents and all those who have directly or indirectly assisted during this work.

**ABSTRACT**

The project TECHEVENT is a web platform that is aimed to help the organizers of the fests, workshops, hackathon and other events to post their event details, resources and related information. The platform provides registration to the events, this will help the participants to register and participate in the events easily. This helps the organizers to get a count of the number of participants for the event. The companies who would like to sponsor the events can sponsor the events based on the event details. This platform maintains the blog feature and community.

The web platform helps the organizers to register an event easily, provide its description, and other details. The interested people who are searching for the events, can easily and quickly register for the event through our platform, after completion of the event they will be prompted for the feedback/review the event they have attended. The organizers can easily get sponsors through this portal, they can also specify the facilities for the sponsors and can determine the different scale of sponsor types. They can share the required resources and contents through this portal.

This web platform is developed in Django framework, which is a python platform for web development.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **Sl. No.** | **Content** | **Page No.** |
| **1** | **Introduction** | **1** |
|  | 1.1 Overview of the system | 1 |
| **2** | **System analysis** | **2** |
|  | 2.1 Existing system | 2 |
|  | 2.1.1 Limitations of the existing system | 2 |
|  | 2.2 Proposed system | 2 |
|  | 2.2.1 Advantages of proposed system | 3 |
|  | 2.3 Literature survey | 3 |
|  | 2.4 Software tools used | 4 |
|  | 2.4.1 Front end | 4 |
|  | 2.4.1.1 Html | 4 |
|  | 2.4.1.2 Bootstrap | 5 |
|  | 2.4.1.3 JavaScript | 5 |
|  | 2.4.2 Backend | 6 |
|  | 2.4.2.1 Django 2.1 | 6 |
|  | 2.4.2.2 PostgreSQL | 7 |
|  | 2.5 Software used | 7 |
|  | 2.5.1 GitHub | 7 |
|  | 2.5.2 Sublime text / Visual studio code | 7 |
|  | 2.5.3 Pgadmin4 (PostgreSQL IDE) | 8 |
| **3** | **System Requirements** | **9** |
|  | 3.1 System model | 9 |
|  | 3.2 User Requirements | 9 |
|  | 3.2.1 Functional requirements | 9 |
|  | 3.2.1.1 Participant | 9 |
|  | 3.2.1.2 Sponsors | 9 |
|  | 3.2.1.3 Organizers | 10 |
|  | 3.2.1.4 Blog | 10 |
|  | 3.2.1.5 Community | 10 |
|  | 3.2.1.6 Resources | 10 |

|  |  |  |
| --- | --- | --- |
|  | 3.2.1.7 Query | 10 |
|  | 3.2.1.8 Accounts | 10 |
|  | 3.2.1.9 About us | 10 |
|  | 3.2.2 Non-Functional requirements | 11 |
|  | 3.3 System requirements | 11 |
|  | 3.3.1 Hardware requirements | 11 |
|  | 3.3.2 Software requirements | 11 |
|  | 3.3.2.1 Front end | 11 |
|  | 3.3.2.2 Back end | 12 |
|  | 3.3.2.3 Software used | 12 |
| **4** | **Design specification** | **13** |
|  | 4.1 Architectural design | 13 |
|  | 4.2 Data flow diagram | 14 |
|  | 4.2.1 Context diagram (Level 0 DFD) | 14 |
|  | 4.2.2 Level 1 DFD | 14 |
|  | 4.2.3 Level 2 DFD | 15 |
|  | 4.3 Entity relationship diagram | 15 |
|  | 4.4 Database design | 17 |
|  | 4.4.1 User | 17 |
|  | 4.4.2 User profile | 17 |
|  | 4.4.3 Blogs info | 18 |
|  | 4.4.4 Event location | 18 |
|  | 4.4.5 Event details | 19 |
|  | 4.4.6 Organize event | 19 |
|  | 4.4.7 Share resource | 20 |
|  | 4.4.8 Sponsorship | 20 |
|  | 4.4.9 Event feedback | 21 |
|  | 4.4.10 Participate event | 21 |
|  | 4.4.11 Sponsor transaction | 22 |
|  | 4.4.12 About Description | 22 |
|  | 4.4.13 Contact | 22 |
|  | 4.4.14 Our services | 23 |
|  | 4.4.15 Team info | 23 |

|  |  |  |
| --- | --- | --- |
|  | 4.4.16 Article | 23 |
|  | 4.4.17 Admin log | 24 |
|  | 4.4.18 Our partners | 24 |
|  | 4.4.19 Slider | 24 |
| **5** | **Implementation details** | **25** |
|  | 5.1 Source code | 25 |
|  | 5.1.1 TechEvent | 25 |
|  | 5.1.1.1 Settings.py | 25 |
|  | 5.1.1.2 Urls.py | 28 |
|  | 5.1.2 Accounts | 28 |
|  | 5.1.2.1 Models.py | 28 |
|  | 5.1.2.2 Urls.py | 29 |
|  | 5.1.2.3 Views.py | 29 |
|  | 5.1.2.4 Login.html | 31 |
|  | 5.1.3 Organizer | 38 |
|  | 5.1.3.1 Models.py | 38 |
|  | 5.1.3.2 Urls.py | 39 |
|  | 5.1.3.3 Views.py | 40 |
|  | 5.1.4 Sponsor | 55 |
|  | 5.1.4.1 Models.py | 55 |
|  | 5.1.4.2 Urls.py | 55 |
|  | 5.1.4.3 Views.py | 56 |
|  | 5.1.5 Participants | 61 |
|  | 5.1.5.1 Models.py | 61 |
|  | 5.1.5.2 Urls.py | 61 |
|  | 5.1.5.3 Views.py | 62 |
|  | 5.2 Screenshots | 68 |
|  | 5.2.1 Login and registration | 68 |
|  | 5.2.2 Our team and services | 68 |
|  | 5.2.3 Sponsors for event | 69 |
|  | 5.2.4 Participants for event | 69 |
|  | 5.2.5 Event details for participants | 69 |
|  | 5.2.6 Sponsored events | 70 |

|  |  |  |
| --- | --- | --- |
|  | 5.2.7 Participated events | 70 |
|  | 5.2.8 Upcoming Events | 70 |
|  | 5.2.9 Enquire about event | 71 |
|  | 5.2.10 Sponsor event | 71 |
|  | 5.2.11 Profile | 72 |
|  | 5.2.12 Write blogs | 72 |
|  | 5.2.13 Event venue | 73 |
|  | 5.2.14 Registered events | 73 |
|  | 5.2.15 Sponsorship details | 74 |
|  | 5.2.16 Share resources | 74 |
|  | 5.2.17 Enter event details | 75 |
|  | 5.2.18 Register event | 75 |
|  | 5.2.19 Organizer dashboard | 76 |
|  | 5.2.20 Blog details | 76 |
|  | 5.2.21 Admin Side | 77 |
|  | 5.2.22 Log Entries | 77 |
| **6** | **Testing** | **78** |
|  | 6.1 Test plan | 61 |
|  | 6.2 Test cases | 79 |
|  | 6.2.1 Login form | 79 |
|  | 6.2.2 Registration form | 80 |
|  | 6.2.3 Organize event form | 81 |
|  | 6.2.4 Organizer dashboard | 82 |
|  | 6.2.5 Your events | 82 |
|  | 6.2.6 Participate events | 82 |
|  | 6.2.7 Enquire about an event | 83 |
|  | 6.2.8 Project home page | 84 |
|  | 6.2.9 Participant dashboard | 84 |
|  | 6.2.10 Event feedback | 84 |
|  | 6.2.11 Blogs | 85 |
|  | 6.2.12 Contact Page | 85 |
|  | 6.2.13 Admin | 86 |
| **7** | **Conclusion** | **87** |
|  | 7.1 Future enhancement | 87 |
|  | References | 88 |

|  |  |  |
| --- | --- | --- |
| **Table No.** | **Table Name** | **Page No.** |
|  | **Design specification** | **13** |
| Table 4.1 | user | 17 |
| Table 4.2 | userprofile | 17 |
| Table 4.3 | blogs\_info | 18 |
| Table 4.4 | eventlocation | 18 |
| Table 4.5 | event\_details | 19 |
| Table 4.6 | organise\_event | 19 |
| Table 4.7 | share\_resource | 20 |
| Table 4.8 | sponsorship | 20 |
| Table 4.9 | event\_feedback | 21 |
| Table 4.10 | participate\_event | 21 |
| Table 4.11 | sponsor\_transaction | 22 |
| Table 4.12 | about\_description | 22 |
| Table 4.13 | contact | 22 |
| Table 4.14 | our\_services | 23 |
| Table 4.15 | team\_info | 23 |
| Table 4.16 | article | 23 |
| Table 4.17 | admin\_log | 24 |
| Table 4.18 | our\_partner | 24 |
| Table 4.19 | slider | 24 |
|  | **Testing** | **78** |
| Table 6.1 | Login test cases | 79 |
| Table 6.2 | Registration form test cases | 80 |
| Table 6.3 | Organize event form test cases | 81 |
| Table 6.4 | Organizer Dashboard test cases | 82 |
| Table 6.5 | Your events test cases | 82 |
| Table 6.6 | Participate events test cases | 82 |
| Table 6.7 | Enquire about an event test cases | 83 |
| Table 6.8 | Project home page test cases | 83 |
| Table 6.9 | Participant Dashboard test cases | 84 |
| Table 6.10 | Event feedback test cases | 84 |
| Table 6.11 | Blogs test cases | 84 |
| Table 6.12 | Contact page test cases | 85 |
| Table 6.13 | Admin test cases | 86 |

|  |  |  |
| --- | --- | --- |
| **Fig. No.** | **Figure Name** | **Page No.** |
|  | **System Requirements** | **9** |
| Fig. 3.1 | System model | 9 |
|  | **Design specification** | **13** |
| Fig. 4.1 | Project architecture | 13 |
| Fig. 4.2 | Level 0 DFD | 14 |
| Fig. 4.3 | Level 1 DFD | 14 |
| Fig. 4.4 | Level 2 DFD | 15 |
| Fig. 4.5 | ER Diagram | 16 |
|  | **Implementation details** | **25** |
| Fig. 5.1 | Login and registration | 68 |
| Fig. 5.2 | Our team and services | 68 |
| Fig. 5.3 | Sponsors for event | 69 |
| Fig. 5.4 | Participants for event | 69 |
| Fig. 5.5 | Event details for participants | 69 |
| Fig. 5.6 | Sponsored events | 70 |
| Fig. 5.7 | Participated events | 70 |
| Fig. 5.8 | Upcoming Events | 70 |
| Fig. 5.9 | Enquire about event | 71 |
| Fig. 5.10 | Sponsor event | 71 |
| Fig. 5.11 | Profile | 72 |
| Fig. 5.12 | Write blogs | 72 |
| Fig. 5.13 | Event venue | 73 |
| Fig. 5.14 | Registered events | 73 |
| Fig. 5.15 | Sponsorship details | 74 |
| Fig. 5.16 | Share resources | 74 |
| Fig. 5.17 | Enter event details | 75 |
| Fig. 5.18 | Register event | 75 |
| Fig. 5.19 | Organizer dashboard | 76 |
| Fig. 5.20 | Blog details | 76 |
| Fig. 5.21 | Admin Side | 77 |
| Fig. 5.22 | Log Entries | 77 |

1. **INTRODUCTION**

**1.1 OVERVIEW OF THE SYSTEM**

TECHEVENT is a web platform that is aimed to help the organizers of the fests, workshops or hackathon events to post their event details, resources and related information. The platform provides registration to the events, this will help the participants to register and participate in the events easily. This helps the organizers to get the count on the number of participants for their events. The companies who would like to sponsor the events can sponsor the events based on the event details. This platform maintains the blog feature to be in connection with different communities (participants, organizers, and sponsors).

The web platform helps the organizers to register an event easily, provide its description, and other details. The interested people who are searching for the events, can easily and quickly register for the event through our platform, after completion of the event they will be prompted for the feedback/review the event they have attended. The organizers can easily get sponsors through this portal, they can also specify the facilities for the sponsors and can determine the different scale of sponsor types. They can share the required resources and contents through this portal.

This web platform will help the users to manage their events quickly. The platform provides unique feature of share resources and blogs. This helps the organizers to share the prerequisite and other resources to the participants. The blogs help the different users to share their ideas on various topics.

1. **SYSTEM ANALYSIS**
   1. **EXISTING SYSTEM**

The existing websites give the registration links and the news when the hackathon is being spread in the corresponding college using their web platform or might be with the help of the email service or through other social media websites like YouTube, etc. There is no way to ask any participants to avoid circulating or distributing their ideas or feedback of the event which can be displayed on the website publicly [1].

There is no option to write blogs on the website by the participants or sponsors [2]. The registration links are most probably google sheets and such things or it might allow the participant to register on our platform and accordingly the data will be shared with organizer of the particular event, the notification of the events and all will not be provided easily. Most often the organizers of the event create their own webpages for promotion and the registrations. the organizers look into sponsorship at their risk, the participants do not have any common platform to look into the upcoming events.

* + 1. **LIMITATIONS OF THE EXISTING SYSTEM**

Some of the limitations of the existing systems include, candidate/participants have individuals account but cannot make a group to participate in the events. Some websites like Hacker Earth is not available to persons under the age of 18, there is an age limit for the participant. Hacker Earth will not be liable for any loss or damage arising from your failure to provide us with accurate information or to keep your password secure. Printing pages from the website for your own personal use are restricted in Hack summit. Hack summit does not warrant that this website will be constantly available, or available at all or the information on this website is complete, true, accurate or non- misleading.

* 1. **PROPOSED SYSTEM**

The proposed web-based platform that is aimed to help the organizers of the fests, workshops or hackathon events to post their event details, resources and related information. The platform provides a common registration to the posted events, this will help the students to register and participate in the events easily. The companies who would like to sponsor the events can sponsor the events based on the event details.

This platform maintains the blog feature to be in connection with different communities (participants, organizers, and sponsors).

The web-based platform helps the organizer to register an event easily, provide its description, and other details. The interested people who are searching for the events, can easily and quickly register for the event through our website, they will be noticed about the start of the event, after completion they will be prompted for the feedback and review the event they have attended. The organizers can easily get sponsors through this portal. they can also specify the facilities for the sponsors and can determine the different scale of sponsor types. They can share the required resources and contents through this portal.

This web-based platform will help the people who wish to be a part of hackathon, IT fest and other related events. To find the appropriate event to be a part of, biased on the feedback of the previous participants. The web-based platform provides the complete details of the events and hence people can be updated with the recent hackathons and events that are going to happen and the other necessary event details. The web portal helps the users to post the event details, the people who like to participate in various events can look here for the various events, and can choose the best ones from his view, quickly without spending much time in searching for different events, since this platform helps all the organizers to register their events.

* + 1. **ADVANTAGES OF PROPOSED SYSTEM**

The proposed system is a web-based platform that focuses in providing a common platform to share the event details and posters, the registration platform, and the other functionalities like the blog and resource sharing. Each event organizer needs to create their own website or platform to share resources and such things. This system will help the organizers to market their event and can quickly find the sponsors for the event. and well-known database and other assistance.

* 1. **LITERATURE SURVEY**

An online event management system software project that serves the functionality of an event manager. The system allows only registered user login and new users are allowed to register on the application. This proposed to be a web application. The project provides most of the basic functionality required for an IT event type e.g.

Hackathon, Conference, etc. With the customers and events increasing at larger rate, it is difficult to manage using traditional system using spreadsheets, traditional database and more. In order to overcome the drawbacks of traditional Event Managing System, a new Smart Event Management System can be introduced which uses modern technology for managing various tasks and planning for employees, customers, location, resources, etc. [5] Though such type positive events make influential impacts on the students. Sponsors are there to provide funds for various events. The main advantage of using this application is it reduces the direct communication between student and event organizer and avoid the difficulties of the student to enter into event and participating. No matter where the participants and organizer are. Highly enhanced and optimized database design and coding techniques has been used. This makes the application user friendly and easy to use for naive users [6]. This literature demonstrates that some of the main themes which have emerged as important research topics include the economic impact of events and the site selection process of conference and convention organizers.

* 1. **SOFTWARE TOOLS USED**
     1. **FRONT END**
        1. **HTML**

HTML stands for Hyper Text Markup Language and was intended to be used solely to provide document structure for a text-based communication medium. Hypertext Markup Language commonly referred to as HTML is the standard markup language used to create web pages. It is written in the form of HTML elements consisting of tags enclosed in angle brackets (like <html>). HTML tags most commonly come in pairs like <h1> and </h1>, although some tags represent empty elements and so are unpaired, for example <img>. The first tag in a pair is the start tag, and the second tag is the end tag (they are also called opening tags and closing tags). Web browsers can read HTML files and compose them into visible or audible web pages. Browsers do not display the HTML tags and scripts, but use them to interpret the content of the page. HTML describes the structure of a website semantically along with cues for presentation, making it a markup language, rather than a programming language.HTML elements form the building blocks of all websites. HTML allows images and objects to be embedded and can be used to create interactive forms. It provides a means to create

structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items.

* + - 1. **BOOTSTRAP**

Bootstrap is a free and open-source CSS framework directed at responsive, mobile-first front-end web development. It contains CSS- and JavaScript-based design templates for typography, forms, buttons, navigation and other interface components. The Bootstrap framework is built on HTML, CSS, and JavaScript (JS) to facilitate the development of responsive, mobile-first sites and apps.

Bootstrap includes user interface components, layouts and JS tools along with the framework for implementation. The software is available precompiled or as source code. Bootstrap also comes with several JavaScript components in the form of jQuery plugins. They provide additional user interface elements such as dialog boxes, tooltips, and carousels. Each Bootstrap component consists of an HTML structure, CSS declarations, and in some cases accompanying JavaScript code. They also extend the functionality of some existing interface elements, including for example an auto- complete function for input fields.

* + - 1. **JAVASCRIPT**

JavaScript can function as both a procedural and an object-oriented language. Objects are created programmatically in JavaScript, by attaching methods and properties to otherwise empty objects at run time, as opposed to the syntactic class definitions common in compiled languages like C++ and Java. Once an object has been constructed it can be used as a blueprint (or prototype) for creating similar objects. JavaScript's dynamic capabilities include runtime object construction, variable parameter lists, function variables, dynamic script creation (via eval introspection (via for ... in), and source code recovery (JavaScript programs can decompile function bodies back into their source text). JavaScript is also used in environments that are not Web-based, such as PDF documents, site-specific browsers, and desktop widgets. Newer and faster JavaScript virtual machines (VMs) and platforms built upon them have also increased the popularity of JavaScript for server- side Web applications. On the client side, JavaScript has been traditionally implemented as an interpreted language, but more recent browsers perform just-in-time compilation. It is also used in game development,

the creation of desktop and mobile applications, and server-side network programming with run-time environments such as Node.js.

* + 1. **BACK END**
       1. **DJANGO 2.1**

Django is a free and open source web application framework written in Python. A framework is nothing more than a collection of modules that make development easier. They are grouped together, and allow you to create applications or websites from an existing source, instead of from scratch.

This is how websites - even simple ones designed by a single person - can still include advanced functionality like authentication support, management and admin panels, contact forms, comment boxes, file upload support, and more. In other words, if you were creating a website from scratch you would need to develop these components yourself. By using a framework instead, these components are already built, you just need to configure them properly to match your site.

Django offers a big collection of modules which you can use in your own projects. Primarily, frameworks exist to save developers a lot of wasted time and headaches and Django is no different.

* + - 1. **POSTGRESQL**

PostgreSQL is a general purpose and object-relational database management system, the most advanced open source database system. PostgreSQL was developed based on POSTGRES at Berkeley Computer Science Department, University of California.

PostgreSQL was designed to run on UNIX-like platforms. However, PostgreSQL was then also designed to be portable so that it could run on various platforms such as Mac OS X, Solaris, and Windows. PostgreSQL is free and open source software. Its source code is available under PostgreSQL license, a liberal open source license. You are free to use, modify and distribute PostgreSQL in any form.

PostgreSQL requires very minimum maintained efforts because of its stability. Therefore, if you develop applications based on PostgreSQL, the total cost of ownership is low in comparison with other database management systems. PostgreSQL features

transactions with Atomicity, Consistency, Isolation, Durability (ACID) properties, automatically updatable views, materialized views, triggers, foreign keys, and stored procedures PostgreSQL is developed by the PostgreSQL Global Development Group, a diverse group of many companies and individual contributors

PostgreSQL has many advanced features that other enterprise database management systems offer, such as: User-defined types, Table inheritance, Sophisticated locking mechanism, Foreign key referential integrity, Views, rules and subquery, Nested transactions (save points), Multi-version concurrency control (MVCC) and Asynchronous replication

* 1. **SOFTWARE USED**
     1. **GITHUB**

GitHub is a website and cloud-based service that helps developers store and manage their code, as well as track and control changes to their code. GitHub is a for-profit company that offers a cloud-based Git repository hosting service. Essentially, it makes it a lot easier for individuals and teams to use Git for version control and collaboration.

GitHub’s interface is user-friendly enough so even novice coders can take advantage of Git. Without GitHub using Git generally requires a bit more technical savvy and use of the command line. GitHub have helped us to collaborate with each other and to synchronize and smooth way at the time of developing, integrating the projects modules.

* + 1. **SUBLIME TEXT / VISUAL STUDIO CODE**

Sublime Text or Visual Studio code are text editors with a Python application programming interface (API). It natively supports many programming languages and markup languages, and functions can be added by users with plugins.

There are many features that allows users to select entire columns at once or place more than one cursor in text, which allows for simultaneous editing. All cursors then behave as if each of them was the only one in the text. Commands like move by character, move by line, text selection, move by words, move by sub words, move to beginning/end of line, etc. Affect all cursors independently, allowing one to edit slightly complex repetitive structures quickly without the need to use macros or regex.

Sublime Text and Visual studio code will offer to complete entries as the user is typing depending on the language being used. It also auto-completes variables created by the user. The dark background is intended to reduce eyestrain and increase the amount of contrast with the text. Syntax highlighting also makes syntaxes of the language easier to read. These editors allow users to run code for certain languages from within the editor, which eliminates the need to switch out to the command line and back again. This function can also be set to build the code automatically every time the file is saved.

This editor will also allow users to save blocks of frequently used code and assign keywords to them. The user can then type the keywords and press tab to paste the block of code whenever they require it. This feature is a tool that allows users to switch between open, recent or project files and also navigate to symbols within them.

* + 1. **PGADMIN4 (POSTGRESQL IDE)**

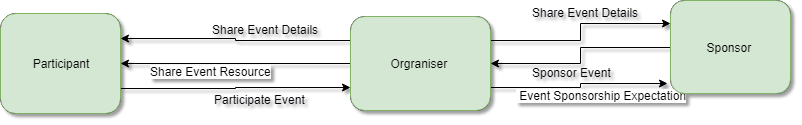
pgAdmin is an open source software for the development and administration of PostgreSQL databases and derived databases such as Enterprise DB Postgres Plus Advanced Server. A graphical user interface facilitates the administration of databases. The SQL Query Editor includes a graphical EXPLAIN that allows you to build more powerful queries. With a native connection to PostgreSQL, the GUI provides access to all PostgreSQL functionality.

PgAdmin provides the functionalities like use of one or more servers simultaneously, slick and pretty web-based Interface, responsive layout, no full-page loading, used AJAX for partial loading, dock-able panels, no sub-dialogues when working with any Postgres object

All the PostgreSQL database objects like servers, databases, table-spaces, roles, extensions, schemas, tables, indexes, constraints, triggers, functions etc. are supported. CRUD operations for all the PostgreSQL database objects. Separate panels for, reverse engineered SQL, statistics, and dependents.

The Query Tool provides Execute arbitrary SQL commands, Export Data to a csv file, Graphical explain plan, View Analytical information about a SQL statement, auto complete and Import SQL script through File Manager. and the View Data helps easy View / Modify data, limited by tables with primary keys, Filter the result set and Export Data as csv file.

1. **SYSTEM REQUIREMENTS**
   1. **SYSTEM MODEL**



**Fig. 3.1 System Model**

The project has three main users the participant, organizer and Sponsors of any events like Hackathon, It Fest, Conference, Seminars and so on. The organizer creates the event and share the event details and resources with the participants, the interested participants register in the events and take part in the events. The sponsors who get the event details and sponsorship details and choose to sponsor the event seeing the facilities provided. The entire project focuses on providing a common platform to share the event details and posters, the registration platform, and the other functionalities like the blog and resource sharing. Each event organizer needs to create their own website or platform to share resources and such things. This system will help the organizers to market their event and can quickly find the sponsors for the event.

* 1. **USER REQUIREMENTS**
     1. **FUNCTIONAL REQUIREMENTS**
        1. **PARTICIPANT**

This module concentrates on the main users where people who wish to participate in different events will register themselves in the event details. They can participate in the event participant can look for prerequisite required for that event also participant can look in events before he have participated. If he likes to write a blog, he can write the blogs as well

* + - 1. **SPONSORS**

This module yields the details about the sponsor of an event, the sponsorship as a three kind of sponsorship as Platinum, Gold and Silver, the sponsor will be given the event

report. Sponsors can check the details of the events and then select the sponsorship type. Sponsor get the whole details of that event.

* + - 1. **ORGANIZERS**

Organizer of the events will post the details of the event and what kind of facilities will be provided to the sponsors also what kind of prizes and Rubrics of the event; they can even add the resources based on the event requirement. Organizer can share the prerequisite required for the hosting event even though he has the facility provide to share the resource with GitHub or with google drive.

* + - 1. **BLOG**

User can write in their respective thoughts and also can share the feedback of the event to the entire users. They have the only facility to write the blog in technical areas as of now. We might add in future.

* + - 1. **COMMUNITY**

There will be three types of communities for each of the main users, the sponsors, organizers and participants. Communities help the different users to share their common ideas and thoughts.

* + - 1. **RESOURCES**

Prerequisite and Resources after the hackathon and events will be provided kind of sharing resources through GitHub platform or google drive. Resources will be shared to the participants which will help them to effectively participate in the events.

* + - 1. **QUERY**

Query will be a module that will help the people to enquire.

* + - 1. **ACCOUNTS**

This module will manage the different user registration and login facilities.

* + - 1. **ABOUT US**

This module will maintain the section about of the web portal. We have planned to add each and every thing to be dynamic as far as UI changes are not required.

* + 1. **NON-FUNCTIONAL REQUIREMENTS**
       1. Easy registration and information about events like (IT Fest, Hackathon, Conferences and so on.)
       2. Create events quickly and make available to all
       3. Share resources for event in an easy and efficient platform
       4. Write the review on events to express the experience
       5. Easy to get sponsors for the events
       6. Remainders about the events to the registered event
       7. Share your views through blog to the interested.
  1. **SYSTEM REQUIREMENTS**
     1. **HARDWARE REQUIREMENTS**

The proposed system does not have any explicit hardware requirements.

* + - 1. Operating System: Windows 7/8/10, ubuntu etc.
      2. Processor: Intel i3 and above
      3. Memory: 4GB
      4. Hard Disk Space: 50GB
    1. **SOFTWARE REQUIREMENTS**

The below are the specific software requirements for the system to run successfully.

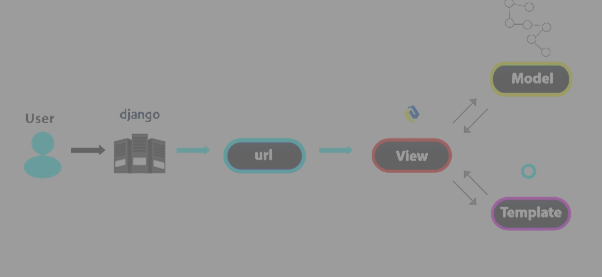
* + - 1. **FRONT END**
         1. Html 4.0+
         2. Bootstrap
         3. JavaScript
      2. **BACK END**
         1. Django 2.1+
         2. PostgreSQL 11.0+
      3. **SOFTWARE USED**
         1. GitHub Desktop
         2. Sublime text/Visual studio code
         3. PGAdmin4 (PostgreSQL IDE)

1. **DESIGN SPECIFICATION**
   1. **ARCHITECTURAL DESIGN**

Django is a Python-based free and open-source web framework, which follows the model-template-view (MTV) architectural pattern It is maintained by the Django Software Foundation (DSF), an independent organization established as a non-profit.

Django's primary goal is to ease the creation of complex, database-driven websites. The framework emphasizes reusability and "pluggability" of components, less code, low coupling, rapid development, and the principle of don't repeat yourself. Python is used throughout, even for settings files and data models. Django also provides an optional administrative create, read, update and delete interface that is generated dynamically through introspection and configured via admin models.

The project architecture is in such a way that when a user sends a URL GET request we send it our application URLs and look for the same pattern of URL we have received and if that URL matches we sends the request to views where we perform the required action as per the request and if it is required to send the data to the database we access model of the required table and store or retrieve the response through templates.

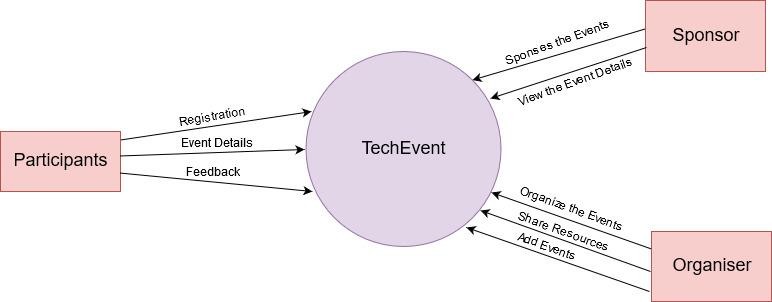


**Fig. 4.1 Project architecture**

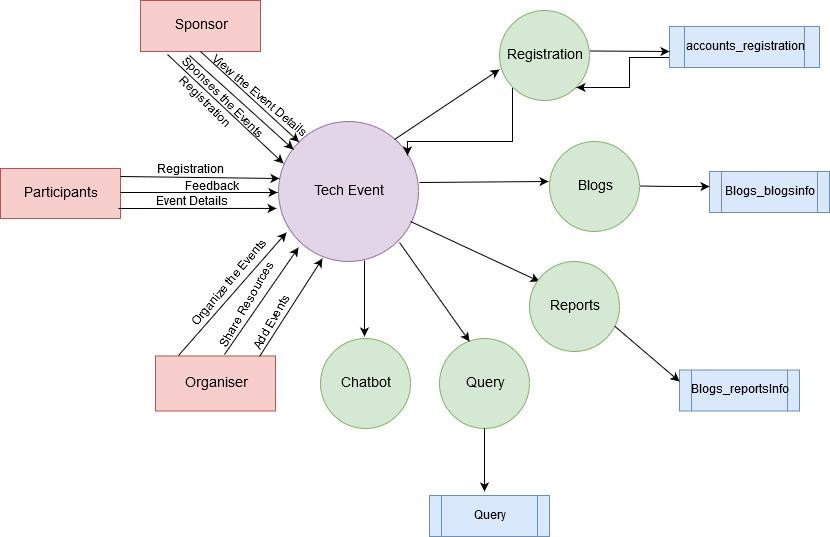
* 1. **DATA FLOW DIAGRAM**

The context diagram also called as the Level 1 DFD (Fig 4.2 level 0 DFD) depicts the overall data flow in the system. The Figure (Fig 4.3 level 1 DFD) depicts the operations on the system and the more detailed details of the data flow has been included in the Figure (Fig 4.4 level 2 DFD). These figures constitute the data flow representation in the system.

* + 1. **CONTEXT DIAGRAM (LEVEL 0 DFD)**

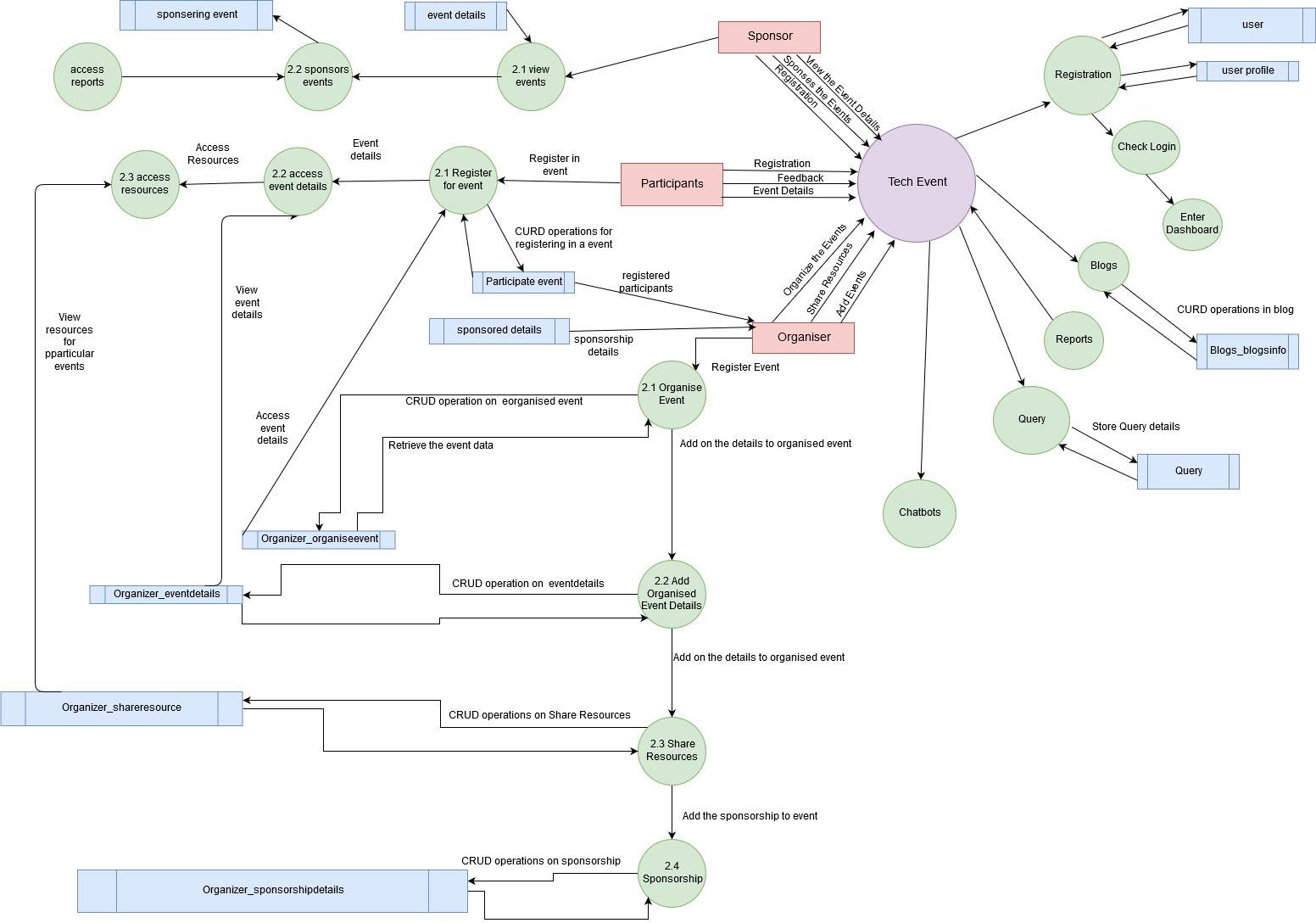


**Fig. 4.2 level 0 DFD**

* + 1. **LEVEL 1 DFD**

**Fig. 4.3 level 1 DFD**

* + 1. **LEVEL 2 DFD**

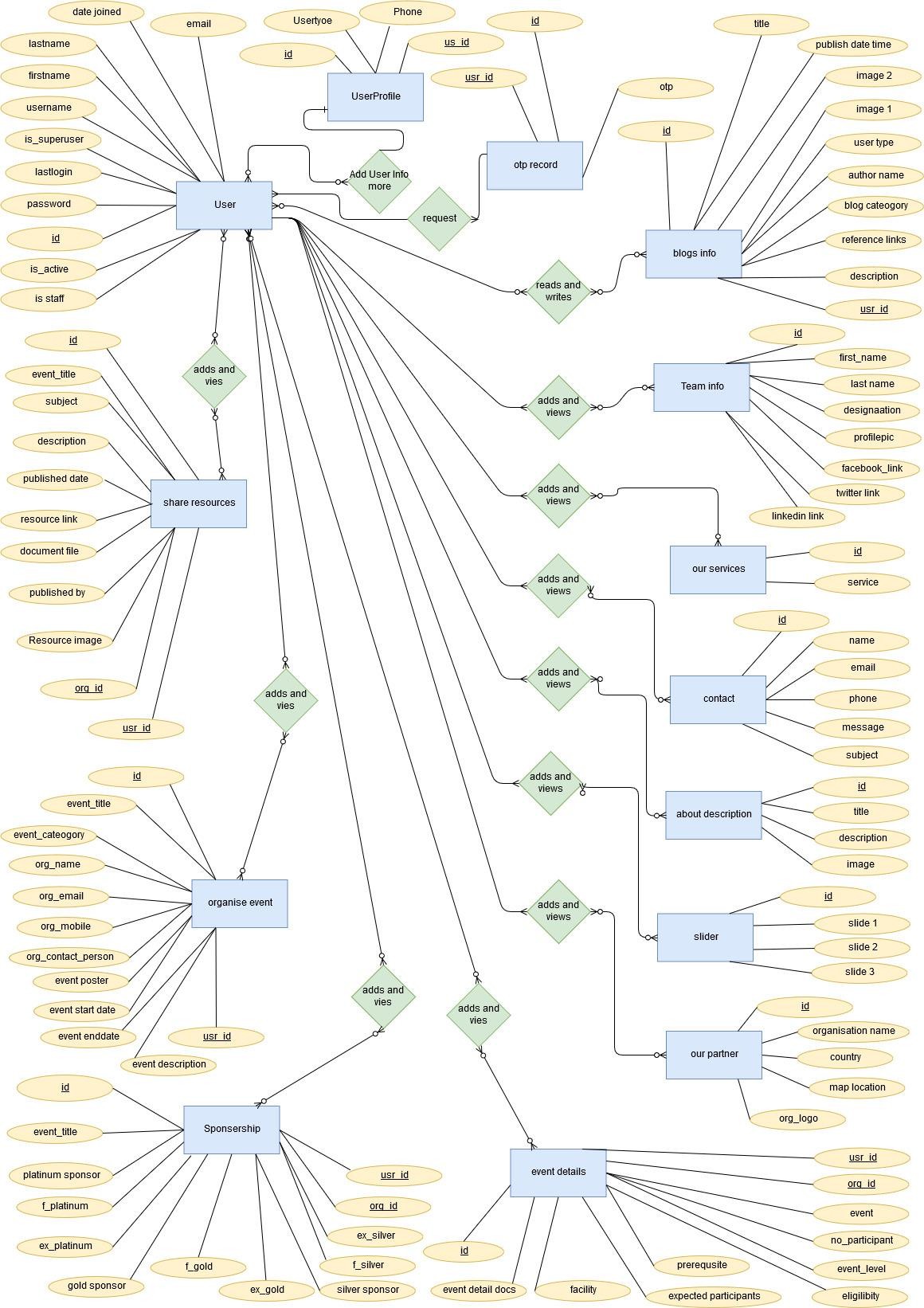


**Fig 4.4 level 2 DFD**

* 1. **ENTITY RELATIONSHIP DIAGRAM**

The entity relationship diagram describes interrelated things in the project. A basic ER model is composed of entity types and specifies relationships that can exist between instances of those entity types. An entity relationship diagram shows the relationships of entity sets stored in a database. An entity in this context is an object, a component of data. An entity set is a collection of similar entities. These entities can have attributes that define its properties.

The entity relationship diagram (Fig. 4.5 ER Diagram), represents the inter relativity of the different project components and how it is related together. the different modules of the project. the entity relationship diagram helps in the design and development of the database.



**Fig. 4.5 ER Diagram**

* 1. **DATABASE DESIGN**
     1. **USER**

Table 4.1 user

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute Name** | **Data Type** | **Constraints** | **Description** |
| id | integer | Primary Key | identify each user |
| password | character varying (128) | Not Null | stores password |
| last\_login | timestamp | Not Null | last login |
| is\_superuser | boolean | Not Null | superuser or not |
| username | character varying (150) | Not Null | stores username |
| first\_name | character varying (30) | Not Null | stores first name |
| last\_name | character varying (30) | Not Null | stores last name |
| email | character varying (256) | Not Null | stores email |
| is\_staff | boolean | Not Null | staff user or not |
| is\_active | boolean | Not Null | active or not |
| date\_joined | timestamp | Not Null | joined date |

* + 1. **USERPROFILE**

Table 4.2 userprofile

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute Name** | **Data Type** | **Constraints** | **Description** |
| id | integer | Primary Key | ID of the table |
| u\_id | integer | Foreign Key | user table ID |
| type\_usr | integer | Not Null | user type |
| phone | character varying (15) | Not Null | phone number |
| profile\_img | character varying (100) | Not Null | profile pic |

## BLOGS INFO

Table 4.3 blogs\_info

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute Name** | **Data Type** | **Constraints** | **Description** |
| id | integer | Primary Key | ID of the table |
| title | character varying (50) | Not Null | blog title |
| pub\_date\_time | timestamp | Not Null | publishing date |
| description | character varying (1000) | Not Null | description |
| imagesecond | character varying (100) | Not Null | image |
| imagefirst | character varying (100) | Not Null | image |
| usertype | character varying (30) | Not Null | user type |
| author\_name | character varying (30) | Not Null | author name |
| blog\_category | character varying (30) | Not Null | blog category |
| reference\_links | character varying (100) | Not Null | reference links |
| us\_id | integer | Foreign Key | user ID |

## EVENT LOCATION

Table 4.4 eventlocation

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute Name** | **Data Type** | **Constraints** | **Description** |
| id | integer | Primary Key | ID of the table |
| event\_venue\_name | character varying (100) | Not Null | venue name |
| event\_venue\_addr | character varying (300) | Not Null | venue address |
| event\_latitude | character varying (100) | Not Null | latitude |
| event\_longitude | character varying (100) | Not Null | longitude |
| event\_name | character varying (100) | Not Null | name |
| eventid\_id | integer | Foreign Key | event ID |

## EVENT DETAILS

Table 4.5 event\_details

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute Name** | **Data Type** | **Constraints** | **Description** |
| id | integer | Primary Key | ID of the table |
| event | character varying (100) | Not Null | event name |
| expected\_participant | integer | Not Null | No. of teams |
| no\_participant | integer | Not Null | No. of participants |
| event\_level | character varying (100) | Not Null | level of event |
| eligibility | character varying (100) | Not Null | eligibility |
| prerequisite | character varying (300) | Not Null | prerequisite |
| facility | character varying (100) | Not Null | facilities |
| event\_detail\_docs | character varying (100) | Not Null | details |
| org\_id\_id | integer | Foreign Key | organizer ID |
| us\_id | integer | Foreign Key | user ID |

## ORGANISE\_EVENT

Table 4.6 organis\_eevent

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute Name** | **Data Type** | **Constraints** | **Description** |
| id | integer | Primary Key | ID of the table |
| event\_title | character varying (100) | Not Null | event title |
| event\_description | character varying (800) | Not Null | event description |
| event\_category | character varying (200) | Not Null | category |
| org\_name | character varying (200) | Not Null | organization name |
| org\_email | character varying (100) | Not Null | email |
| org\_mobile | biginteger | Not Null | mobile number |
| org\_contact\_person | character varying (100) | Not Null | contact person |
| event\_poster | character varying (100) | Not Null | poster image |

|  |  |  |  |
| --- | --- | --- | --- |
| event\_start\_date | timestamp | Not Null | start date |
| event\_end\_date | timestamp | Not Null | end date |
| us\_id | integer | Foreign Key | user ID |

## SHARE RESOURCE

Table 4.7 share\_resource

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute Name** | **Data Type** | **Constraints** | **Description** |
| id | integer | Primary Key | ID of table |
| event\_title | character varying (100) | Not Null | event name |
| subject | character varying (100) | Not Null | subject |
| description | character varying (400) | Not Null | description |
| published\_date | timestamp | Not Null | publishing date |
| resource\_link | character varying (100) | Not Null | link |
| document\_file | character varying (100) | Not Null | document file |
| published\_by | character varying (100) | Not Null | publisher |
| resource\_image | character varying (100) | Not Null | resource image |
| us\_id | integer | Foreign Key | user ID |
| org\_id\_id | integer | Foreign Key | organizer ID |

## SPONSORSHIP

Table 4.8 sponsorship

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute Name** | **Data Type** | **Constraints** | **Description** |
| id | integer | Primary Key | ID of the table |
| event\_title | character varying (100) | Not Null | event name |
| platinum\_sponsor | character varying (800) | Not Null | sponsor type |
| f\_platinum | character varying (200) | Not Null | facilities |

|  |  |  |  |
| --- | --- | --- | --- |
| ex\_platinum | integer | Not Null | amount |
| gld\_sponsor | character varying (100) | Not Null | sponsor type |
| f\_gold | character varying (200) | Not Null | facilities |
| ex\_gold | integer | Not Null | amount |
| silver\_sponsor | character varying (100) | Not Null | sponsor type |
| f\_silver | character varying (200) | Not Null | facilities |
| ex\_silver | integer | Not Null | amount |
| us\_id | integer | Foreign Key | user ID |
| org\_id\_id | integer | Foreign Key | organizer ID |

## EVENT FEEDBACK

Table 4.9 event\_feedback

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute Name** | **Data Type** | **Constraints** | **Description** |
| id | integer | Primary Key | ID of table |
| subject | character varying (100) | Not Null | subject |
| feedback | character varying (800) | Not Null | feedback |
| pub\_date\_time | timestamp | Not Null | date |
| event\_id\_id | integer | Foreign Key | event ID |
| us\_id | integer | Foreign Key | user ID |

## PARTICIPATE EVENT

Table 4.10 participate\_event

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute Name** | **Data Type** | **Constraints** | **Description** |
| id | integer | Primary Key | ID of table |
| us\_id | integer | Foreign Key | user ID |
| event\_id\_id | integer | Foreign Key | event ID |

## SPONSOR TRANSCATION

Table 4.11 sponsor\_transaction

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute Name** | **Data Type** | **Constraints** | **Description** |
| id | integer | Primary Key | ID of the table |
| sponsor\_type | character varying (100) | Not Null | sponsor type |
| amount | integer | Not Null | amount |
| Event\_id\_id | integer | Foreign Key | event ID |
| us\_id | integer | Foreign Key | user ID |

## ABOUT DESCRIPTION

Table 4.12 about\_description

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute Name** | **Data Type** | **Constraints** | **Description** |
| id | integer | Primary Key | ID of table |
| title | character varying (100) | Not Null | title |
| description | character varying (100) | Not Null | description |
| image | character varying (100) | Not Null | image |

## CONTACT

Table 4.13 contact

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute Name** | **Data Type** | **Constraints** | **Description** |
| id | integer | Primary Key | ID of table |
| name | character varying (100) | Not Null | name |
| email | character varying (100) | Not Null | email |
| subject | character varying (100) | Not Null | subject |
| phone | character varying (100) | Not Null | phone |
| message | character varying (1000) | Not Null | message |

## OUR SERVICES

Table 4.14 our\_services

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute Name** | **Data Type** | **Constraints** | **Description** |
| id | integer | Primary Key | ID of table |
| service\_name | character varying (100) | Not Null | service name |

## TEAM INFO

Table 4.15 team\_info

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute Name** | **Data Type** | **Constraints** | **Description** |
| id | integer | Primary Key | ID of table |
| first\_name | character varying (100) | Not Null | first name |
| last\_name | character varying (100) | Not Null | last name |
| designation | character varying (100) | Not Null | designation |
| profile\_pic | character varying (100) | Not Null | image |
| facebook\_link | character varying (100) | Not Null | link Facebook |
| twitter\_link | character varying (100) | Not Null | link twitter |
| linkedin\_link | character varying (100) | Not Null | link linkedIn |

## ARTICLE

Table 4.16 article

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute Name** | **Data Type** | **Constraints** | **Description** |
| id | integer | Primary Key | ID of table |
| title | character varying (1000) | Not Null | title |
| desc | character varying (1000) | Not Null | description |

## ADMIN LOG

Table 4.17 admin\_log

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute Name** | **Data Type** | **Constraints** | **Description** |
| id | integer | Primary Key | ID of table |
| action\_time | timestamp | Not Null | action details |
| object\_id | text | Not Null | object |
| object\_repr | character varying (200) | Not Null | description |
| action\_flag | smallint | Not Null | flag |
| change\_message | text | Not Null | message |
| content\_type\_id | integer | Not Null | content type |
| user\_id | integer | Foreign Key | user ID |

## OUR PARTNER

Table 4.18 our\_partner

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute Name** | **Data Type** | **Constraints** | **Description** |
| id | integer | Primary Key | ID of table |
| organisation\_name | character varying (100) | Not Null | organization name |
| country | character varying (100) | Not Null | country |
| map\_location | character varying (100) | Not Null | map location |
| org\_logo | character varying (100) | Not Null | logo |

## SLIDER

Table 4.18 slider

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute Name** | **Data Type** | **Constraints** | **Description** |
| id | integer | Primary Key | ID of table |
| slide\_1 | character varying (100) | Not Null | image |
| slide\_2 | character varying (100) | Not Null | image |
| slide\_3 | character varying (100) | Not Null | image |

# IMPLEMENTATION DETAILS

## SOURCE CODE

* + 1. **TECHEVENT**
       1. **Settings.py**

"""

Django settings for Hackathon project.

Generated by 'django-admin startproject' using Django 2.1.

For more information on this file, see https://docs.djangoproject.com/en/2.1/topics/settings/

For the full list of settings and their values, see https://docs.djangoproject.com/en/2.1/ref/settings/ """

import os

from django.contrib.messages import constants as messages

# Build paths inside the project like this: os.path.join(BASE\_DIR, ...) BASE\_DIR = os.path.dirname(os.path.dirname(os.path.abspath( file )))

# Quick-start development settings - unsuitable for production

# See https://docs.djangoproject.com/en/2.1/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret! SECRET\_KEY = 'imo@uk+=u)zhh-9+ateup+vy^ #wfcyi7gx()$\_+$&htkw=ju'

# SECURITY WARNING: don't run with debug turned on in production! DEBUG = True

ALLOWED\_HOSTS = []

# Application definition INSTALLED\_APPS = [

'aboutus.apps.AboutusConfig', 'accounts.apps.AccountsConfig', 'Blog.apps.BlogConfig', 'Events.apps.EventsConfig', 'home.apps.HomeConfig', 'Organizer.apps.OrganizerConfig', 'Participant.apps.ParticipantConfig', 'Report.apps.ReportConfig', 'Sponsor.apps.SponsorConfig', 'django.contrib.admin', 'django.contrib.auth', 'django.contrib.contenttypes',

'django.contrib.sessions', 'django.contrib.messages', 'django.contrib.staticfiles',

]

MIDDLEWARE = [

'django.middleware.security.SecurityMiddleware', 'django.contrib.sessions.middleware.SessionMiddleware', 'django.middleware.common.CommonMiddleware', 'django.middleware.csrf.CsrfViewMiddleware', 'django.contrib.auth.middleware.AuthenticationMiddleware', 'django.contrib.messages.middleware.MessageMiddleware', 'django.middleware.clickjacking.XFrameOptionsMiddleware',

]

ROOT\_URLCONF = 'Hackathon.urls' TEMPLATES = [

{

'BACKEND': 'django.template.backends.django.DjangoTemplates', 'DIRS': [os.path.join(BASE\_DIR, 'templates')],

'APP\_DIRS': True, 'OPTIONS': {

'context\_processors': [ 'django.template.context\_processors.debug', 'django.template.context\_processors.request', 'django.contrib.auth.context\_processors.auth', 'django.contrib.messages.context\_processors.messages',

],

},

},

]

WSGI\_APPLICATION = 'Hackathon.wsgi.application'

# Database

# https://docs.djangoproject.com/en/2.1/ref/settings/#databases

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.postgresql', 'NAME': 'Hackathon',

'USER': 'postgres',

'PASSWORD': '10875',

'HOST': 'localhost',

'PORT': '5432',

}

}

# Password validation

# https://docs.djangoproject.com/en/2.1/ref/settings/#auth-password-validators

AUTH\_PASSWORD\_VALIDATORS = [

{

'NAME': 'django.contrib.auth.password\_validation.UserAttributeSimilarityValidator',

},

{

'NAME': 'django.contrib.auth.password\_validation.MinimumLengthValidator',

},

{

'NAME': 'django.contrib.auth.password\_validation.CommonPasswordValidator',

},

{

'NAME': 'django.contrib.auth.password\_validation.NumericPasswordValidator',

},

]

# Internationalization

# https://docs.djangoproject.com/en/2.1/topics/i18n/ LANGUAGE\_CODE = 'en-us'

TIME\_ZONE = 'UTC'

USE\_I18N = True USE\_L10N = True USE\_TZ = True

# Static files (CSS, JavaScript, Images)

# https://docs.djangoproject.com/en/2.1/howto/static-files/

STATICFILES\_DIRS = [

os.path.join(BASE\_DIR, 'Hackathon/static'),

]

STATIC\_ROOT = os.path.join(BASE\_DIR, 'static') STATIC\_URL = '/static/'

MEDIA\_ROOT = os.path.join(BASE\_DIR, 'media') MEDIA\_URL = '/media/'

# Message MESSAGE\_TAGS = {

messages.ERROR: 'danger'

}

EMAIL\_HOST = 'smtp.gmail.com' EMAIL\_PORT = 25

EMAIL\_HOST\_USER = 'sachin.thakur9614@gmail.com' EMAIL\_HOST\_PASSWORD = 'hjbgycwcbqra' EMAIL\_USE\_TLS = True

AUTHENTICATION = 'plain' OPENSSL\_VERIFY\_MODE = 'none'

## Urls.py

from django.contrib import admin from django.urls import path, include from Blog import views

from django.conf import settings

from django.conf.urls.static import static urlpatterns = [

path('', include('home.urls'), name='home'), path('aboutus/', include('aboutus.urls'), name='aboutus'),

path('accounts/', include('accounts.urls'), name='accounts'), path('admin/', admin.site.urls),

path('Blog/', include('Blog.urls'), name='Blog'), path('Events/', include('Events.urls'), name='Events'),

path('Organizer/', include('Organizer.urls'), name='Organizer'), path('Participant/', include('Participant.urls'), name='Participant'), path('Report/', include('Report.urls'), name='Report'), path('Sponsor/', include('Sponsor.urls'), name='Sponsor'),

] + static(settings.MEDIA\_URL, document\_root=settings.MEDIA\_ROOT)

## ACCOUNTS

* + - 1. **Models.py**

from django.db import models

from django.contrib.auth.models import User from django.db.models.signals import post\_save from django.dispatch import receiver

class Registration(models.Model):

firstName = models.CharField(max\_length=25) lastName = models.CharField(max\_length=25) email = models.EmailField(max\_length=50) password = models.CharField(max\_length=40)

confirmPassword = models.CharField(max\_length=40) mobile = models.BigIntegerField()

userType = models.CharField(max\_length=50, blank=True) # add is published and status active one!

class Article(models.Model):

title = models.CharField(max\_length=1000) desc = models.CharField(max\_length=1000)

class UserProfile(models.Model): # u\_id = models.IntegerField() type\_usr = models.IntegerField()

phone = models.CharField(max\_length=15) profile\_img=models.ImageField(upload\_to='media/images/Profile/')

def str (self):

return self.phone

## Urls.py

from django.urls import path, include

from accounts import views urlpatterns = [

path('', views.indx, name='accounts'), path('signup', views.signup, name='signup'), path('logout', views.logout, name='logout'), path('login', views.loginn, name='login'),

path('organizer', include('Organizer.urls'), name='organiserIndex'), path('sponsor', include('Sponsor.urls'), name='sponsorIndex'), path('participant/', include('Participant.urls'), name='participantIndex'),

]

## Views.py

from django.shortcuts import render, redirect, get\_object\_or\_404 from django.contrib import auth

from django.urls import path, include

from django.contrib.auth.models import User from django.utils import timezone

from django.contrib.auth import authenticate, login, logout from django.http import HttpResponseRedirect, HttpResponse from django.urls import reverse

from django.contrib.auth.decorators import login\_required from .models import Article, UserProfile

from django.core.mail import send\_mail from django.contrib import messages

from django.views.generic import TemplateView from .forms import ArticleForm

def loginn(request):

if request.method == 'POST': email = request.POST['email']

password = request.POST['password']

user = auth.authenticate(username=email, password=password) if user is not None:

key = user.\_get\_pk\_val()

user\_1 = list(UserProfile.objects.values\_list( 'type\_usr', 'id').filter(id=key).values('type\_usr'))

user\_type = user\_1[0]

var = user\_type['type\_usr']

if user\_type['type\_usr'] == 100: auth.login(request, user) request.session['userid'] = key return redirect('participantIndex') # return redirect('participantIndex')

elif user\_type['type\_usr'] == 200: auth.login(request, user) request.session['userid'] = key return redirect('sponsorIndex')

elif user\_type['type\_usr'] == 300:

auth.login(request, user) request.session['userid'] = key return redirect('organiserIndex')

else:

messages.error(

request, 'invalid user type dectected contact admin')

return render(request, 'login.html', {'error': 'invalid user type dectected contact admin'})

else:

return render(request, 'login.html', {'error': 'Invalid credentials'})

else:

return render(request, 'login.html')

def logout(request):

if request.method == 'POST': auth.logout(request)

messages.success(request, ' You are now loged out') return render(request, 'login.html')

def signup(request):

if request.method == 'POST':

first\_name = request.POST['firstname'] last\_name = request.POST['lastname'] username = request.POST['username'] email = request.POST['email'] password = request.POST['password']

password2 = request.POST['confirmpassword'] phone = request.POST['phone']

type\_usr = request.POST['type\_usr'] src = request.POST['profile\_img'] #mobile = request.POST['mobile'] #userType = request.POST['usertype'] if password == password2:

if User.objects.filter(username=username).exists(): messages.error(request, "That username is taken")

return render(request, 'login.html', {'errorr': 'Username is already taken'}) # return redirect('accounts')

else:

if User.objects.filter(email=email).exists(): messages.error(request, "That email is been use")

return render(request, 'login.html', {'errorr': 'Email is in use '+email}) return redirect('accounts')

else:

user = User.objects.create\_user(

username=username, password=password, email=email, first\_name=first\_name, last\_name=last\_name)

# auth.login(request,user) #messages.success(request,'You are now Loged in') user.save()

key = user.\_get\_pk\_val()

var = UserProfile(id=key, type\_usr=type\_usr, phone=phone, profile\_img=src)

var.save() message = ""

if type\_usr == 100:

message = "User Name :"+email + " PASSWORD :" + password + \

" EMAIL : "+email+" PHONE : "+phone+" Role : PARTICIPANT"

elif type\_usr == 200:

message = "User Name :"+email + " PASSWORD :" + password + \ " EMAIL : "+email+" PHONE : "+phone+" Role : SPONSER"

elif type\_usr == 300:

message = "User Name :"+email + " PASSWORD :" + password + \ " EMAIL : "+email+" PHONE : "+phone+" Role : ORGANISER"

send\_mail('Subject' + "YOU HAVE NOW REGISTERED TO TECH EVENT",

'Message' + message + 'Will Contact your soon', 'sachin.thakur9614@gmail.com',

[email, 'sachin.thakur@mca.christuniversity.in', ], fail\_silently=True)

messages.success(request, 'You are now registered in') return redirect('accounts')

elif first\_name == '':

messages.error(request, 'FirstName field is empty') elif last\_name == '':

messages.error(request, 'Last Name field is empty') elif username == '':

messages.error(request, 'Username field is empty') elif email == '':

messages.error(request, 'Email field is empty') elif password == '':

messages.error(request, 'Password is empty') elif password2 == '':

messages.error(request, 'Confirm password field is empty') else:

messages.error(request, 'Password do not match') return redirect('accounts')

else:

return render(request, 'base.html')

def indx(request):

return render(request, 'login.html')

## login.html

{% extends 'base.html' %}

{%block content %}

{%load staticfiles %}

<br><br><br>

<div class="container col-md-10">

<div class="row shadow p-3 mb-5 bg-white rounded">

<div class="col-md-6">

<h3 class="text-center">Login</h3>

<hr class="bg-info col-md-8">

<script type="text/javascript">

//validation

function validate\_login() {

var reg\_email = /^([A-Za-z0-9\_\-\.]){1,}\@([A-Za-z0-9\_\-\.]){1,}\.([A-Za-z]{2,4})$/; var email = document.forms["login\_form"]["email"].value;

if (reg\_email.test(email) == false) {

var msg = "INVALID EMAIL ADDRESS. PLEASE ENTER A VALID EMAIL";

document.getElementById("error\_title\_div").style.visibility = 'visible'; document.getElementById("error\_para").innerHTML = msg;

// document.getElementById("error\_email").innerHTML = msg;

document.forms["login\_form"]["email"].focus(); return false;

}

}

</script>

{% include 'alerts.html' %}

<form method="POST" action="{% url 'login' %}" name="login\_form" onsubmit="return validate\_login()">

{% csrf\_token %}

<div class="alert alert-info alert-dismissible fade show" role="alert" id="error\_title\_div" style="visibility: hidden">

<p id="error\_para"></p>

<button type="button" class="close" data-dismiss="alert">

<span aria-hidden="true">&times;</span>

</button>

</div>

<div class="form-group">

<label for="email">Email&ast; </label>

<input type="text" class="form-control border border-info" id="email" name="email" required aria-describedby="emailHelp" placeholder="Email">

</div>

<div class="form-group">

<label for="Password">Password&ast; </label>

<input type="Password" class="form-control border border-info" name="password" id="password" required

aria-describedby="emailHelp" placeholder="Password">

</div>

<div class="form-group">

<input type="submit" class="btn btn-outline-info bg-dark" id="password" aria- describedby="emailHelp"

value="Submit">

<!-- <a href="">forgot password ?</a> -->

<!--Add buttons to initiate auth sequence and sign out--> <br>

</div>

</form>

<br>

<br>

<div class="row">

<div class="col-md-6">

<button class="btn btn-outline-info bg-dark" id="authorize\_button" style="">Sign in with Google</button>

</div>

<div class="col-md-6">

<button class="btn btn-outline-info bg-dark" id="authorize\_button" style="">Sign in with Facebook</button>

</div>

</div>

<button class="btn btn-outline-info bg-dark" id="signout\_button" style="display: none;">Sign Out</button>

</div>

<div class="col-md-6">

<h3 class="text-center">Registration</h3>

<hr class="bg-info col-md-8">

{% if error %}

<div class="alert alert-info alert-dismissible fade show" role="alert">

<strong> {{error}}</strong>

<button type="button" class="close" data-dismiss="alert" aria-label="Close">

<span aria-hidden="true">&times;</span>

</button>

</div>

{%endif%}

<!-- Alerts -->

{% include 'alerts.html' %}

<form action=" {% url 'signup' %}" method="POST" name="reg\_form" onsubmit="return validate\_registration(this);">

{% csrf\_token %}

<div class="alert alert-info alert-dismissible fade show" role="alert" id="error\_reg\_div" style="visibility: hidden">

<p id="error\_reg\_para"></p>

<button type="button" class="close" data-dismiss="alert">

<span aria-hidden="true">&times;</span>

</button>

</div>

<div class="form-group">

<label for="firstname">First Name&ast; </label>

<input type="text" class="form-control border border-info" required name="firstname" id="firstname"

aria-describedby="emailHelp" placeholder="First Name">

</div>

<div class="form-group">

<label for="lastname">Last Name&ast; </label>

<input type="text" class="form-control border border-info" required name="lastname" id="lastname"

aria-describedby="emailHelp" placeholder="Last Name">

</div>

<div class="form-group">

<label for="city">User name&ast; </label>

<input type="text" class="form-control border border-info" required name="username" id="username"

aria-describedby="emailHelp" placeholder="User Name ">

</div>

<div class="form-group">

<label for="email">Email&ast; </label>

<input type="text" class="form-control border border-info" required name="email" id="email" aria-describedby="emailHelp" placeholder="Email ">

</div>

<div class="form-group">

<label for="Password">Password&ast; </label>

<input type="Password" class="form-control border border-info" required name="password" id="password"

placeholder="Password">

</div>

<div class="form-group">

<label for="Password">Confirm Password&ast; </label>

<input type="Password" class="form-control border border-info" required name="confirmpassword"

id="confirmpassword" placeholder=" Comfirm Password">

</div>

<div class="form-group">

<label for="phone">Mobile Number&ast; </label>

<input type="text" class="form-control border border-info" required name="phone" id="phone" placeholder="Mobile Number">

</div>

<div class="form-group">

<label for="type\_usr"> User Role&ast; </label>

<select name="type\_usr" id="type\_usr" class="form-control border border-info">

<option>Select Your Role</option>

<option value='100'>Participant</option>

<option value='200'>Sponser</option>

<option value='300'>Organiser</option>

</select>

</div>

<div class="form-group">

<label for="profile\_img">Profile Picture&ast;</label>

<input type="file" name="profile\_img" id="profile\_img" class="form-control border border- info">

</div>

<div class="form-group">

<input type="submit" class="btn btn-outline-info bg-dark" id="password" aria- describedby="emailHelp"

value="Submit">

</div>

</form>

<script type="text/javascript">

//validation

function validate\_registration(e) {

var reg\_email = /^([A-Za-z0-9\_\-\.]){1,}\@([A-Za-z0-9\_\-\.]){1,}\.([A-Za-z]{2,4})$/; var reg\_only\_char = /^([A-Za-z ]){5,}$/;

var reg\_only\_num = /^([0-9]){10}$/; var reg\_file = /(\.jpeg|\.jpg|\.png)$/i;

var f\_name = document.forms["reg\_form"]["firstname"].value; var l\_name = document.forms["reg\_form"]["lastname"].value; var u\_name = document.forms["reg\_form"]["username"].value; var email = document.forms["reg\_form"]["email"].value;

var password\_1 = document.forms["reg\_form"]["password"].value;

var password\_2 = document.forms["reg\_form"]["confirmpassword"].value; var mob = document.forms["reg\_form"]["phone"].value;

var type = document.forms["reg\_form"]["type\_usr"].value; var src=document.forms["reg\_form"]["profile\_img"].value; var str = "Select";

// error\_reg\_div/error\_reg\_para

if (reg\_only\_char.test(f\_name)==false) {

var msg = "INVALID NAME. PLEASE ENTER A VALID FIRST NAME";

document.getElementById("error\_reg\_div").style.visibility = 'visible'; document.getElementById("error\_reg\_para").innerHTML = msg;

// document.getElementById("error\_cp").innerHTML = msg; document.forms["reg\_form"]["firstname"].focus();

return false;

}

if (reg\_only\_char.test(l\_name) == false) {

var msg = "INVALID NAME. PLEASE ENTER A VALID SECOND NAME";

document.getElementById("error\_reg\_div").style.visibility = 'visible'; document.getElementById("error\_reg\_para").innerHTML = msg;

// document.getElementById("error\_cp").innerHTML = msg; document.forms["reg\_form"]["lastname"].focus();

return false;

}

if (reg\_only\_char.test(u\_name) == false) {

var msg = "CHARACTERS ARE EXPECTED! PLEASE ENTER A VALID USERNAME";

document.getElementById("error\_reg\_div").style.visibility = 'visible';

document.getElementById("error\_reg\_para").innerHTML = msg;

// document.getElementById("error\_email").innerHTML = msg; document.forms["reg\_form"]["username"].focus();

return false;

}

if (reg\_email.test(email) == false) {

var msg = "MAIL ID EXPECTED! PLEASE ENTER A VALID EMAIL";

document.getElementById("error\_reg\_div").style.visibility = 'visible'; document.getElementById("error\_reg\_para").innerHTML = msg;

// document.getElementById("error\_email").innerHTML = msg; document.forms["reg\_form"]["email"].focus();

return false;

}

if (password\_1 != password\_2) {

var msg = "PASSWORD DESNOT MATCH";

document.getElementById("error\_reg\_div").style.visibility = 'visible'; document.getElementById("error\_reg\_para").innerHTML = msg;

//document.forms["reg\_form"]["confirmpassword"].focus(); return false;

}

if (reg\_only\_num.test(mob) == false) {

var msg = "INVALID PHONE NUMBER. PLEASE ENTER A VALID PHONE NUMBER";

document.getElementById("error\_reg\_div").style.visibility = 'visible'; document.getElementById("error\_reg\_para").innerHTML = msg;

// document.getElementById("error\_phone").innerHTML = msg; document.forms["reg\_form"]["mobile"].focus();

return false;

}

if (type == str) {

var msg = "INVALID USER TYPE. PLEASE ENTER A VALID USER TYPE";

document.getElementById("error\_reg\_div").style.visibility = 'visible'; document.getElementById("error\_reg\_para").innerHTML = msg;

// document.getElementById("error\_type").innerHTML = msg; document.forms["reg\_form"]["event\_category"].focus();

return false;

}

if (reg\_file.test(src) == false) { msg = "INVALID FILE TYPE";

document.getElementById("error\_title\_div").style.visibility = 'visible'; document.getElementById("error\_para").innerHTML = msg;

// document.getElementById("error\_file\_type").innerHTML = msg; document.forms["reg\_form"]["profile\_img"].focus();

return false;

}

}

</script>

</div>

</div>

</div>

<!--Add buttons to initiate auth sequence and sign out-->

<pre id="content" style="white-space: pre-wrap;"></pre>

<script type="text/javascript">

// Client ID and API key from the Developer Console

var CLIENT\_ID = '117313785974-13346rmlvpfpqia7knlds1sbd5k7iqsi.apps.googleusercontent.com'; var API\_KEY = 'AIzaSyBZs4Ue6SwaAyAOeB3YnxdFvBQWcdnI\_Rc';

// Array of API discovery doc URLs for APIs used by the quickstart

var DISCOVERY\_DOCS = ["https[://www.](http://www.googleapis.com/discovery/v1/apis/gmail/v1/rest)goo[gleapis.com/discovery/v1/apis/gmail/v1/rest"]](http://www.googleapis.com/discovery/v1/apis/gmail/v1/rest);

// Authorization scopes required by the API; multiple scopes can be

// included, separated by spaces.

var SCOPES = 'https[://www.](http://www.googleapis.com/auth/gmail.readonly%27%3B)goo[gleapis.com/auth/gmail.readonly';](http://www.googleapis.com/auth/gmail.readonly%27%3B)

var authorizeButton = document.getElementById('authorize\_button'); var signoutButton = document.getElementById('signout\_button');

/\*\*

\* On load, called to load the auth2 library and API client library.

\*/

function handleClientLoad() { gapi.load('client:auth2', initClient);

}

/\*\*

* Initializes the API client library and sets up sign-in state
* listeners.

\*/

function initClient() { gapi.client.init({ apiKey: API\_KEY, clientId: CLIENT\_ID,

discoveryDocs: DISCOVERY\_DOCS, scope: SCOPES

}).then(function () {

// Listen for sign-in state changes. gapi.auth2.getAuthInstance().isSignedIn.listen(updateSigninStatus);

// Handle the initial sign-in state. updateSigninStatus(gapi.auth2.getAuthInstance().isSignedIn.get()); authorizeButton.onclick = handleAuthClick;

signoutButton.onclick = handleSignoutClick;

}, function (error) { appendPre(JSON.stringify(error, null, 2));

});

}

/\*\*

* Called when the signed in status changes, to update the UI
* appropriately. After a sign-in, the API is called.

\*/

function updateSigninStatus(isSignedIn) { if (isSignedIn) { authorizeButton.style.display = 'none'; signoutButton.style.display = 'block'; listLabels();

} else {

authorizeButton.style.display = 'block'; signoutButton.style.display = 'none';

}

}

/\*\*

* Sign in the user upon button click.

\*/

function handleAuthClick(event) { gapi.auth2.getAuthInstance().signIn();

}

/\*\*

* Sign out the user upon button click.

\*/

function handleSignoutClick(event) { gapi.auth2.getAuthInstance().signOut();

}

/\*\*

* Append a pre element to the body containing the given message
* as its text node. Used to display the results of the API call.

\*

* @param {string} message Text to be placed in pre element.

\*/

function appendPre(message) {

var pre = document.getElementById('content');

var textContent = document.createTextNode(message + '\n'); pre.appendChild(textContent);

}

/\*\*

* Print all Labels in the authorized user's inbox. If no labels
* are found an appropriate message is printed.

\*/

function listLabels() { gapi.client.gmail.users.labels.list({ 'userId': 'me'

}).then(function (response) {

var labels = response.result.labels; appendPre('Labels:');

if (labels && labels.length > 0) { for (i = 0; i < labels.length; i++) { var label = labels[i]; appendPre(label.name)

}

} else {

appendPre('No Labels found.');

}

});

}

</script>

<script async defer src="https://apis.google.com/js/api.js" onload="this.onload=function(){};handleClientLoad()" onreadystatechange="if (this.readyState === 'complete') this.onload()">

</script>

{% endblock %}

## ORGANISER

* + - 1. **Models.py**

from django.db import models

from django.contrib.auth.models import User from django.conf import settings

from django.utils.timezone import now from django.utils import timezone

# from froala\_editor.fields import FroalaField

from django.contrib.auth import get\_user\_model # Create your models here.

# from .models import OrganiseEvent

class OrganiseEvent(models.Model):

event\_title = models.CharField(max\_length=200) event\_description = models.CharField(max\_length=800) event\_category = models.CharField(max\_length=200) org\_name = models.CharField(max\_length=200) org\_email = models.EmailField(max\_length=100) org\_mobile = models.BigIntegerField() org\_contact\_person = models.CharField(max\_length=100) event\_poster = models.ImageField(

upload\_to='images/event\_poster/', default="images/noimage.png") event\_startdate = models.DateTimeField(default=now) event\_enddate = models.DateTimeField()

us = models.ForeignKey(User, on\_delete=models.CASCADE)

def summary(self):

return self.event\_description[:150]

class EventDetails(models.Model):

event = models.CharField(max\_length=200) expected\_participant = models.IntegerField() no\_participant = models.IntegerField() event\_level = models.CharField(max\_length=200) eligibility = models.CharField(max\_length=200)

prerequisite = models.TextField(max\_length=1500) facility = models.CharField(max\_length=100) event\_detail\_docs = models.FileField(

upload\_to='images/event\_details\_docs/')

us = models.ForeignKey(User, on\_delete=models.CASCADE)

org\_id = models.ForeignKey(OrganiseEvent, on\_delete=models.CASCADE)

class ShareResource(models.Model):

event\_title = models.CharField(max\_length=100) subject = models.CharField(max\_length=100) description = models.TextField(max\_length=1500) publishedDate = models.DateTimeField(default=now) resourceLink = models.CharField(max\_length=100)

documentFile = models.FileField(upload\_to='images/shared\_resources\_docs/') publisedBy = models.CharField(max\_length=100)

resourceImage = models.ImageField(upload\_to='images/shared\_resources/') us = models.ForeignKey(User, on\_delete=models.CASCADE)

org\_id = models.ForeignKey(OrganiseEvent, on\_delete=models.CASCADE)

class SponsorShip(models.Model):

event\_title = models.CharField(max\_length=100, default=True) platinum\_sponsor = models.CharField(max\_length=100) f\_platinum = models.TextField(max\_length=1500) ex\_platinum = models.IntegerField()

gold\_sponsor = models.CharField(max\_length=100) f\_gold = models.TextField(max\_length=1500) ex\_gold = models.IntegerField()

silver\_sponsor = models.CharField(max\_length=100) f\_silver = models.TextField(max\_length=1500) ex\_silver = models.IntegerField()

us = models.ForeignKey(User, on\_delete=models.CASCADE)

org\_id = models.ForeignKey(OrganiseEvent, on\_delete=models.CASCADE)

class Event\_Location(models.Model):

event\_venue\_name = models.CharField(max\_length=200) event\_venue\_addr = models.CharField(max\_length=300) event\_latitude = models.CharField(max\_length=100) event\_longitude = models.CharField(max\_length=100)

eventid = models.ForeignKey(OrganiseEvent, on\_delete=models.CASCADE) event\_name = models.CharField(max\_length=200)

## Urls.py

from django.urls import path from django.conf import settings

from django.conf.urls.static import static from Organizer import views

urlpatterns = [

path('', views.indx, name='organiserIndex'),

path('organiseEvent', views.organiseEvent, name='organiseEvent'), path('formpage/', views.form\_name\_view, name='form\_name'), path('organiseEventFormData', views.organiseEventFormData,

name="organiseEventFormData"),

path('registeredEvent', views.registeredEvent, name='registeredevent'), path('eventRegistered/<int:id>', views.eventEdit, name='eventEdit'), path('eventUpdate/<int:id>', views.eventUpdate, name='eventupdate'), path('evenDetailsdelete/<int:id>',

views.evenDetailsdelete, name='evenDetailsdelete'), path('addEventDetails', views.addrubrics, name='addEventDetails'), path('editRubrics/<int:id', views.returnEditRubrics, name='editRubrics'), path('evenDetailsUpdate/<int:id>',

views.evenDetailsUpdate, name='evenDetailsUpdate'), path('addEvent', views.eventDetails, name='eventdetails'), path('shareResourcese', views.shareResource, name='shareResource'), path('shareResource', views.resource, name='Resource'), path('edit\_resource', views.editResource, name='editResource'), path('updateShareResources/<int:id>',

views.updateShareResources, name='update\_shareResources'), path('blog', views.blog, name='blog'),

path('blogsDetail', views.blogsDetails, name='blogsDetails'), path('WriteBlog', views.writeBlogs, name='WriteBlog'), path('sponsorShip', views.sponsorShip, name='sponsorShip'), path('sponsorShipDetails', views.sponsorShipDetails,

name='sponsorShipDetails'),

path('editSponsorShip', views.editSponsorShip, name='editSponsorShip'), path('updateSponsorInfo/<int:id>',

views.updateSponsorInfo, name='updateSponsorInfo'), path('deletesponsor/<int:id>', views.deletesponsor, name='deletesponsor'), path('deleteeventdetails/<int:id>',

views.deleteeventdetails, name='deleteeventdetails'), path('deleteresourcedetails/<int:id>',

views.deleteresourcedetails, name='deleteresourcedetails'), path('EventLocation', views.EventLocation, name='EventLocation'), path('location\_view', views.Location\_view, name='view\_locatiion'), path('load\_map', views.load\_map, name='load\_map'),

path('EventLocationload', views.EventLocationload, name='EventLocationload'), path('locationUpdate/<int:id>', views.locationupdate, name='locationupdate'), path('locationdelete/<int:id>', views.locationdelete, name='locationdelete'), path('profile\_org', views.profile\_org, name='profile\_org'),

path('editProfile', views.editProfile, name='editProfile'), path('req\_pdf', views.req\_pdf, name="req\_pdf"), path('updateProfile', views.updateProfile, name='update\_profile'), path('del\_profile/<int:id>', views.del\_profile, name='del\_profile'), path('participants\_event/<int:id>/',

views.participants\_event, name='participants\_event'), path('sponsors\_for\_event/<int:id>/',

views.sponser\_for\_event, name='sponser\_for\_event'), path('view\_org\_blog', views.view\_org\_blog, name='view\_org\_blog')

] + static(settings.MEDIA\_URL, document\_root=settings.MEDIA\_ROOT)

## Views.py

from django.shortcuts import render, redirect, get\_object\_or\_404 from django.http import HttpResponse

from . import forms

from .models import OrganiseEvent, EventDetails, ShareResource, SponsorShip, Event\_Location from django.contrib.auth.models import User

from accounts.models import UserProfile from Blog.models import BlogsInfo from django.utils import timezone import io

from django.http import FileResponse from reportlab.pdfgen import canvas from reportlab.lib.pagesizes import A4

from django.http import HttpResponse,Http404 from wsgiref.util import FileWrapper

import os

from django.conf import settings

from Participant.models import ParticipateEvent from Sponsor.models import SponsorTransaction

def participants\_event(request,id):

event = OrganiseEvent.objects.get(pk=id) try:

participants = ParticipateEvent.objects.get(Event\_id\_id=id) usr = User.objects.get(pk=participants.us\_id)

otherdetails\_usr = UserProfile.objects.get(id=participants.us\_id)

return render(request, 'participants\_event.html', {'event': event, 'part': participants, 'usr': usr, 'otherdetails\_usr': otherdetails\_usr})

except:

return render(request, 'participants\_event.html', {'event': event})

def sponser\_for\_event(request, id):

event = OrganiseEvent.objects.get(pk=id)

sponsor\_id = SponsorTransaction.objects.get(org\_id\_id=id) sponsor = User.objects.get(pk=sponsor\_id.us\_id) otherdetails\_usr = UserProfile.objects.get(id=sponsor\_id.us\_id) try:

return render(request, 'sponsor\_for\_event.html', {'otherdetails\_usr': otherdetails\_usr,'sponsor': sponsor, 'event': event})

except:

return render(request, 'sponsor\_for\_event.html',{'event': event})

def EventLocation(request):

if request.method == 'POST':

event\_name = request.POST['event\_name'] event\_venue\_name = request.POST['venue\_name'] event\_venue\_addr = request.POST['venue\_addr'] event\_latitude = request.POST['event\_latitude'] event\_longitude = request.POST['event\_longitude']

if event\_venue\_name == '':

return render(request, 'eventlocationinfo.html', {'error': "Event Venue is missing"}) else:

event\_venue\_name = request.POST['venue\_name'] if event\_venue\_addr == '':

return render(request, 'eventlocationinfo.html', {'error': "Event Address is missing"}) else:

event\_venue\_addr = request.POST['venue\_addr'] if event\_latitude == '':

return render(request, 'eventlocationinfo.html', {'error': "Event Latitude is missing"}) else:

event\_latitude = request.POST['event\_latitude'] if event\_longitude == '':

return render(request, 'eventlocationinfo.html', {'error': "Event longitude is missing"}) else:

event\_longitude = request.POST['event\_longitude'] if event\_name == '':

return render(request, 'eventlocationinfo.html', {'error': "Event Name is missing"}) else:

event\_name = request.POST['event\_name']

eid = OrganiseEvent.objects.get(event\_title=event\_name)

location\_data = Event\_Location(event\_venue\_name=event\_venue\_name, event\_venue\_addr=event\_venue\_addr,

event\_latitude=event\_latitude, event\_longitude=event\_longitude, eventid\_id=eid.id, event\_name=event\_name)

location\_data.save()

return render(request, 'organiser\_index.html') else:

return render(request, 'eventlocationinfo.html')

def EventLocationload(request):

event = OrganiseEvent.objects.filter(us=request.user.id)

return render(request, 'eventlocationinfo.html', {'event': event})

def Location\_view(request):

event = OrganiseEvent.objects.filter(us=request.user.id)

return render(request, 'locationinfo\_view.html', {'event': event})

def load\_map(request):

if request.method == 'POST':

event\_name = request.POST['event\_name']

event = OrganiseEvent.objects.filter(us=request.user.id)

loc = Event\_Location.objects.filter(event\_name=event\_name)

return render(request, 'locationinfo\_view.html', {'event': event, 'location': loc}) else:

event = OrganiseEvent.objects.filter(us=request.user.id)

return render(request, 'locationinfo\_view.html', {'event': event})

def locationupdate(request,id):

# event = OrganiseEvent.objects.filter(us=request.user.id)

# return render(request, 'locationinfo\_view.html', {'event': event}) if request.method == 'POST':

evnt=Event\_Location.objects.get(pk=id) event\_venue\_name = request.POST['event\_venue\_name'] event\_venue\_addr = request.POST['event\_venue\_addr'] event\_latitude = request.POST['event\_latitude'] event\_longitude = request.POST['event\_longitude'] event\_name = request.POST['event\_name']

submitbutton = request.POST.get('Submit') if submitbutton:

eid = OrganiseEvent.objects.get(event\_title=event\_name)

location\_data = Event\_Location(id=id,event\_venue\_name=event\_venue\_name, event\_venue\_addr=event\_venue\_addr,

event\_latitude=event\_latitude, event\_longitude=event\_longitude, eventid\_id=eid.id, event\_name=event\_name)

location\_data.save()

event = OrganiseEvent.objects.filter(us=request.user.id)

return render(request, 'locationinfo\_view.html', {'event': event,'error': "Successfully edited the details"})

else:

event = OrganiseEvent.objects.filter(us=request.user.id)

return render(request, 'locationinfo\_view.html', {'event': event})

else:

event = OrganiseEvent.objects.filter(us=request.user.id)

return render(request, 'locationinfo\_view.html', {'event': event})

def locationdelete(request,id):

event = Event\_Location.objects.get(pk=id) event.delete()

event = OrganiseEvent.objects.filter(us=request.user.id)

return render(request, 'locationinfo\_view.html', {'event': event,'error': "Successfully removed the details"})

def editSponsorShip(request):

sponsorShip = SponsorShip.objects.filter(us=request.user.id)

return render(request, 'edit\_sponsorship.html', {'sponsorShip': sponsorShip})

def deletesponsor(request, id):

sponsorShip = SponsorShip.objects.get(id=id) sponsorShip.delete()

sponsorShip = SponsorShip.objects.filter(us=request.user.id)

return render(request, 'edit\_sponsorship.html', {'sponsorShip': sponsorShip})

def deleteeventdetails(request, id):

events = OrganiseEvent.objects.get(id=id)

loc = Event\_Location.objects.filter(eventid\_id=request.user.id) events.delete()

loc.delete()

events = OrganiseEvent.objects.filter(us=request.user.id)

loc = Event\_Location.objects.filter(eventid\_id=request.user.id) context = {'events': events, 'loc': loc}

return render(request, 'registered\_event.html', context)

def evenDetailsdelete(request,id): if request.method == 'POST':

events = EventDetails.objects.get(pk=id) events.delete()

eventDetails = EventDetails.objects.filter(us=request.user.id) eventsOrganise = OrganiseEvent.objects.filter(us=request.user.id) context = { 'eventDetails': eventDetails,

'eventsOrganise': eventsOrganise, 'registered': 'Event Details Removed!'

}

return render(request, 'addrubrics.html', context)

def deleteresourcedetails(request,id): resource=ShareResource.objects.get(id=id) resource.delete()

share\_resource = ShareResource.objects.filter(us=request.user.id)

return render(request, 'edit\_shareresource.html', {'share\_resource': share\_resource}) def updateSponsorInfo(request,id):

spon = SponsorShip.objects.get(pk=id) submitbutton = request.POST.get('Submit') if submitbutton:

sponsor = SponsorShip.objects.get(pk=id) event\_title = request.POST['event\_title']

platinum\_sponsor = request.POST['platinum\_sponsor'] f\_platinum = request.POST['f\_platinum']

ex\_platinum = request.POST['ex\_platinum'] gold\_sponsor = request.POST['gold\_sponsor'] f\_gold = request.POST['f\_gold']

ex\_gold = request.POST['ex\_gold'] silver\_sponsor = request.POST['silver\_sponsor'] f\_silver = request.POST['f\_silver']

ex\_silver = request.POST['ex\_silver']

userr = User.objects.get(pk= request.user.id)

orgid = OrganiseEvent.objects.get(id=spon.org\_id.id)

sponsor\_ship = SponsorShip(id=id, event\_title=event\_title, platinum\_sponsor=platinum\_sponsor, f\_platinum=f\_platinum, ex\_platinum=ex\_platinum, gold\_sponsor=gold\_sponsor,

f\_gold=f\_gold, ex\_gold=ex\_gold, silver\_sponsor=silver\_sponsor, f\_silver=f\_silver, ex\_silver=ex\_silver, us=userr,org\_id=orgid)

sponsor\_ship.save()

sponsorShip = SponsorShip.objects.filter(us=request.user.id)

return render(request, 'edit\_sponsorship.html', {'sponsorShip': sponsorShip}) else:

if platinum\_sponsor == '':

return render(request, 'edit\_sponsorship.html', {'error': "Title is not given"}) else:

platinum\_sponsor = request.POST['platinum\_sponsor'] if f\_platinum == '':

return render(request, 'edit\_sponsorship.html', {'error': "Title is not given"}) else:

f\_platinum == request.POST['f\_platinum'] if ex\_platinum == '':

return render(request, 'edit\_sponsorship.html', {'error': "Title is not given"}) else:

ex\_platinum = request.POST['ex\_platinum'] if gold\_sponsor == '':

return render(request, 'edit\_sponsorship.html', {'error': "Title is not given"}) else:

gold\_sponsor = request.POST['gold\_sponsor'] if f\_gold == '':

return render(request, 'edit\_sponsorship.html', {'error': "Title is not given"}) else:

f\_gold = request.POST['f\_gold'] if ex\_gold == '':

return render(request, 'edit\_sponsorship.html', {'error': "Title is not given"}) else:

ex\_gold = request.POST['ex\_gold'] if silver\_sponsor == '':

return render(request, 'edit\_sponsorship.html', {'error': "Title is not given"}) else:

silver\_sponsor = request.POST['silver\_sponsor'] if f\_silver == '':

return render(request, 'edit\_sponsorship.html', {'error': "Title is not given"}) else:

f\_silver = request.POST['f\_silver'] if ex\_silver == '':

return render(request, 'edit\_sponsorship.html', {'error': "Title is not given"}) else:

ex\_silver = request.POST['ex\_silver']

def sponsorShip(request):

event = OrganiseEvent.objects.filter(us\_id=request.user.id) return render(request, 'sponsorShip.html', { 'event': event})

def sponsorShipDetails(request): if request.method == 'POST':

eventDetail = EventDetails.objects.filter(us=request.user.id) event = request.POST['event\_title']

event\_title = request.POST['event\_title'] platinum\_sponsor = request.POST['platinum\_sponsor'] f\_platinum = request.POST['f\_platinum']

ex\_platinum = request.POST['ex\_platinum'] gold\_sponsor = request.POST['gold\_sponsor'] f\_gold = request.POST['f\_gold']

ex\_gold = request.POST['ex\_gold'] silver\_sponsor = request.POST['silver\_sponsor'] f\_silver = request.POST['f\_silver']

ex\_silver = request.POST['ex\_silver']

if platinum\_sponsor == '':

return render(request, 'shareresource.html', {'error': "Title is not given"}) else:

platinum\_sponsor = request.POST['platinum\_sponsor'] if f\_platinum == '':

return render(request, 'shareresource.html', {'error': "Title is not given"}) else:

f\_platinum == request.POST['f\_platinum']

if ex\_platinum == '':

return render(request, 'shareresource.html', {'error': "Title is not given"}) else:

ex\_platinum = request.POST['ex\_platinum'] if gold\_sponsor == '':

return render(request, 'shareresource.html', {'error': "Title is not given"}) else:

gold\_sponsor = request.POST['gold\_sponsor'] if f\_gold == '':

return render(request, 'shareresource.html', {'error': "Title is not given"}) else:

f\_gold = request.POST['f\_gold'] if ex\_gold == '':

return render(request, 'shareresource.html', {'error': "Title is not given"}) else:

ex\_gold = request.POST['ex\_gold'] if silver\_sponsor == '':

return render(request, 'shareresource.html', {'error': "Title is not given"}) else:

silver\_sponsor = request.POST['silver\_sponsor'] if f\_silver == '':

return render(request, 'shareresource.html', {'error': "Title is not given"}) else:

f\_silver = request.POST['f\_silver'] if ex\_silver == '':

return render(request, 'shareresource.html', {'error': "Title is not given"}) else:

ex\_silver = request.POST['ex\_silver']

orgid = OrganiseEvent.objects.get(event\_title=event\_title)

sponsor\_ship = SponsorShip(event\_title=event, platinum\_sponsor=platinum\_sponsor, f\_platinum=f\_platinum, ex\_platinum=ex\_platinum, gold\_sponsor=gold\_sponsor,

f\_gold=f\_gold, ex\_gold=ex\_gold, silver\_sponsor=silver\_sponsor, f\_silver=f\_silver, ex\_silver=ex\_silver, us\_id=request.user.id, org\_id=orgid)

sponsor\_ship.save()

event = OrganiseEvent.objects.filter(us\_id=request.user.id) return render(request, 'sponsorShip.html', {'event': event})

def blogsDetails(request):

blogsInfo = BlogsInfo.objects.filter(us=request.user.id)

return render(request, 'blog\_edit.html', {'blogsInfo': blogsInfo})

def blog(request):

blogsInfo = BlogsInfo.objects.filter(us=request.user.id)

return render(request, 'blog\_write.html', {'blogsInfo': blogsInfo})

def view\_org\_blog(request):

blogsInfo = BlogsInfo.objects.filter(us=request.user.id)

return render(request, 'view\_org\_blog.html', {'blogsInfo': blogsInfo})

def writeBlogs(request):

if request.method == 'POST':

title = request.POST['blog\_title'] pubDateTime = timezone.now() description = request.POST['description']

imageSecond = request.POST['image\_second']

imageFirst = request.POST['image\_first'] blogCategory = request.POST['blog\_category'] refrenceLinks = request.POST['refrence\_link'] UserType = 'Organiser'

if request.user.is\_authenticated:

authorName = request.user.first\_name + " " + request.user.last\_name if title == '':

return render(request, 'shareresource.html', {'error': "Title is not given"}) else:

title = request.POST['blog\_title'] if pubDateTime == '':

pubDateTime = timezone.now() if description == '':

return render(request, 'shareresource.html', {'error': "Description is not written"}) else:

description = request.POST['description'] if imageFirst == '':

return render(request, 'shareresource.html', {'error': "Image first is not given"}) else:

imageFirst = request.POST['image\_first'] if imageSecond == '':

return render(request, 'shareresource.html', {'error': "Image Two is not given"}) else:

imageSecond = request.POST['image\_second'] if blogCategory == '':

return render(request, 'shareresource.html', {'error': "Blog Category is not selected"}) else:

blogCategory = request.POST['blog\_category'] if refrenceLinks == '':

return render(request, 'shareresource.html', {'error': "Refrence links not given"}) else:

refrenceLinks = request.POST['refrence\_link']

blogsInfo = BlogsInfo(title=title, pubDateTime=pubDateTime, description=description, imageSecond=imageSecond, imageFirst=imageFirst,

UserType=UserType, authorName=authorName, blogCategory=blogCategory, refrenceLinks=refrenceLinks, us\_id=request.user.id)

blogsInfo.save()

print("author name:" + authorName)

return render(request, 'blog\_write.html', {'error': "Successfully added the blog!"})

def shareResource(request):

event = OrganiseEvent.objects.filter(us\_id=request.user.id) return render(request, 'shareresource.html', {'event': event})

def editResource(request):

share\_resource = ShareResource.objects.filter(us=request.user.id)

return render(request, 'edit\_shareresource.html', {'share\_resource': share\_resource})

def updateShareResources(request, id): share = ShareResource.objects.get(pk=id) if request.method == 'POST':

submitbutton = request.POST.get('Submit') if submitbutton:

share\_resource = ShareResource.objects.get(pk=id) event\_title = request.POST['event\_name']

subject = request.POST['subject']

description = request.POST['description'] publishedDate = timezone.now() resourceLink = request.POST['addlinks']

documentFile = request.POST['document\_file'] publisedBy = request.POST['published\_by'] resourceImage = request.POST['share\_image'] if documentFile == '':

resourceImage = share\_resource.resourceImage else:

resourceImage = request.POST['share\_image'] if resourceImage == '':

documentFile = share\_resource.documentFile else:

documentFile = request.POST['document\_file']

userr = User.objects.get(pk= request.user.id)

orgid = OrganiseEvent.objects.get(id=share.org\_id.id) # print("org id",orgid)

share\_resource = ShareResource(id=share\_resource.id, event\_title=event\_title, subject=subject, description=description, publishedDate=publishedDate,

resourceLink=resourceLink, documentFile=documentFile, publisedBy=publisedBy, resourceImage=resourceImage, us=userr,org\_id=orgid )

share\_resource.save() share\_resource.refresh\_from\_db() share\_resource = ShareResource.objects.filter(

us\_id=request.user.id)

return render(request, 'edit\_shareresource.html', {'share\_resource': share\_resource})

def resource(request):

if request.method == 'POST':

event\_title = request.POST['event\_name'] subject = request.POST['subject'] description = request.POST['descr']

publishedDate = request.POST['published\_date'] resourceLink = request.POST['addlinks'] documentFile = request.POST['document\_file'] publisedBy = request.POST['published\_by'] resourceImage = request.POST['share\_image']

if event\_title == '':

return render(request, 'shareresource.html', {'error': "Event is not selected"}) else:

event\_title = request.POST['event\_name'] if subject == '':

return render(request, 'shareresource.html', {'error': "Event is not selected"}) else:

subject = request.POST['subject'] if description == '':

return render(request, 'shareresource.html', {'error': "Event is not selected"}) else:

description = request.POST['descr'] if publishedDate == '':

return render(request, 'shareresource.html', {'error': "Event is not selected"}) else:

publishedDate = request.POST['published\_date'] if resourceLink == '':

return render(request, 'shareresource.html', {'error': "Event is not selected"})

else:

resourceLink = request.POST['addlinks'] if documentFile == '':

return render(request, 'shareresource.html', {'error': "Event is not selected"}) else:

documentFile = request.POST['document\_file'] if publisedBy == '':

return render(request, 'shareresource.html', {'error': "Event is not selected"}) else:

publisedBy = request.POST['published\_by'] if resourceImage == '':

return render(request, 'shareresource.html', {'error': "Event is not selected"}) else:

resourceImage = request.POST['share\_image']

orgid = OrganiseEvent.objects.get(event\_title=event\_title)

share\_resource = ShareResource(event\_title=event\_title, subject=subject, description=description, publishedDate=publishedDate, resourceLink=resourceLink,

documentFile=documentFile, publisedBy=publisedBy, resourceImage=resourceImage, us\_id=request.user.id, org\_id=orgid)

share\_resource.save()

share\_resource = ShareResource.objects.filter(us\_id=request.user.id)

return render(request, 'shareresource.html', {'share\_resource': share\_resource})

def evenDetailsUpdate(request, id): if request.method == 'POST':

events = EventDetails.objects.get(pk=id) event = request.POST['event\_title']

no\_participant = request.POST['no\_participant'] expected\_participant = request.POST['expected\_participant'] event\_level = request.POST['event\_level']

eligibility = request.POST['eligibility'] prerequisite = request.POST['prerequisite'] facility = request.POST['facilities']

event\_detail\_docs = request.POST['event\_detail\_docs'] submitbutton = request.POST.get('Submit')

if submitbutton:

if event\_detail\_docs == '':

event\_detail\_docs = events.event\_detail\_docs userr = User.objects.get(pk= request.user.id) # print("orggg id",events.org\_id.id)

# print("ehj",events.id)

orgid = OrganiseEvent.objects.get(id=events.org\_id.id) # print("org id",orgid)

eventInstance = EventDetails(id=id, event=event, no\_participant=no\_participant, expected\_participant=expected\_participant, event\_level=event\_level,

eligibility=eligibility, prerequisite=prerequisite, facility=facility, event\_detail\_docs=event\_detail\_docs, us=userr,org\_id= orgid)

eventInstance.save()

return render(request, 'addrubrics.html', {'registered': 'Event Successfully Registered!'})

else:

if event == '':

return render(request, 'update\_rubrics.html', {'registered': 'Event Successfully Registered!'}) else:

event = request.POST['event\_name'] if no\_participant == '':

return render(request, 'update\_rubrics.html', {'registered': 'Event Successfully Registered!'}) else:

no\_participant = request.POST['no\_participant'] if expected\_participant == '':

return render(request, 'update\_rubrics.html', {'registered': 'Event Successfully Registered!'}) else:

expected\_participant = request.POST['expected\_participant'] if event\_level == '':

return render(request, 'update\_rubrics.html', {'registered': 'Event Successfully Registered!'}) else:

event\_level = request.POST['event\_level']

if eligibility == '':

return render(request, 'update\_rubrics.html', {'registered': 'Event Successfully Registered!'}) else:

eligibility = request.POST['eligibility'] if prerequisite == '':

return render(request, 'update\_rubrics.html', {'registered': 'Event Successfully Registered!'}) else:

prerequisite = request.POST['prerequisite'] if facility == '':

return render(request, 'update\_rubrics.html', {'registered': 'Event Successfully Registered!'}) else:

facility = request.POST['facilities'] if event\_detail\_docs == '':

return render(request, 'update\_rubrics.html', {'registered': 'Event Successfully Registered!'}) else:

event\_detail\_docs = request.POST['event\_detail\_docs']

def returnEditRubrics(request):

events = EventDetails.objects.filter(us=request.user.id)

return render(request, 'update\_rubrics.html', {'events': events})

def eventUpdate(request, id):

if request.method == 'POST':

submitbutton = request.POST.get('Submit') if submitbutton:

eventid = OrganiseEvent.objects.get(pk=id) # print('get the id ', eventid.id)

userl = User.objects.get(pk=request.user.id) # print("user id is ",userl)

event\_title = request.POST['event\_title'] event\_description = request.POST['event\_description'] event\_category = request.POST['event\_category'] org\_name = request.POST['org\_by']

org\_email = request.POST['org\_email'] org\_mobile = request.POST['mobile']

org\_contact\_person = request.POST['person\_name'] event\_poster = request.POST['event\_poster'] event\_startdate = request.POST['event\_startdate'] event\_enddate = request.POST['event\_enddate']

if event\_startdate == '': event\_startdate=eventid.event\_startdate

else:

event\_startdate = request.POST['event\_startdate'] if event\_enddate == '':

event\_enddate =eventid.event\_enddate else:

event\_enddate = request.POST['event\_enddate'] if event\_poster == '':

event\_poster = eventid.event\_poster else:

event\_poster = request.POST['event\_poster'] if event\_title == '':

return render(request, 'registered\_event.html', {'error': "Please Enter Event Title!"}) else:

event\_title = request.POST['event\_title'] if event\_category == '':

return render(request, 'registered\_event.html', {'error': "Please Select Event Category!"}) else:

event\_category = request.POST['event\_category'] if org\_name == '':

return render(request, 'registered\_event.html', {'error': "Please Enter Org Name!"}) else:

org\_name = request.POST['org\_by'] if org\_email == '':

return render(request, 'registered\_event.html', {'error': "Please Enter Org Email!"}) else:

org\_email = request.POST['org\_email'] if org\_mobile == '':

return render(request, 'registered\_event.html', {'error': "Please Enter Org Mobile!"}) else:

org\_mobile = request.POST['mobile'] if org\_contact\_person == '':

return render(request, 'registered\_event.html', {'error': "Please Enter Contact Person

Name!"})

else:

org\_contact\_person = request.POST['person\_name']

sve=OrganiseEvent(id=eventid.id,event\_title=event\_title,event\_category= event\_category,org\_name=org\_name,org\_email=org\_email,org\_mobile=org\_mobile,org\_contact\_perso n=org\_contact\_person,event\_poster=event\_poster,event\_startdate=event\_startdate,event\_enddate=even t\_enddate,us=userl,event\_description=event\_description)

sve.save()

event = OrganiseEvent.objects.filter(us=request.user.id)

context = {'events': event,'updated':'Successfully updated event '} return render(request, 'registered\_event.html',context)

else:

events = OrganiseEvent.objects.filter(us=request.user.id) context = {'events': events}

return render(request, 'registered\_event.html', context)

def eventDetails(request):

if request.method == 'POST':

event = request.POST['event\_name'] no\_participant = request.POST['no\_participant']

expected\_participant = request.POST['expected\_participant'] event\_level = request.POST['event\_level']

eligibility = request.POST['eligibility'] prerequisite = request.POST['prerequisite'] facility = request.POST['facilities']

event\_detail\_docs = request.POST['event\_detail\_docs']

if event == '':

return render(request, 'addrubrics.html', {'error': "Event is not selected"}) else:

event = request.POST['event\_name'] if no\_participant == '':

return render(request, 'addrubrics.html', {'error': "No Participant not filled!"}) else:

no\_participant = request.POST['no\_participant'] if expected\_participant == '':

return render(request, 'addrubrics.html', {'error': "No Participant not filled!"}) else:

expected\_participant = request.POST['expected\_participant'] if event\_level == '':

return render(request, 'addrubrics.html', {'error': "Event level field not selected"}) else:

event\_level = request.POST['event\_level'] if eligibility == '':

return render(request, 'addrubrics.html', {'error': "Eligibility field not selected"}) else:

eligibility = request.POST['eligibility'] if prerequisite == '':

return render(request, 'addrubrics.html', {'error': "Prerequisite field not filled"}) else:

prerequisite = request.POST['prerequisite'] if facility == '':

return render(request, 'addrubrics.html', {'error': "Facility field not filled"}) else:

facility = request.POST['facilities'] if event\_detail\_docs == '':

return render(request, 'addrubrics.html', {'error': "File is not attached"}) else:

event\_detail\_docs = request.POST['event\_detail\_docs'] orgid = OrganiseEvent.objects.get(event\_title=event)

eventInstance = EventDetails(event=event, no\_participant=no\_participant, expected\_participant=expected\_participant, event\_level=event\_level, eligibility=eligibility, prerequisite=prerequisite,

facility=facility, event\_detail\_docs=event\_detail\_docs, us\_id=request.user.id,

org\_id=orgid)

eventInstance.save()

eventDetails = EventDetails.objects.filter(us=request.user.id) eventsOrganise = OrganiseEvent.objects.filter(us=request.user.id) context = { 'eventDetails': eventDetails,

'eventsOrganise': eventsOrganise, 'registered': 'Event Successfully Registered!'

}

return render(request, 'addrubrics.html', context)

def organiseEventFormData(request): if request.method == 'POST':

event\_title = request.POST['event\_title'] event\_description = request.POST['event\_description'] event\_category = request.POST['event\_category'] org\_name = request.POST['org\_by']

org\_email = request.POST['org\_email'] org\_mobile = request.POST['mobile']

org\_contact\_person = request.POST['person\_name']

event\_poster = request.POST['event\_poster'] event\_startdate = request.POST['event\_startdate'] event\_enddate = request.POST['event\_enddate']

if OrganiseEvent.objects.filter(event\_title=event\_title).exists():

return render(request, 'organise\_event.html', {'error': "Event already Registered!"}) else:

if event\_title == '':

return render(request, 'organise\_event.html', {'error': "Please Enter Event Title!"}) else:

event\_title = request.POST['event\_title']

if event\_description == '':

return render(request, 'organise\_event.html', {'error': "Please Enter Event Title!"}) else:

event\_description = request.POST['event\_description']

if event\_category == '':

return render(request, 'organise\_event.html', {'error': "Please Select Event Category!"}) else:

event\_category = request.POST['event\_category']

if org\_name == '':

return render(request, 'organise\_event.html', {'error': "Please Enter Org Name!"}) else:

org\_name = request.POST['org\_by'] if org\_email == '':

return render(request, 'organise\_event.html', {'error': "Please Enter Org Email!"}) else:

org\_email = request.POST['org\_email'] if org\_mobile == '':

return render(request, 'organise\_event.html', {'error': "Please Enter Org Mobile!"}) else:

org\_mobile = request.POST['mobile'] if org\_contact\_person == '':

return render(request, 'organise\_event.html', {'error': "Please Enter Contact Person Name!"}) else:

org\_contact\_person = request.POST['person\_name'] if event\_poster == '':

return render(request, 'organise\_event.html', {'error': "Please Select Event Poster!"}) else:

event\_poster = request.POST['event\_poster'] if event\_startdate == '':

return render(request, 'organise\_event.html', {'error': "Please Select Start Date!"}) else:

event\_startdate = request.POST['event\_startdate'] if event\_enddate == '':

return render(request, 'organise\_event.html', {'error': "Please Select End Date !"}) else:

event\_enddate = request.POST['event\_enddate'] userid = User.objects.get(pk=request.user.id)

event = OrganiseEvent(event\_title=event\_title, event\_description=event\_description, event\_category=event\_category, org\_name=org\_name,

org\_email=org\_email, org\_mobile=org\_mobile, org\_contact\_person=org\_contact\_person,

event\_poster=event\_poster, event\_startdate=event\_startdate, event\_enddate=event\_enddate, us=userid)

event.save()

# return render(request,'organise\_event.html',{'event':'Event Successfully Registered!'}) events = OrganiseEvent.objects.filter(us=request.user.id)

loc = Event\_Location.objects.filter(eventid\_id=request.user.id) context = {'events': events, 'loc': loc}

return render(request, 'registered\_event.html', context)

def addrubrics(request):

eventDetails = EventDetails.objects.filter(us=request.user.id) eventsOrganise = OrganiseEvent.objects.filter(us=request.user.id) context = { 'eventDetails': eventDetails,

'eventsOrganise': eventsOrganise

}

return render(request, 'addrubrics.html', context)

def organiseEvent(request):

events = OrganiseEvent.objects.filter(us=request.user.id) return render(request, 'organise\_event.html', {'events': events})

def eventEdit(request, id):

event = get\_object\_or\_404(OrganiseEvent, pk=id) #event =get\_object\_or\_404(EventDetails,pk=id)

return render(request, 'sponsor\_event.html', {'event': event})

def registeredEvent(request):

events = OrganiseEvent.objects.filter(us=request.user.id)

loc = Event\_Location.objects.filter(eventid\_id=request.user.id) context = {'events': events, 'loc': loc}

return render(request, 'registered\_event.html', context)

def indx(request):

return render(request, 'organiser\_index.html')

def form\_name\_view(request): form = forms.FormName()

if request.method == 'POST':

form = forms.FormName(request.POST) form.save()

if form.is\_valid(): print("Validation Success!!")

print("name :" + form.cleaned\_data['name']) print("Email: " + form.cleaned\_data['email']) print("Text:" + form.cleaned\_data['text'])

return render(request, 'organiser.html', {'form': form})

def about(request):

return render(request, 'pages/index.html')

def profile\_org(request): profile=UserProfile.objects.filter(id=request.user.id) return render(request, 'profile\_org.html', {'profil':profile})

def editProfile(request): profiles = UserProfile.objects

return render(request, 'editProfile.html', {'profiles': profiles})

def about(request):

return render(request, 'pages/index.html')

def updateProfile(request):

if request.method == 'POST':

submitbutton = request.POST.get('Submit') if submitbutton:

user = request.POST['user'] fname = request.POST['fname'] lname = request.POST['lname'] email = request.POST['email']

profiles = User(id=request.user.id, username=user, first\_name=fname, last\_name=lname, email=email)

profiles.save() profiles.refresh\_from\_db() edit\_profile = UserProfile.objects

return render(request, 'profile.html', {'edit\_profile': edit\_profile})

def del\_profile(request,id): profile=User.objects.filter(id=id) profile.delete()

return render(request, 'login.html',{'error':"CREATE A NEW PROFILE NOW"})

def req\_pdf(request):

response = HttpResponse(content\_type='application/pdf')

# response['Content-Disposition'] = 'as\_attachment=True; filename="mypdf.pdf"' response['Content-Disposition'] = 'inline'

filename = "mypdf.pdf"

buffer = io.BytesIO() buffer.pagesize = A4

p = canvas.Canvas(buffer) p.setLineWidth(.3) p.setFont('Helvetica', 12)

p.line(80, 770, 480, 770)

p.drawString(80, 755, 'TECHEVENT USER DETAILS :') p.line(80, 750, 480, 750)

# Start writing the PDF here p.drawString(80, 725, 'Email ') p.drawString(200, 725, ': ') p.drawString(250, 725, request.user.email)

p.line(80, 700, 480, 700)

p.drawString(80, 675, 'User Name ') p.drawString(200, 675, ': ')

p.drawString(250, 675, request.user.username) p.drawString(80, 650, 'First Name ') p.drawString(200, 650, ': ')

p.drawString(250, 650, request.user.first\_name) p.drawString(80, 625, 'Last Name ') p.drawString(200, 625, ': ')

p.drawString(250, 625, request.user.last\_name)

p.line(80, 550, 480, 550)

p.drawString(

80, 532, 'DATA RETRIVED FROM DATA BASE IF ERROR CONTACT ADMIN') p.line(80, 525, 480, 525)

# End writing

p.showPage() p.save()

pdf = buffer.getvalue() buffer.close() response.write(pdf)

return response

## SPONSOR

* + - 1. **Models.py**

from django.db import models # Create your models here.

from Organizer.models import OrganiseEvent from django.contrib.auth.models import User

class SponsorTransaction(models.Model): spnsortype = models.CharField(max\_length=100)

org\_id = models.OneToOneField(OrganiseEvent, on\_delete=models.CASCADE) amount = models.IntegerField()

us = models.OneToOneField(User, on\_delete=models.CASCADE)

## Urls.py

from django.urls import path from django.conf import settings

from django.conf.urls.static import static from Sponsor import views as views

urlpatterns = [

path('', views.sponsor, name='sponsorIndex'), path('profile', views.profile, name='profile'),

path('enquireInfoMail', views.enquireInfoMail, name='enquireInfoMail'), path('sponsor\_event', views.sponsor\_event, name='sponsor\_event'), path('addsponsershipinfo', views.addsponsershipinfo, name='addsponsershipinfo'), path('addsponsership', views.addsponsership, name='addsponsership'), path('sponsored', views.sponsoredEvent, name='sponsoredEvent'),

path('enquire', views.enquire, name='enquire'), path('enquire\_event', views.enquire\_event, name='enquire\_event'), path('blogSponsor', views.blogSponsor, name='blog\_sponsor'),

path('blogSponsorWrite', views.writeBlogSponsor, name='blog\_write\_sponsor'), path('blogDetailsSpnsorship', views.blogDetailSponsor, name="blog\_Sponsor"), path('upcoming/', views.events, name='upcoming'),

path('event\_details/<int:id>', views.event\_details, name='event\_details'),

path('handlerequest/', views.handleRequest, name='handlerequest'), path('payAmount/', views.payAmount, name='payAmount'), path('amount/<int:id>', views.amount, name='amount'), path('transaction/<int:id>/',views.transaction,name='transaction'),

] + static(settings.MEDIA\_URL, document\_root=settings.MEDIA\_ROOT)

## Views.py

from django.shortcuts import render, get\_object\_or\_404 from Blog.models import BlogsInfo

from Organizer.models import EventDetails, SponsorShip, OrganiseEvent, Event\_Location from django.contrib.auth.models import User

from django.utils import timezone

from django.core.mail import send\_mail

from django.utils.datastructures import MultiValueDictKeyError from django.views.decorators.csrf import csrf\_exempt

from media.paytm import Checksum from .models import SponsorTransaction

MERCHANT\_KEY = 'P7HtuKq8&zpmzxZE'

def sponsoredEvent(request):

sponsor = SponsorTransaction.objects.get(us=request.user.id) event = OrganiseEvent.objects.get(pk=sponsor.org\_id.id)

return render(request, 'sponsored\_event.html', {'sponsor': sponsor, 'event': event})

def transaction(request, id):

spnsr = SponsorShip.objects.get(pk=id) print("Spnsor id is", spnsr.org\_id.id)

userid = User.objects.get(pk=request.user.id)

orgid = OrganiseEvent.objects.get(pk=spnsr.org\_id.id) if request.method == 'POST':

try:

spnsrtype = 'sponsertype' in request.POST and request.POST['sponsertype'] amount = 'amount' in request.POST and request.POST['amount']

# is\_private = request.POST['is\_private'] # spnsrtype

# spnsrtype = request.POST['sponsertype'] # amount = request.POST['amount']

try:

spn = SponsorTransaction.objects.get(org\_id=orgid) events = OrganiseEvent.objects

return render(request, 'sponsor\_event.html', {'error': 'Already Sponsored ', }) except SponsorTransaction.DoesNotExist:

spn = None

if spn is None:

transaction = SponsorTransaction(

spnsortype=spnsrtype, amount=amount, org\_id=orgid, us=userid) transaction.save()

return render(request, 'sponsored\_event.html') else:

events = OrganiseEvent.objects

return render(request, 'sponsor\_event.html', {'error': 'Already sponsored'}) except MultiValueDictKeyError:

spnsrtype = False

amount = False

return render(request, 'sponsored\_event.html')

def sponsor(request):

return render(request, 'sponsor\_index.html')

def profile(request):

profile = User.objects.filter(id=request.user.id)

return render(request, 'profile\_spnsr.html', {'profile': profile})

@csrf\_exempt

def handleRequest(request): form = request.POST response\_dict = {}

for i in form.keys(): response\_dict[i] = form[i]

if i == 'CHECKSUMHASH':

checksum = form[i]

verify = Checksum.verify\_checksum(response\_dict, MERCHANT\_KEY, checksum) if verify:

if response\_dict['RESPCODE'] == '01': print("successful")

return render(request, 'paymentstatus.html', {'respon': response\_dict})

else:

print('not successful'+response\_dict['RESPMSG'])

return render(request, 'paymentstatus.html', {'response': response\_dict})

@csrf\_exempt

def payAmount(request): # amount=10

if request.method == 'POST':

idd = request.POST.get('sp', False)

# idd = request.POST['sp'] print("id is", idd)

spnr = SponsorShip.objects.get(pk=idd) print("SponsorShip", spnr.ex\_platinum)

param\_dict = {

"MID": "bClJNU50572288826034", "ORDER\_ID": str(7),

"CUST\_ID": str(7),

"TXN\_AMOUNT": str(spnr.ex\_platinum), "CHANNEL\_ID": "WEB", "INDUSTRY\_TYPE\_ID": "Retail", "WEBSITE": "WEBSTAGING",

'CALLBACK\_URL': 'http://127.0.0.1:8000/Sponsor/amount/' + str(spnr.id)}

param\_dict['CHECKSUMHASH'] = Checksum.generate\_checksum( param\_dict, MERCHANT\_KEY)

return render(request, 'payment\_paytm.html', {'param\_dict': param\_dict}) @csrf\_exempt

def amount(request, id):

return render(request, 'payment\_paytm.html')

def payment(request): param\_dict = {

"MID": "bClJNU50572288826034",

"ORDER\_ID": str(request.user.id), "CUST\_ID": str(3),

"TXN\_AMOUNT": str(amt), "CHANNEL\_ID": "WEB", "INDUSTRY\_TYPE\_ID": "Retail", "WEBSITE": "WEBSTAGING",

'CALLBACK\_URL': 'http://127.0.0.1:8000/Sponsor/payAmount/'}

param\_dict['CHECKSUMHASH'] = Checksum.generate\_checksum( param\_dict, MERCHANT\_KEY)

# to contact event organizers def enquireInfoMail(request):

if request.method == 'POST': subject = request.POST['subject'] to = request.POST.get('to')

fromm = request.POST.get('from') mobile = request.POST['mobile'] message = request.POST['message'] send\_mail('Subject' + subject,

'Message' + message + 'Will Contact your soon', 'sachin.thakur9614@gmail.com',

[to, 'sachin.thakur@mca.christuniversity.in', ], fail\_silently=False)

return render(request, 'sponsor\_enquire.html', {'send': "MAIL SEND SUCCESSFULLY !"}) else:

events = OrganiseEvent.objects

return render(request, 'sponsor\_enquire.html', {'event': events}) # list out events to sponser

def sponsor\_event(request): try:

events = OrganiseEvent.objects

return render(request, 'sponsor\_event.html', {'event': events}) except:

return render(request, 'sponsor\_event.html')

# list out sponsership details of events to sponser # @csrf\_exempt

def addsponsershipinfo(request): if request.method == 'POST':

event = request.POST['event\_name']

spnsr\_det = SponsorShip.objects.filter(event\_title=event) events = OrganiseEvent.objects

return render(request, 'sponsor\_event.html', {'event': events, 'spnsr\_details': spnsr\_det}) else:

events = OrganiseEvent.objects

return render(request, 'sponsor\_event.html', {'event': events}) # save the sponsership info to db

def addsponsership(request): events = OrganiseEvent.objects

return render(request, 'sponsor\_event.html', {'event': events}) # list out all event info

def events(request):

events = OrganiseEvent.objects

return render(request, 'sponsor\_up\_coming.html', {'events': events}) # load more event info

def event\_details(request, id): if(OrganiseEvent.objects.filter(id=id) is not None):

events = OrganiseEvent.objects.filter(id=id) if(EventDetails.objects.filter(id=id) is not None):

info\_event = EventDetails.objects.filter(id=id) if(SponsorShip.objects.filter(id=id) is not None):

spnsr\_info = SponsorShip.objects.filter(id=id) if(Event\_Location.objects.filter(id=id) is not None):

loc = Event\_Location.objects.filter(id=id)

return render(request, 'sponsor\_event\_details.html', {'events': events, 'info': info\_event, 'spnsr': spnsr\_info, 'venue': loc})

else:

return render(request, 'sponsor\_event\_details.html', {'events': events, 'info': info\_event, 'spnsr': spnsr\_info})

else:

return render(request, 'sponsor\_event\_details.html', {'events': events, 'info': info\_event})

else:

return render(request, 'sponsor\_event\_details.html', {'events': events})

else:

events = OrganiseEvent.objects

return render(request, 'sponsor\_up\_coming.html', {'events': events})

def enquire(request):

events = OrganiseEvent.objects

return render(request, 'sponsor\_enquire.html', {'event': events})

def enquire\_event(request): if request.POST:

eve = request.POST['event\_name'] if eve == " ":

events = OrganiseEvent.objects

return render(request, 'sponsor\_enquire.html', {'event': events}) else:

events = OrganiseEvent.objects

even = OrganiseEvent.objects.get(event\_title=eve)

return render(request, 'sponsor\_enquire.html', {'event': events, 'evet': even})

else:

events = OrganiseEvent.objects

return render(request, 'sponsor\_enquire.html', {'event': events})

def blogSponsor(request):

return render(request, 'blog\_write\_sponsor.html')

def blogDetailSponsor(request):

blogsInfo = BlogsInfo.objects.filter(us=request.user.id)

return render(request, 'blog\_edit-sponsor.html', {'blogsInfo': blogsInfo})

def writeBlogSponsor(request): if request.method == 'POST':

title = request.POST['blog\_title'] pubDateTime = timezone.now() description = request.POST['description']

imageSecond = request.POST['image\_second'] imageFirst = request.POST['image\_first'] blogCategory = request.POST['blog\_category'] refrenceLinks = request.POST['refrence\_link'] UserType = 'Sponsor'

if request.user.is\_authenticated:

authorName = request.user.first\_name + " " + request.user.last\_name if title == '':

return render(request, 'blog\_write\_sponsor.html', {'error': "Title is not given"}) else:

title = request.POST['blog\_title'] if pubDateTime == '':

pubDateTime = timezone.now() if description == '':

return render(request, 'blog\_write\_sponsor.html', {'error': "Description is not written"}) else:

description = request.POST['description'] if imageFirst == '':

return render(request, 'blog\_write\_sponsor.html', {'error': "Image first is not given"}) else:

imageFirst = request.POST['image\_first'] if imageSecond == '':

return render(request, 'blog\_write\_sponsor.html', {'error': "Image Two is not given"}) else:

imageSecond = request.POST['image\_second'] if blogCategory == '':

return render(request, 'blog\_write\_sponsor.html', {'error': "Blog Category is not selected"}) else:

blogCategory = request.POST['blog\_category'] if refrenceLinks == '':

return render(request, 'blog\_write\_sponsor.html', {'error': "Refrence links not given"}) else:

refrenceLinks = request.POST['refrence\_link']

blogsInfo = BlogsInfo(title=title, pubDateTime=pubDateTime, description=description, imageSecond=imageSecond, imageFirst=imageFirst,

UserType=UserType, authorName=authorName, blogCategory=blogCategory, refrenceLinks=refrenceLinks, us\_id=request.user.id)

blogsInfo.save()

print("author name:" + authorName)

return render(request, 'blog\_write\_sponsor.html', {'error': title + "Blog is written"})

## PARTICIPANTS

* + - 1. **Models.py**

from django.db import models from django.conf import settings

from django.utils.timezone import now

# from froala\_editor.fields import FroalaField from django.contrib.auth import get\_user\_model

from Organizer.models import OrganiseEvent,EventDetails from django.contrib.auth.models import User

class ParticipateEvent(models.Model):

us = models.ForeignKey(User, on\_delete=models.CASCADE)

Event\_id = models.ForeignKey(OrganiseEvent,on\_delete=models.CASCADE)

class EventFeeback(models.Model):

subject = models.CharField(max\_length=50) feedback = models.CharField(max\_length=500) pubDateTime = models.DateTimeField(default=now)

Event\_id = models.ForeignKey(OrganiseEvent,on\_delete=models.CASCADE) us = models.ForeignKey(User, on\_delete=models.CASCADE)

def str (self): return self.subject

def summary(self):

return self.description[:175]

## Urls.py

from django.urls import path from django.conf import settings

from django.conf.urls.static import static from Participant import views as ve

urlpatterns = [

path('', ve.about, name='participantIndex'), path('participate', ve.participate, name='participate'), path('part\_profile', ve.part\_profile, name='part\_profile'),

path('part\_editProfile', ve.part\_editProfile, name='part\_editProfile'), path('part\_updateProfile', ve.part\_updateProfile, name='part\_updateProfile'), path('deleteparticipant', ve.deleteparticipant, name='deleteparticipant'), path('event\_details', ve.event\_details, name='event\_details'), path('participatein\_event/<int:id>',

ve.participatein\_event, name='participatein\_event'), path('enquire/<int:id>', ve.enquireInfo, name='enquire'), path('part\_enquire', ve.part\_enquire, name='part\_enquire'), path('enquireInfoMail', ve.enquireInfoMail, name='enquireInfoMail'), path('part\_shareresources/<int:id>',

ve.part\_shareresources, name='part\_shareresources'),

path('partblog', ve.partblog, name='partblog'), path('partWriteBlog', ve.partWriteBlog, name='part-blog\_write'),

path('partblogsDetail', ve.partblogsDetails, name='partblogsDetails'), path('req\_pdf', ve.req\_pdf, name="req\_pdf"),

path('part\_feedback', ve.part\_feedback, name="part\_feedback"), path('feedback\_load', ve.feedback\_load, name='feedback\_load'), path('feedback', ve.feedback, name="feedback"),

path('part\_for', ve.part\_for, name="part\_for")

] + static(settings.MEDIA\_URL, document\_root=settings.MEDIA\_ROOT)

## Views.py

from django.shortcuts import render, redirect, get\_object\_or\_404

from Organizer.models import OrganiseEvent, EventDetails, ShareResource from Participant.models import ParticipateEvent, EventFeeback

from accounts.models import UserProfile from django.contrib.auth.models import User from django.utils import timezone

from Blog.models import BlogsInfo from django.http import FileResponse from reportlab.pdfgen import canvas from reportlab.lib.pagesizes import A4 from django.http import HttpResponse import io

def about(request):

return render(request, 'participant\_index.html')

def participate(request):

even = EventDetails.objects

return render(request, 'participant\_index.html', {'even': even})

def part\_profile(request):

return render(request, 'part\_profile.html', {})

def part\_editProfile(request): part\_profiles = UserProfile.objects

return render(request, 'part\_editProfile.html', {'part\_profiles': part\_profiles})

def part\_updateProfile(request): if request.method == 'POST':

submitbutton = request.POST.get('Submit') if submitbutton:

user = request.POST['user'] fname = request.POST['fname'] lname = request.POST['lname'] email = request.POST['email']

profiles = User(id=request.user.id, username=user, first\_name=fname, last\_name=lname, email=email)

profiles.save() profiles.refresh\_from\_db()

edit\_profile = UserProfile.filter(us=request.user.id)

return render(request, 'part\_profile.html', {'part\_profile': edit\_profile})

def deleteparticipant(request): User.objects.filter(username=request.user.username).delete() #messages.success(request,'Profile have been Successfully deleted')

return render(request, 'login.html', {'error': 'Profile have been Successfully deleted'})

def event\_details(request):

events = OrganiseEvent.objects

return render(request, 'event\_details.html', {'events': events})

def part\_for(request):

part\_event = ParticipateEvent.objects.get(us=request.user.id) events = OrganiseEvent.objects.filter(id=part\_event.Event\_id\_id) return render(request, 'list\_part\_events.html', {'events': events})

def participatein\_event(request, id):

part\_event = ParticipateEvent.objects.filter(us=request.user.id) events = OrganiseEvent.objects.filter(us=request.user.id) part\_early = ParticipateEvent.objects.filter(

us\_id=request.user.id, Event\_id\_id=id) if part\_early is not None:

try:

events = OrganiseEvent.objects.get(pk=id) eventdet = EventDetails.objects.get(org\_id=id) print("hello", eventdet)

context = { 'events': events, 'eventdet': eventdet,

}

print("event is", events.event\_title)

return render(request, 'participated\_event.html', context) except EventDetails.DoesNotExist:

eventdet = None if eventdet:

pass else:

events = OrganiseEvent.objects.get(pk=id) context = { 'events': events,

'eventdet': eventdet,

}

else:

print("event is", events.event\_title)

return render(request, 'participated\_event.html', context)

participated\_event = ParticipateEvent( Event\_id\_id=id, us\_id=request.user.id)

participated\_event.save() participated\_event.refresh\_from\_db() try:

events = OrganiseEvent.objects.get(pk=id) eventdet = EventDetails.objects.get(org\_id=id) print("hello", eventdet)

context = { 'events': events, 'eventdet': eventdet,

}

print("event is", events.event\_title)

return render(request, 'participated\_event.html', context) except EventDetails.DoesNotExist:

eventdet = None if eventdet:

pass else:

events = OrganiseEvent.objects.get(pk=id) context = { 'events': events,

'eventdet': eventdet,

}

print("event is", events.event\_title)

return render(request, 'participated\_event.html', context)

def part\_shareresources(request, id): try:

resources = ShareResource.objects.get(org\_id\_id=id)

return render(request, 'part\_shareresources.html', {'resources': resources}) except ShareResource.DoesNotExist:

resources = None if resources:

pass else:

return render(request, 'part\_shareresources.html', {'error': 'Resources are not published'})

def enquireInfoMail(request): if request.POST:

subject = request.POST['subject'] to = request.POST['to']

fromm = request.POST['from'] mobile = request.POST['mobile'] message = request.POST['message'] send\_mail('Subject' + subject,

'Message' + message + 'Will Contact your soon', 'sachin.thakur9614@gmail.com',

[to, 'sachin.thakur@mca.christuniversity.in', ], fail\_silently=False)

return render('Participant\_index.html')

def enquireInfo(request, id):

event = OrganiseEvent.objects.get(pk=id)

usr\_det = UserProfile.objects.get(pk=request.user.id)

return render(request, 'part\_enquire.html', {'usr\_det': usr\_det, 'event': event})

def part\_enquire(request):

usr\_det = UserProfile.objects.get(pk=request.user.id)

return render(request, 'part\_enquire.html', {'usr\_det': usr\_det})

def partblogsDetails(request):

blogsInfo = BlogsInfo.objects.filter(us=request.user.id)

return render(request, 'part-blog\_edit.html', {'blogsInfo': blogsInfo})

def partblog(request):

return render(request, 'part-blog\_write.html')

def partWriteBlog(request):

if request.method == 'POST':

title = request.POST['blog\_title'] pubDateTime = timezone.now() description = request.POST['description']

imageSecond = request.POST['image\_second'] imageFirst = request.POST['image\_first'] blogCategory = request.POST['blog\_category'] refrenceLinks = request.POST['refrence\_link'] UserType = 'Participant'

if request.user.is\_authenticated:

authorName = request.user.first\_name + " " + request.user.last\_name if title == '':

return render(request, 'part-blog\_write.html', {'error': "Title is not given"}) else:

title = request.POST['blog\_title'] if pubDateTime == '':

pubDateTime = timezone.now() if description == '':

return render(request, 'part-blog\_write.html', {'error': "Description is not written"}) else:

description = request.POST['description'] if imageFirst == '':

return render(request, 'part-blog\_write.html', {'error': "Image first is not given"}) else:

imageFirst = request.POST['image\_first'] if imageSecond == '':

return render(request, 'part-blog\_write.html', {'error': "Image Two is not given"}) else:

imageSecond = request.POST['image\_second'] if blogCategory == '':

return render(request, 'part-blog\_write.html', {'error': "Blog Category is not selected"}) else:

blogCategory = request.POST['blog\_category'] if refrenceLinks == '':

return render(request, 'part-blog\_write.html', {'error': "Refrence links not given"}) else:

refrenceLinks = request.POST['refrence\_link']

blogsInfo = BlogsInfo(title=title, pubDateTime=pubDateTime, description=description, imageSecond=imageSecond, imageFirst=imageFirst,

UserType=UserType, authorName=authorName, blogCategory=blogCategory, refrenceLinks=refrenceLinks, us\_id=request.user.id)

blogsInfo.save()

print("author name:" + authorName)

return render(request, 'part-blog\_write.html', {'error': "Blog is selected"})

def req\_pdf(request):

response = HttpResponse(content\_type='application/pdf')

# response['Content-Disposition'] = 'as\_attachment=True; filename="mypdf.pdf"' response['Content-Disposition'] = 'inline'

filename = "mypdf.pdf"

buffer = io.BytesIO() buffer.pagesize = A4

p = canvas.Canvas(buffer) p.setLineWidth(.3) p.setFont('Helvetica', 12)

p.line(80, 770, 480, 770)

p.drawString(80, 755, 'HACKATHON USER DETAILS :') p.line(80, 750, 480, 750)

# Start writing the PDF here p.drawString(80, 725, 'Email ') p.drawString(200, 725, ': ') p.drawString(250, 725, request.user.email)

p.line(80, 700, 480, 700)

p.drawString(80, 675, 'User Name ') p.drawString(200, 675, ': ')

p.drawString(250, 675, request.user.username) p.drawString(80, 650, 'First Name ') p.drawString(200, 650, ': ')

p.drawString(250, 650, request.user.first\_name) p.drawString(80, 625, 'Last Name ') p.drawString(200, 625, ': ')

p.drawString(250, 625, request.user.last\_name)

p.line(80, 550, 480, 550)

p.drawString(

80, 532, 'DATA RETRIVED FROM DATA BASE IF ERROR CONTACT ADMIN') p.line(80, 525, 480, 525)

# End writing

p.showPage() p.save()

pdf = buffer.getvalue() buffer.close() response.write(pdf)

return response

def part\_feedback(request):

event\_id = ParticipateEvent.objects.get(us=request.user.id) event = OrganiseEvent.objects.filter(pk=event\_id.Event\_id\_id) usr = User.objects.get(pk=request.user.id)

usr\_det = UserProfile.objects.get(pk=request.user.id)

return render(request, 'part\_feedback.html', {'usr': usr, ' usr\_det': usr\_det, 'event': event})

def feedback\_load(request):

event\_id = ParticipateEvent.objects.get(us=request.user.id) event = OrganiseEvent.objects.filter(pk=event\_id.Event\_id\_id) usr = User.objects.get(pk=request.user.id)

usr\_det = UserProfile.objects.get(pk=request.user.id)

return render(request, 'part\_feedback.html', {'usr': usr, ' usr\_det': usr\_det, 'event': event, 'event\_part': event})

def feedback(request):

if request.method == 'POST':

event = OrganiseEvent.objects.filter(us=request.user.id) subject = request.POST['subject']

pubDateTime = timezone.now() feedback = request.POST['feedback'] UserType = 'Participant'

if subject == '':

return render(request, 'part\_feedback.html', {'error': "Subject is not given"})

else:

subject = request.POST['subject'] if pubDateTime == '':

pubDateTime = timezone.now() if feedback == '':

return render(request, 'part\_feedback.html', {'error': "Feedback is not written"}) else:

feedback = request.POST['feedback']

eventFeeback = EventFeeback(subject=subject, feedback=feedback, pubDateTime=pubDateTime, Event\_id\_id=event, us\_id=request.user.id)

eventFeeback.save() eventFeeback.refresh\_from\_db() print("author name:" + authorName)

return render(request, 'part\_feedback.html', {'error': "Feedback is stored"})

## SCREENSHOTS

* + 1. **LOGIN AND REGISTRATION**

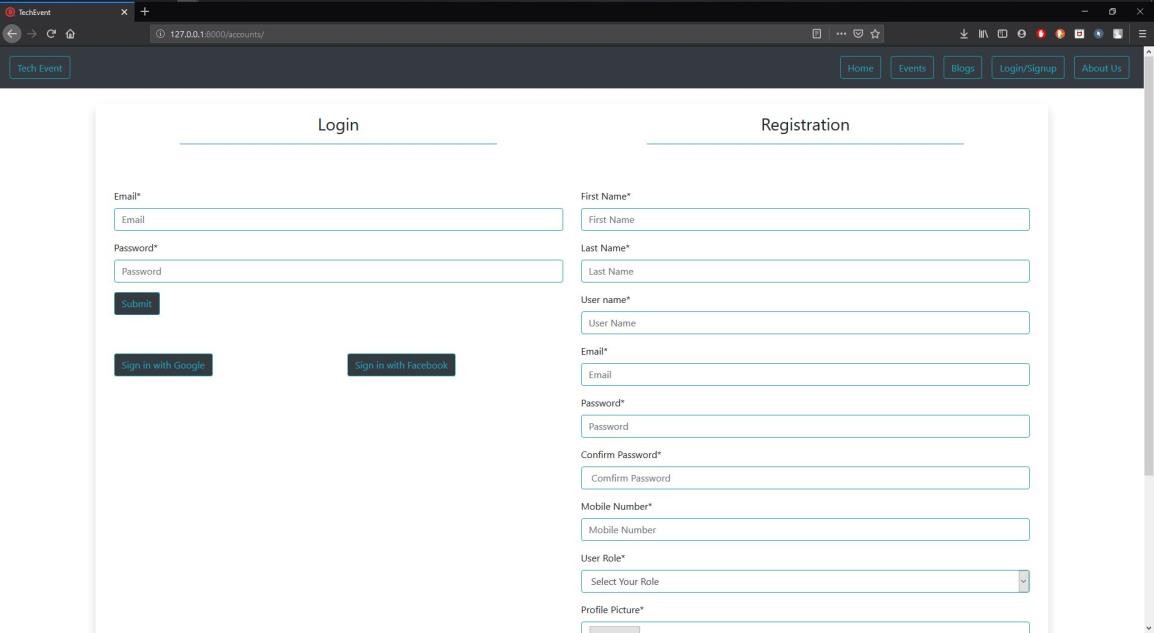


Fig .5.1 login and registration

## OUR TEAM AND SERVICES

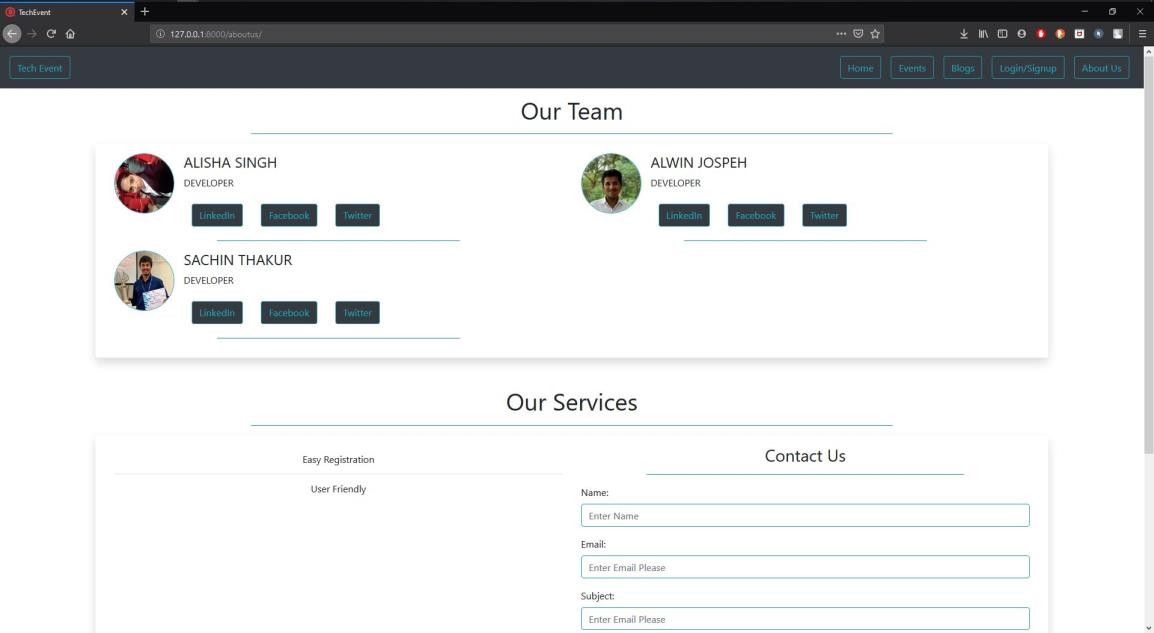


Fig. 5.2 our team and services

## SPONSORS FOR EVENT

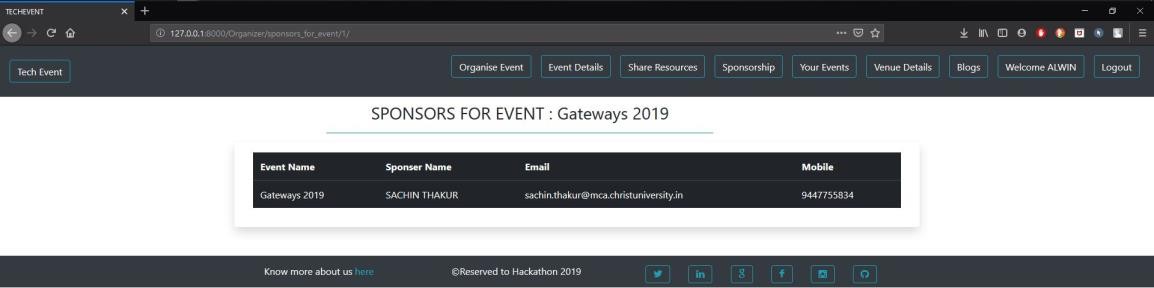


Fig. 5.3 sponsors for event

## PARTICIPANTS FOR EVENT

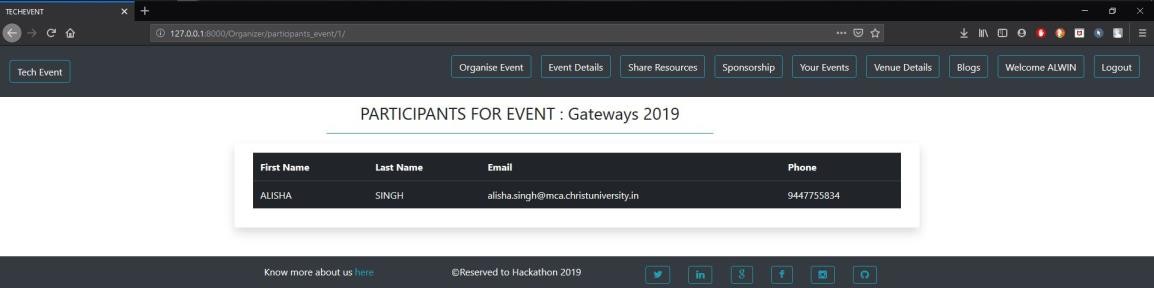


Fig. 5.4 participants for event

## EVENT DETAILS FOR PARTICIPANTS

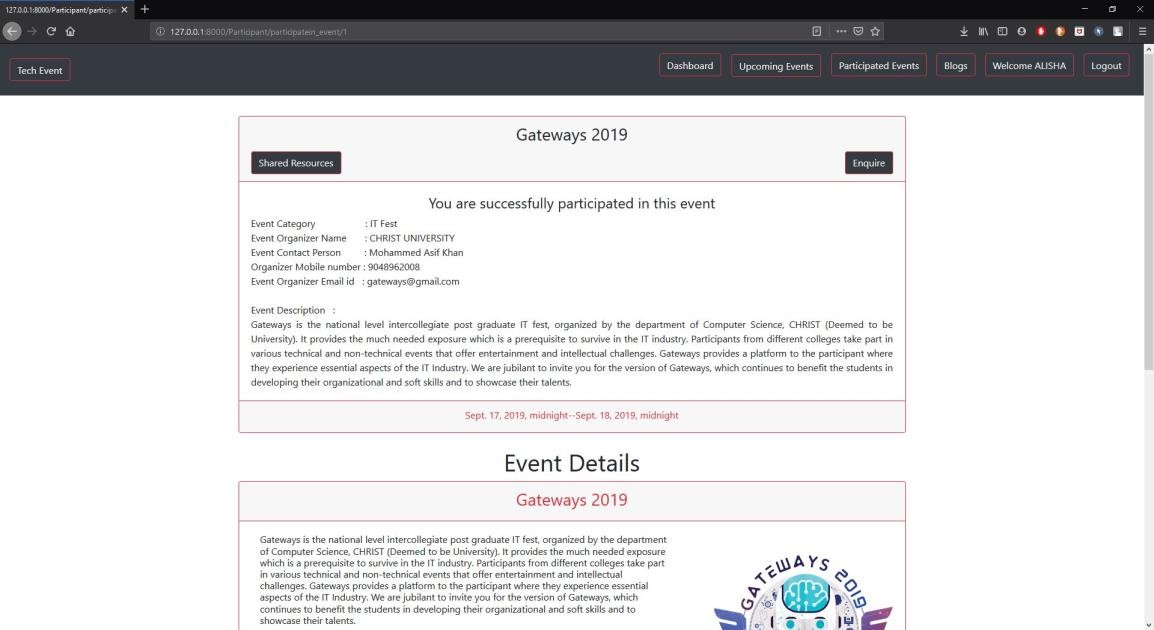


Fig. 5.5 Event details for participants

## SPONSORED EVENTS

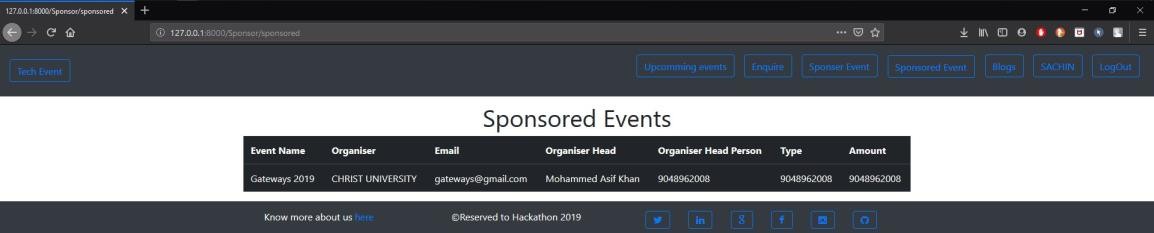


Fig. 5.6 Sponsored events

## PARTICIPATED EVENTS

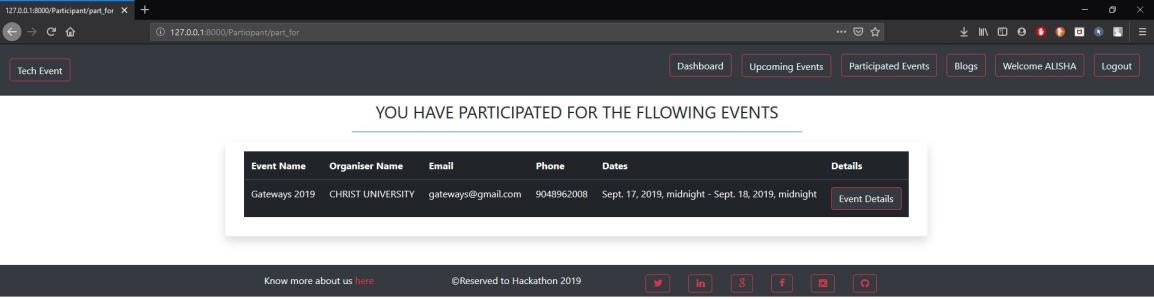


Fig. 5.7 Participated events

## UPCOMING EVENTS

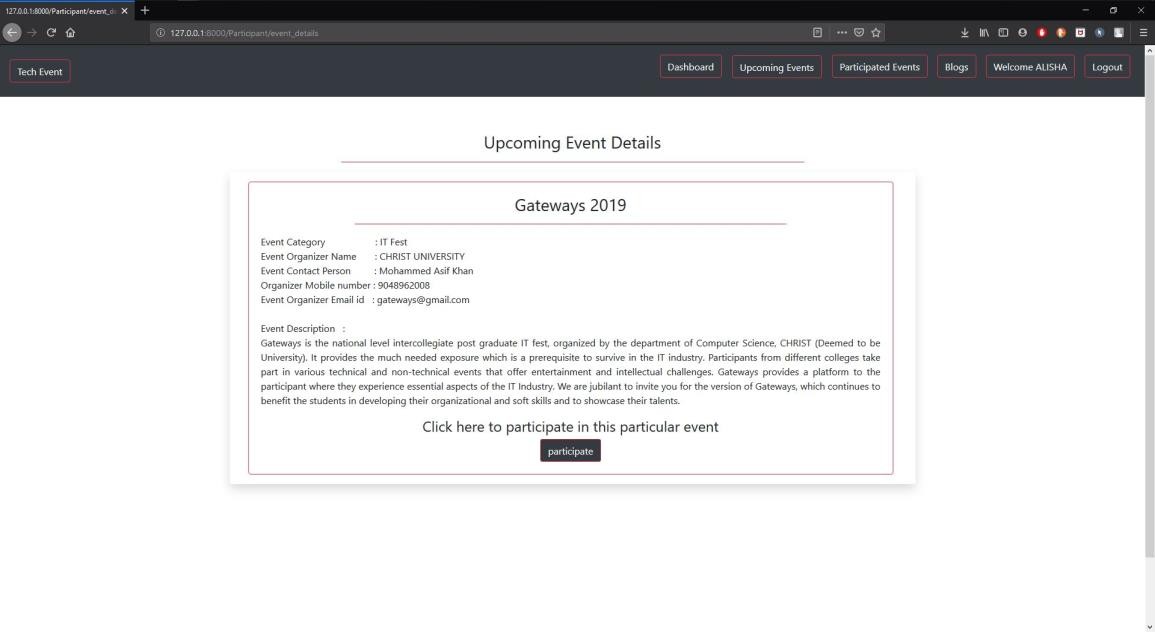


Fig. 5.8 Upcoming events

## ENQUIRE ABOUT EVENT

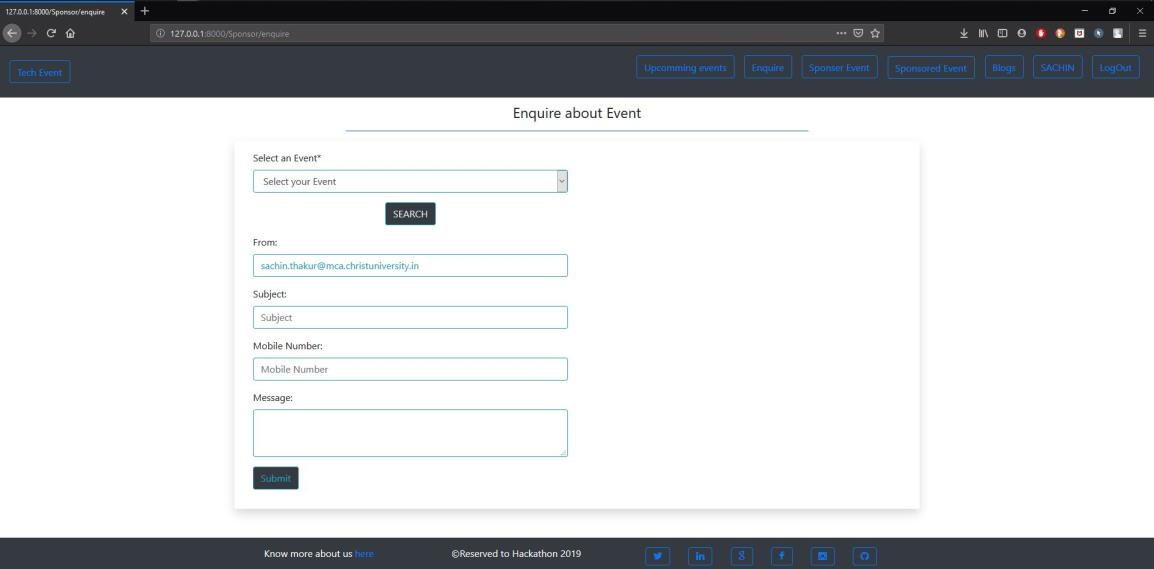


Fig. 5.9 Enquire about event

## SPONSOR EVENT

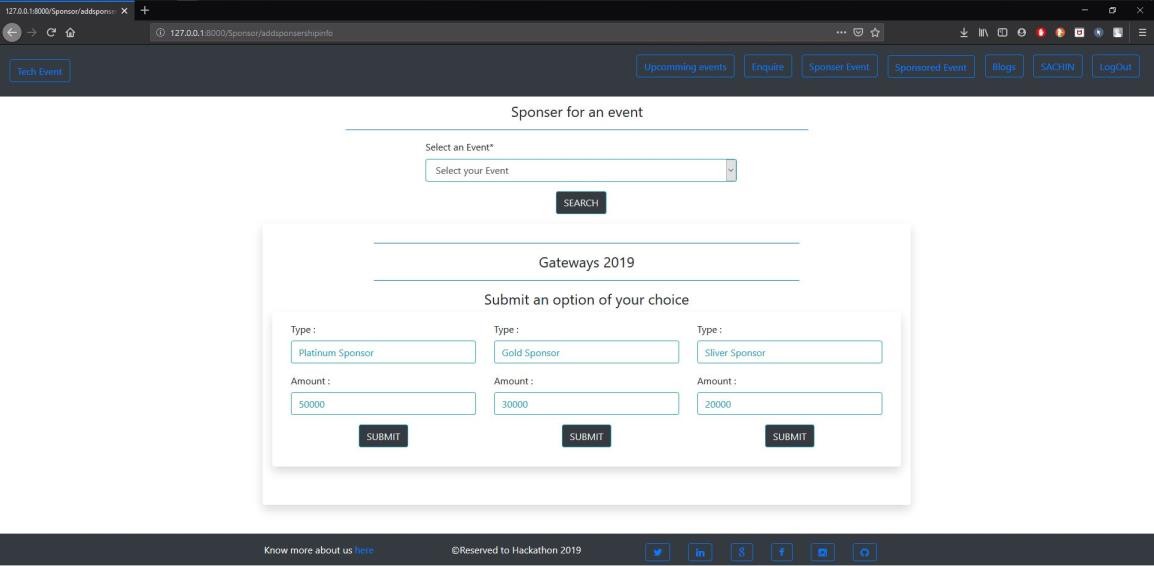


Fig. 5.10 Sponsor event

## PROFILE

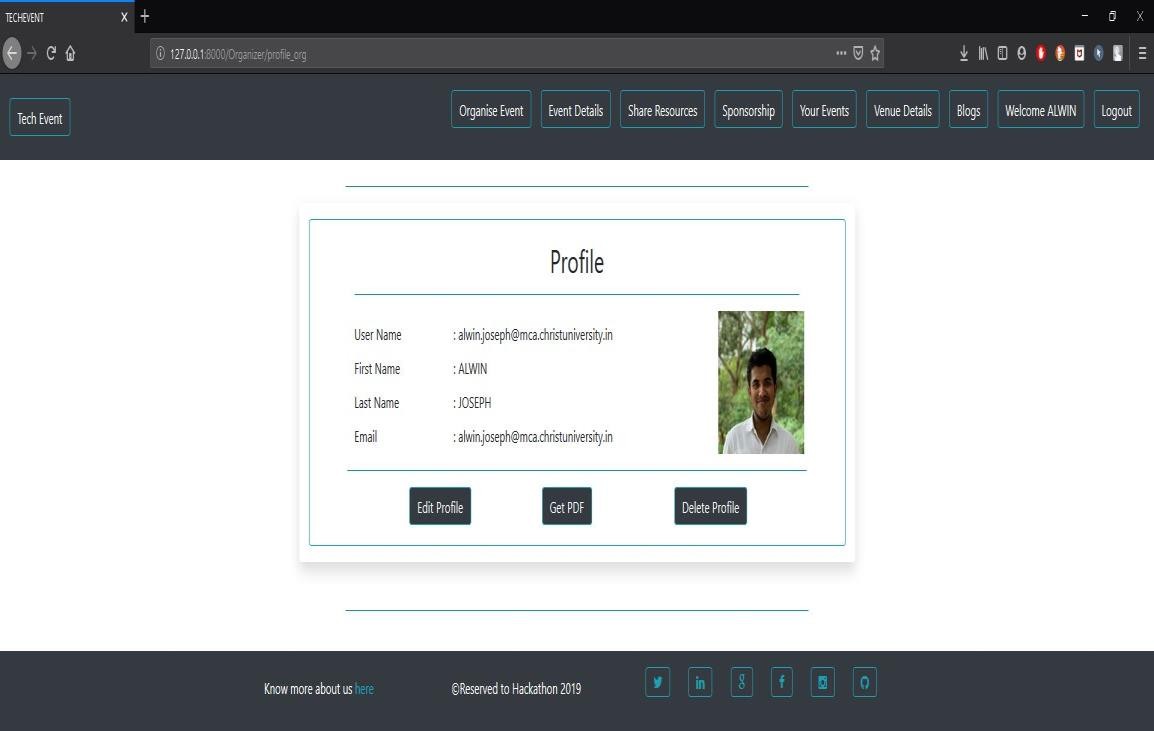


Fig. 5.11 Profile

## WRITE BLOGS

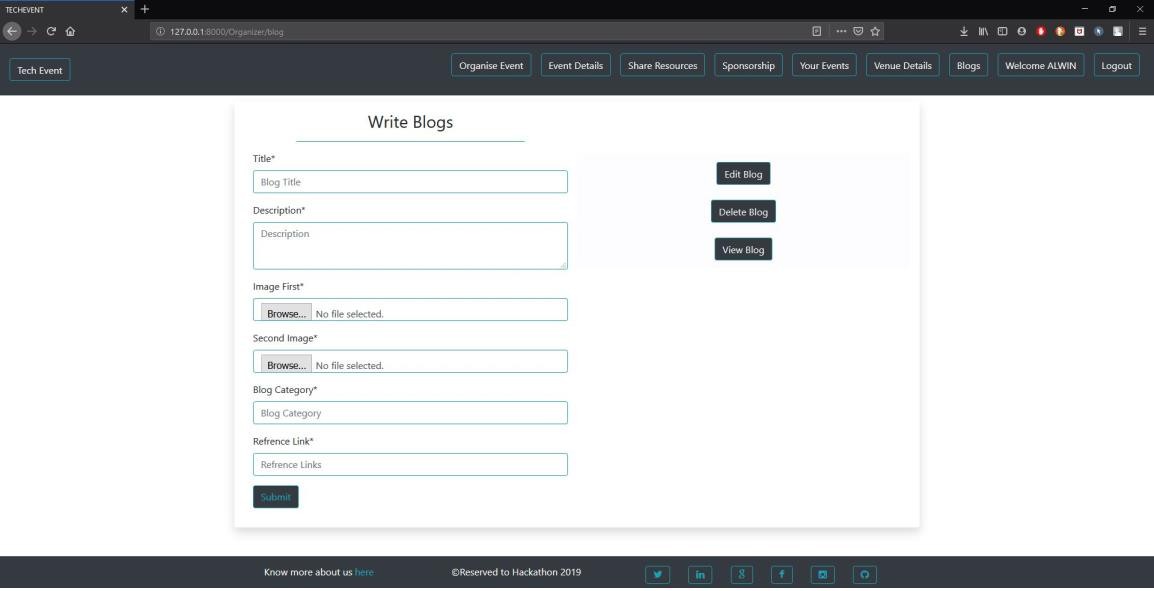


Fig. 5.12 Write blogs

## EVENT VENUE

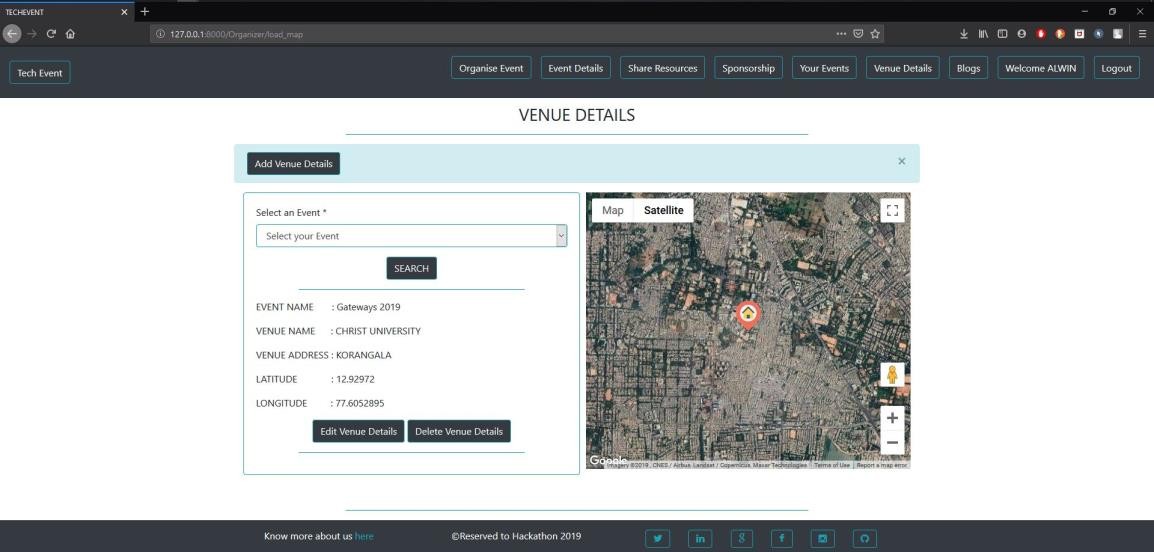


Fig. 5.13 Event venue

## REGISTERED EVENTS

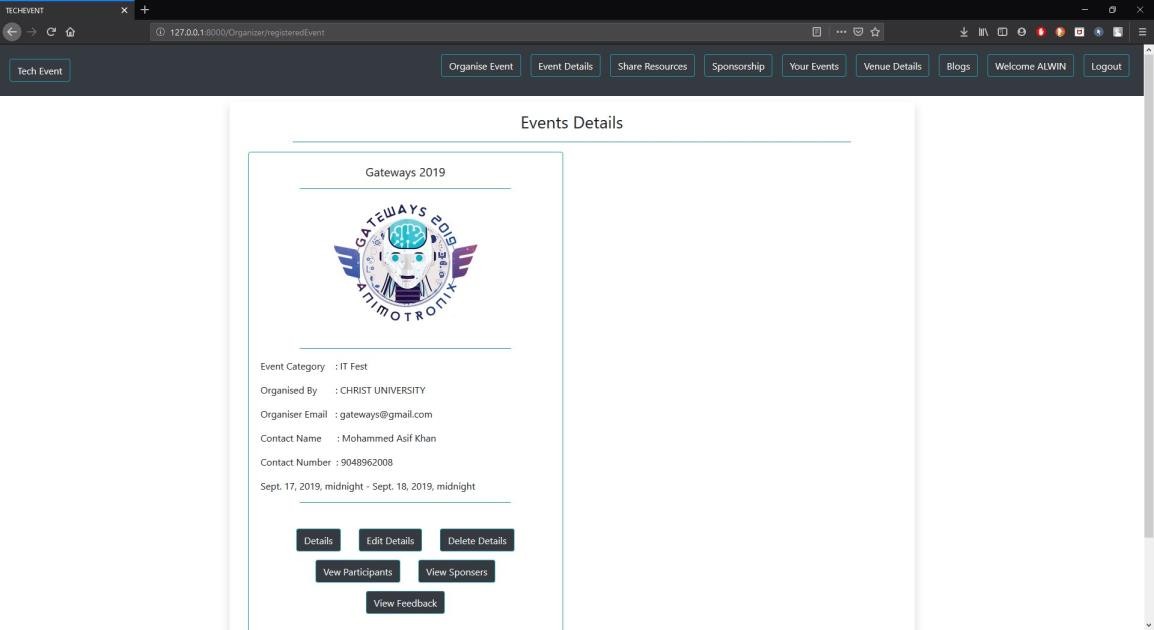


Fig. 5.14 Registered events

## SPONSORSHIP DETAILS

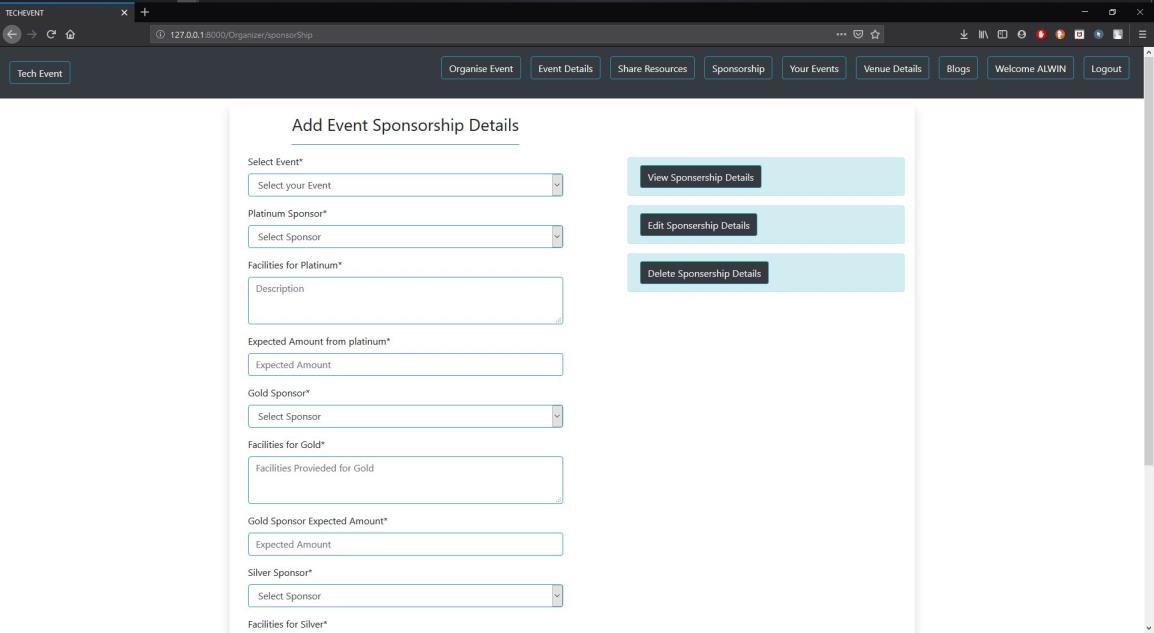


Fig. 5.15 Sponsorship details

## SHARE RESOURCES

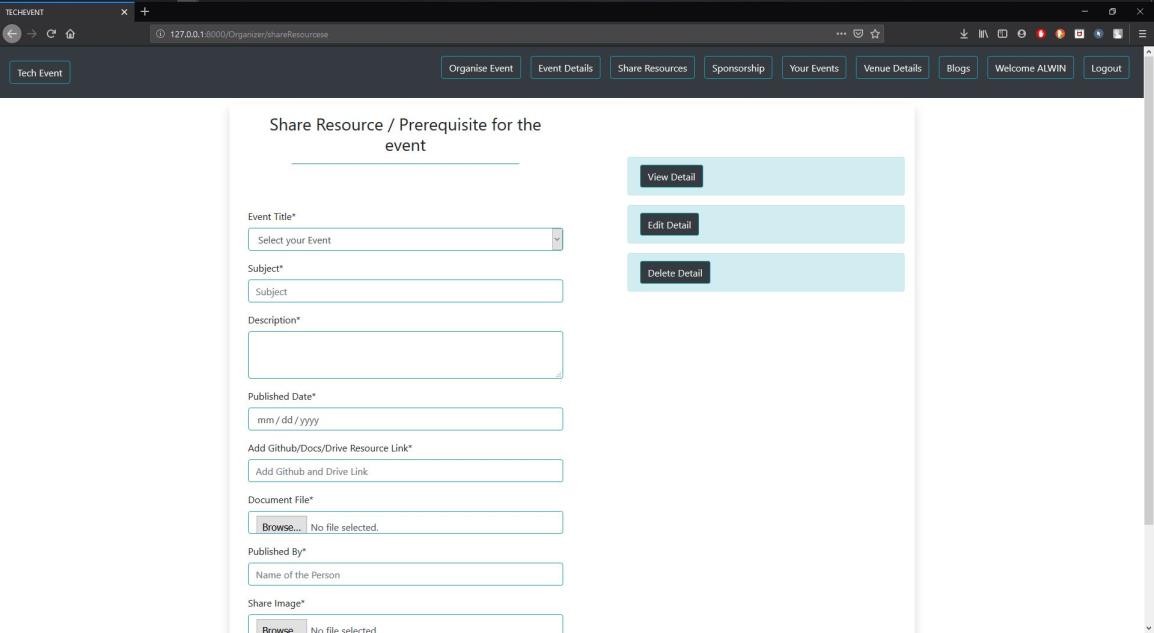


Fig. 5.16 Share resources

## ENTER EVENT DETAILS

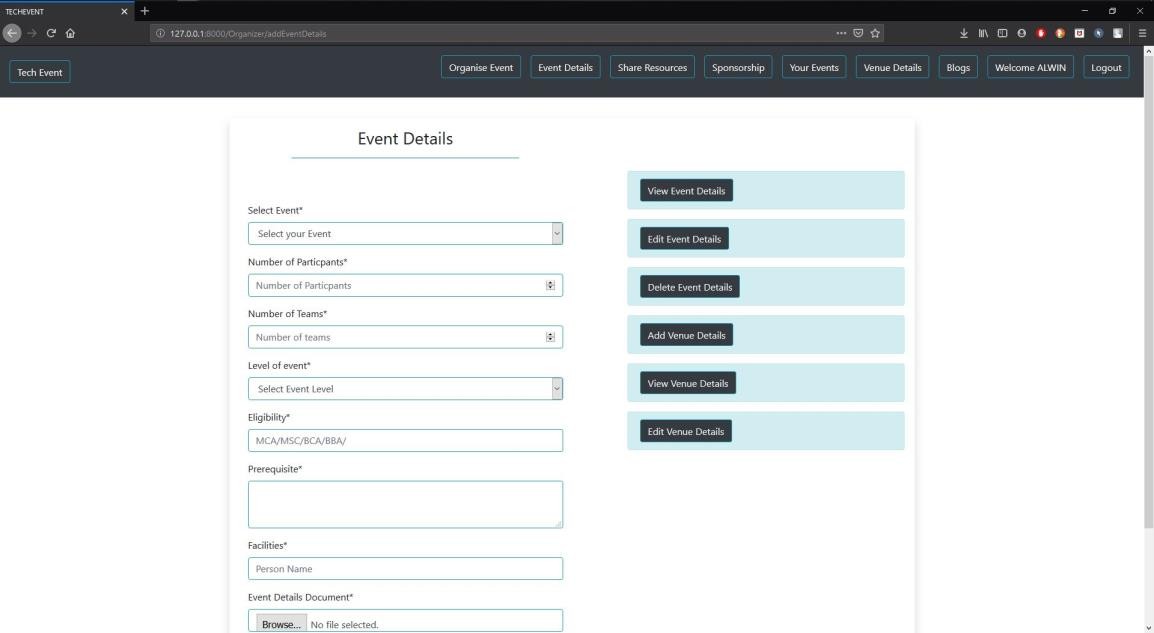


Fig. 5.17 Enter event details

## REGISTER EVENT

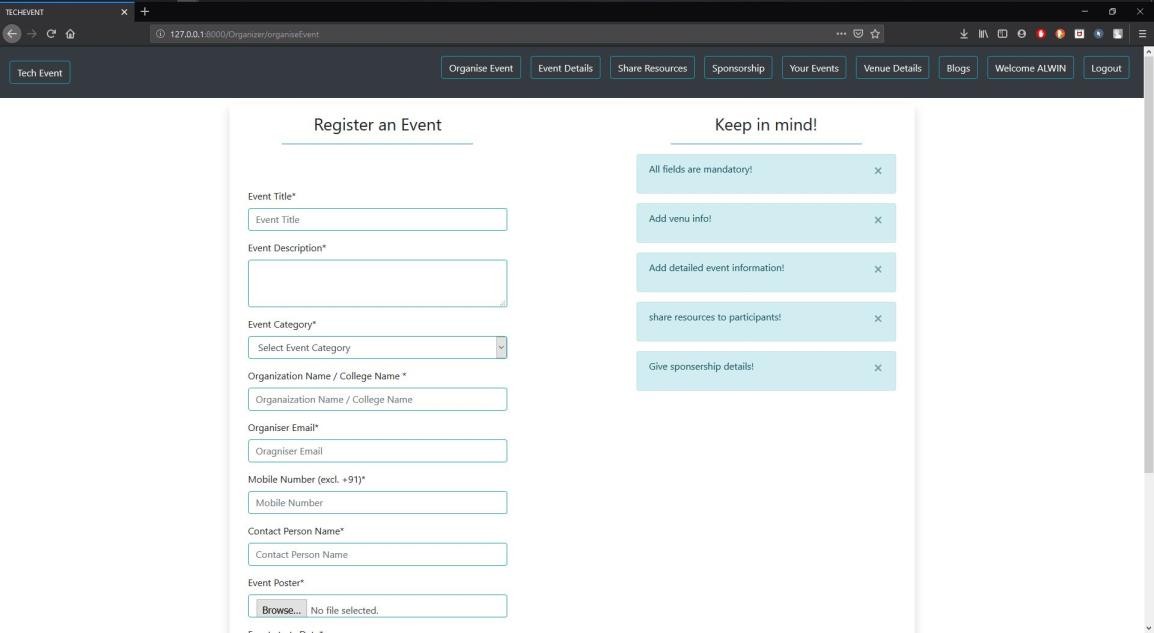


Fig. 5.18 Register event

## ORGANISER DASHBOARD

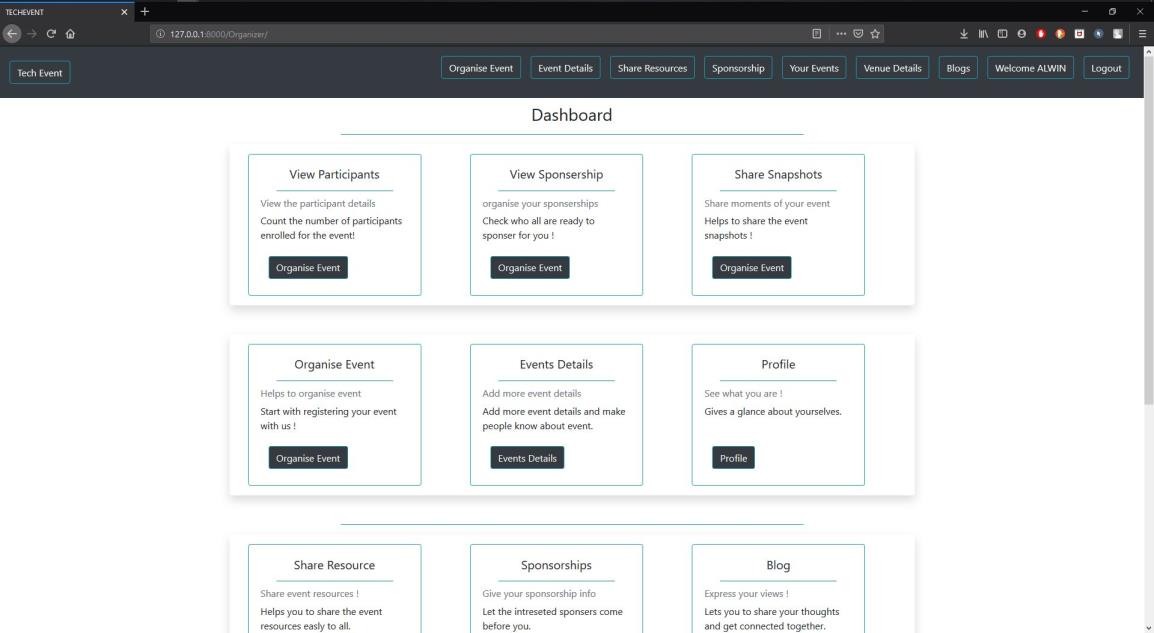


Fig. 5.19 Organiser dashboard

## BLOG DETAILS

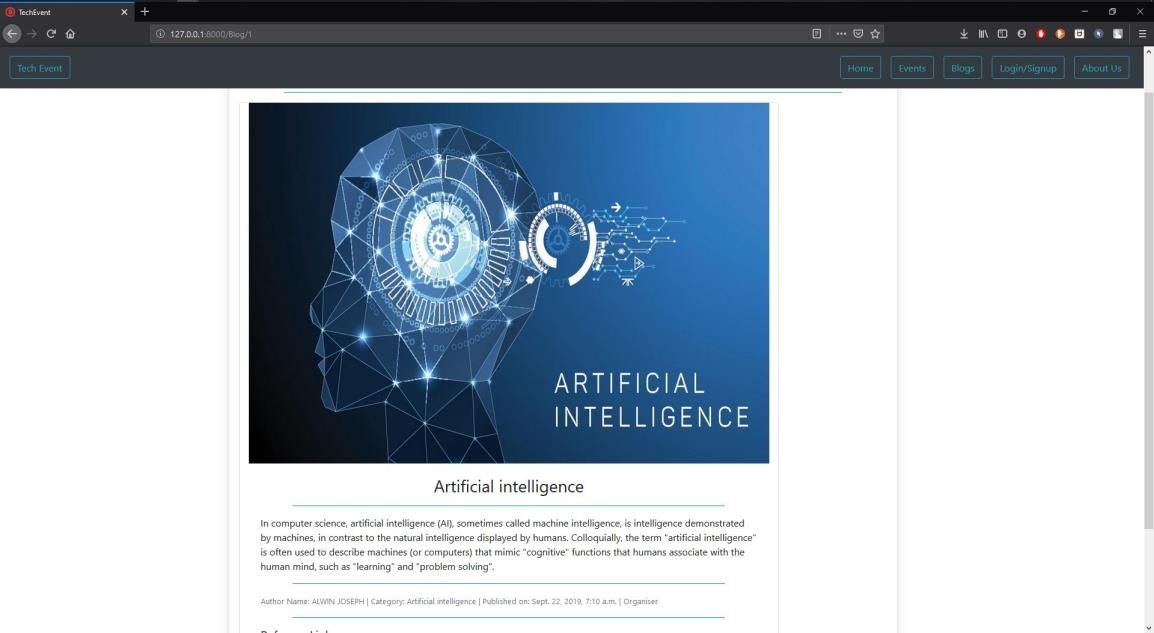


Fig. 5.20 Blog details

## ADMIN SIDE

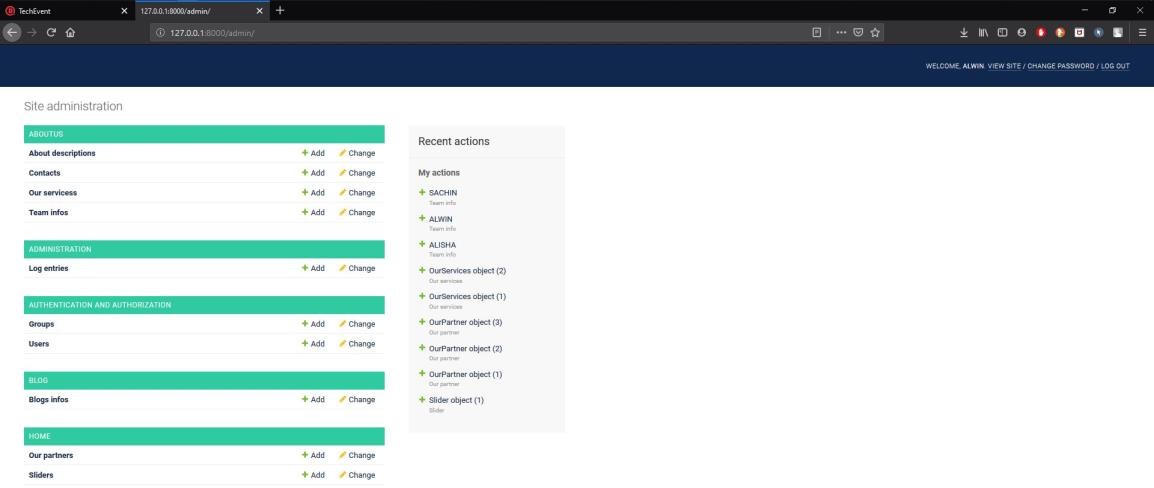


Fig. 5.21 Admin side

## LOG ENTRIES



Fig. 5.22 Log entries

# TESTING

## TEST PLAN

### Testing provides an objective, independent view of the software to allow the developers to appreciate and understand the risks of software implementation. Test techniques include, but are not limited to, the process of executing a program or an application with the intent of finding errors, bugs and defects. Software testing, depending on the testing method employed, can be implemented at any time in the development process. Traditionally most of the test effort occurs after the requirements have been defined and the coding process has been completed.

A primary purpose of testing is to detect software failures so that defects may be discovered and corrected. Testing cannot establish that a product functions properly under all conditions but can only establish that it does not function properly under specific conditions. The scope of software testing often includes examination of code as well as execution of that code in various environments and conditions as well as examining all aspects of code: does it do what it is supposed to do and do what it needs to do. Not all software defects are caused by coding errors. One common source of expensive defects is caused by requirement gaps, e.g., unrecognized requirements that result in errors of omission by the program designer. A common source of requirements gaps in non-functional requirements such as testability, scalability, maintainability, usability, performance and security.

A very fundamental problem with software testing is that testing under all combinations of inputs and preconditions (initial state) is not feasible, even with a simple product. This means that the number of defects in a software product can be very large and defects that occur infrequently are difficult to find in testing. More significantly, non- functional dimensions of quality (how it is supposed to be versus what it is supposed to do) usability, scalability, performance, compatibility, reliability can be highly subjective; something that constitutes sufficient value to one person may be intolerable to another. We plan to test our project by various methods of testing. We aim at testing our project and run it under maximum testing combinations possible. By doing this we can check the functionality and the durability of our product. We plan to test our project in three phases, being, unit testing, integration testing and validation testing. Integration

testing is the most detailed and longest process of testing as it consists of the top down approach, a bottom up approach, umbrella approach, black box testing and white box testing.

* 1. **TEST CASES**
     1. **Login Form**

Table 6.1 Login test cases

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test Scenario** | **Expected Result** | **Observed Result** | **Result** |
| 1 | Form displayed with all the controls | Display the login form with all the controls | Form loaded successfully with all the controls | Pass |
| 2 | Enter wrong username and password | Display login form again with invalid message. | Message displayed | Pass |
| 3 | Enter not registered details. | Display invalid message. | Message displayed | Pass |
| 4 | Enter correct username and correct password of any of the registered users. | The appropriate home page is loaded. | Login to their home pages | Pass |
| 5 | Press login button without filling the username and password | Display warning message to fill the details | Warning message displayed | Pass |

## Registration Form

Table 6.2 Registration form test cases

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test Scenario** | **Expected Result** | **Observed Result** | **Result** |
| 1 | Form displayed | Display form with all controls | Form loaded with all controls | Pass |
| 2 | Click Register button without data | Display warning message to fill all the details | Warning Message displayed to fill data | Pass |
| 3 | Enter correct data and fill the registration form | The data is stored in the database and a mail is sent to the registered mail address. And a message for successful registration is displayed | Registration is successful and a message to login appears. | Pass |
| 4 | Enter Wrong Mail Address | Display error message to enter correct mail address | Error message displayed | Pass |
| 5 | Enter Wrong Phone number | Display error message to enter correct phone number | Error message displayed | Pass |
| 6 | Enter different password | Display error message to enter matching passwords | Error message displayed | Pass |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 7 | Enter invalid picture format | Display error message to select correct file type | Error message displayed | Pass |

## Organise Event Form

Table 6.3 Organise event form test cases

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test Scenario** | **Expected Result** | **Observed Result** | **Result** |
| 1 | Form displayed | Display form with all controls | Form loaded with all controls | Pass |
| 2 | Click Submit button without data | Display warning message to fill all the details | Warning Message displayed | Pass |
| 3 | Enter correct data to the fields | Event registered successfully and redirect to the events page | Events gets successfully registered | Pass |
| 4 | Click save button with wrong inputs | Display corresponding warning messages | Message displayed | Pass |
| 5 | Give invalid inputs to different fields | Appropriate validation messages are shown | The messages gets shown | Pass |

## Organiser Dashboard

Table 6.4 Organiser dashboard test cases

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test Scenario** | **Expected Result** | **Observed Result** | **Result** |
| 1 | On login the organizer dashboard is displayed | Dashboard is displayed | Dashboard appears when login as organizer | Pass |
| 2 | Check different navigation links | All links redirect to the corresponding pages | All links worked | Pass |

## Your Events

Table 6.5 Your events test cases

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test Scenario** | **Expected Result** | **Observed Result** | **Result** |
| 1 | Page loaded with all the events registered | Events listed | Events listed successfully | Pass |
| 2 | Click on various CRUD buttons for the event | CRUD operations working | All operations worked | Pass |

## Participate Events

Table 6.6 Participate events test cases

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test Scenario** | **Expected Result** | **Observed Result** | **Result** |
| 1 | Page loaded with all the events registered | Events listed | Events listed successfully | Pass |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2 | Event details are listed | All the coming up events are listed | All events are listed | Pass |
| 3 | Click on participate button | The participation gets recorded | Successful in participating the event | Pass |
| 4 | Again click participate button | Warning message should be shown | Warning message is shown | Pass |

## Enquire about an event

Table 6.7 Enquire about an event test cases

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test Scenario** | **Expected Result** | **Observed Result** | **Result** |
| 1 | Page loaded with all the contact form | Page loaded with the required controls | Page loaded successfully | Pass |
| 2 | Press the contact button without submit button | The contact form shows warning message | The message gets displayed | Pass |
| 3 | Select the event for contact | The to mail id gets filled | The to mail address appears | Pass |
| 4 | Check the mail sent items | The sent mail can be seen | The mail gets sent | Pass |

## Project home page

Table 6.8 Project home page test cases

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test Scenario** | **Expected Result** | **Observed Result** | **Result** |
| 1 | The pages are loaded with the appropriate controls | The pages and controls are loaded | The components are loaded properly | Pass |
| 2 | Checking different links | The links are redirecting to the appropriate pages | All links worked | Pass |

## Participant Dashboard

Table 6.9 Participant dashboard test cases

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test Scenario** | **Expected Result** | **Observed Result** | **Result** |
| 1 | On login the dashboard is displayed | Dashboard is displayed | Dashboard appears when login as participant | Pass |
| 2 | Check different navigation links | All links redirect to the corresponding pages | All links worked | Pass |

## Event feedback

Table 6.10 Event feedback test cases

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test Scenario** | **Expected Result** | **Observed Result** | **Result** |
| 1 | On selecting the | The feedback | Form loaded | Pass |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | event participated the feedback can be given | form is displayed when the button is clicked | correctly |  |
| 2 | Fill the form with correct details | The message gets sent | Message sent successfully | Pass |
| 3 | Fill the form with invalid inputs | The corresponding warning messages are shown | The messages shown as warnings | Pass |

## Blogs

Table 6.11 blogs test cases

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test Scenario** | **Expected Result** | **Observed Result** | **Result** |
| 1 | The page loaded with all the controls | The page is shown with all the controls | The page is loaded with all the controls | Pass |
| 2 | Fill the form with appropriate data | The Data gets saved | The blogs gets added | Pass |
| 3 | Fill the fields with inappropriate data | Appropriate warnings are shown | The warnings worked | Pass |

## Contact page

Table 6.12 contact page test cases

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test Scenario** | **Expected Result** | **Observed Result** | **Result** |
| 1 | The page loaded with | The page is | The page is | Pass |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | all the controls | shown with all the controls | loaded with all the controls |  |
| 2 | Fill the form with appropriate data | The Data gets saved | The blogs gets added | Pass |
| 3 | Fill the fields with inappropriate data | Appropriate warnings are shown | The warnings worked | Pass |

## Admin

Table 6.13 admin test cases

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test Scenario** | **Expected Result** | **Observed Result** | **Result** |
| 1 | change the project URL to the admin page | admin login window appears | window loaded correctly | Pass |
| 2 | Add details to the project pages | Add the details should be saved and viewed in the pages corresponding | could save the details and view in the modified the places | Pass |
| 3 | Delete details from the admin side | Details get deleted | details got deleted | Pass |

# CONCLUSION

### As we started the development of Tech Event, we were not much clear with the idea, so as we started the development there was clearly started to come while facing issues while developing module and we were looking at how we will integrate particular module. We were clear with the idea in a way that we were knowing our target audience and as our team have been to many workshops and college event so while taking feedback from organizer of the college events. This project helps the organizer of the colleges and schools so we have developed with the future requirement required to host a particular event and we have made easy for them to host workshops, fest and tech talks etc. and to get sponsors and participants of that event. Tech Events holds the enrollment to sponsors from NGO, startups, MNC and incubator cells of a particular organization. As the same way we have given the enrollment to participants from different colleges and school which helps in the way where we need less efforts to either promote the events and also less team efforts is required for a marketing and to get the sponsors by sponsored team of the event hosted by a particular colleges.

* 1. **FUTURE ENHANCEMENT**
     1. Payment gateways implementation is half done in which we need a college accounts detail to transfer the amount.
     2. Community should be created to the users, which will help to collaborate with different colleges and as the same way sponsors and Participants.
     3. As we were focused with to have functionality and we need to focus on UI design of Tech Event.
     4. Deployment of product on servers.
     5. We need to add API’s for having online IDE to run Hackathon or coding and debugging events.
     6. Participants team collaboration with different colleges.

**REFERENCES**

### [1]. "Introducing Django". The Django Book., 29 July 2018.

[2]. "Django 2.1 released" Django weblog. 2 August 2018.

[3]. “Django documentation” url: “https://docs.djangoproject.com/en/2.1/” [4]. Patrick Dalton, Paul Whitehead, “SQL SERVER BLACK BOOK”

Dream Tech New Delhi, 2005.

[5]. Roger Pressman, “Software Engineering A Practitioner’s Approach”, McGraw-Hill Company, 2001.