**DAKSHA-STATE YOUTH FESTIVAL AUTOMATION SYSTEM**

*Project Report Submitted By*

**MUHAMMED SONU RIBIN KA**

**Reg. No.: AJC20MCA-2054**

*In Partial fulfillment for the Award of the Degree Of*

**MASTER OF COMPUTER APPLICATIONS (2 Years)**

**(MCA)**

**APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY**



**AMAL JYOTHI COLLEGE OF ENGINEERING KANJIRAPPALLY**

[Affiliated to APJ Abdul Kalam Technological University, Kerala. Approved by AICTE, Accredited by NAAC with ‘A’ grade. Koovappally, Kanjirappally, Kottayam, Kerala – 686518]

**2021-2022**

**DEPARTMENT OF COMPUTER APPLICATIONS**

**AMAL JYOTHI COLLEGE OF ENGINEERING**

**KANJIRAPPALLY**



**CERTIFICATE**

This is to certify that the Project report, “**DAKSHA-STATE YOUTH FESTIVAL AUTOMATION SYSTEM”** is the bonafide work of **MUHAMMED SONU RIBIN KA (Reg.No:AJC20MCA-2054)** in partial fulfillment of the requirements for the award of the Post Graduation Degree of Master of Computer Applications under APJ Abdul Kalam Technological University during the year 2021-22.

**Ms.Gloriya Mathew Rev.Fr.Dr.Rubin Thottupurathu Jose**

**Internal guide Co-ordinator**

**Rev.Fr.Dr.Rubin Thottupurathu Jose**

**Head of the Department**

**DECLARATION**

I hereby declare that the project report **“DAKSHA-STATE YOUTH FESTIVAL AUTOMATION SYSTEM”** is a bonafided work done at Amal Jyothi College of Engineering, towards the partial fulfilment of the requirements for the award of the Post Graduation Degree of Master of Computer Applications (MCA) from APJ Abdul Kalam Technological University, during the academic year 2021-2022.

**Date: 22/07/2022 MUHMAMMED SONU RIBIN KA**

**KANJIRAPPALLY Reg. No: AJC20MCA-2054**

**ACKNOWLEDGEMENT**

First and foremost, I thank God almighty for his eternal love and protection throughout the project. I take this opportunity to express my gratitude to all who helped me in completing this project successfully. It has been said that gratitude is the memory of the heart. I wish to express my sincere gratitude to our manager **Rev. Fr. Dr. Mathew Paikatt** and Principal Dr. Lillykutty Jacob for providing good faculty for guidance.

I owe a great depth of gratitude towards our Head of the Department and our project coordinator **Rev.Fr.Dr. Rubin Thottupurathu Jose** for helping us with their valuable suggestions and for overwhelming concern and guidance from the beginning to the end of the project. I would also like to express sincere gratitude to my guide**, Ms**. **Gloriya mathew** for her inspiration and helping hand.

I thank our beloved teachers for their cooperation and suggestions that helped me throughout the project. I express my thanks to all my friends and classmates for their interest, dedication, and encouragement shown towards the project. I convey my hearty thanks to my family for the moral support, suggestions, and encouragement to make this venture a success.

MUHAMMED SONU RIBIN KA

**ABSTRACT**

Kerala state youth festival is one of the biggest cultural event in Asia, Keralians are very proud to conduct this festival. Participants from all over Kerala attend the programs. But the existing system of the conducting of youth festival is not so fair. Everything from begining to end is being done manually, from declaration to result publishing everything is done as paper and pen form. All the records are stored just in ledger notes which is not a safe way of storing files,because damage of these ledger books may clear all datas of the concerened youth festivals.

As the technology has made a vast grow in the world,use of these technologies can be implemented for state youth festival also. My project proposes such a system to make everything form registration to result publication automated. This system works mainly with 4 actors:-admin, staff, district rep, participant by which each of them having their own duties. Admin manage all other users and has much more functions also.

**CONTENT**

|  |  |  |
| --- | --- | --- |
| **Sl. No** | **Topic** | **Page No** |
| **1** | **INTRODUCTION** | **2** |
| **1.1** | **PROJECT OVERVIEW** | **3** |
| **1.2** | **PROJECT SPECIFICATION** | **3** |
| **2** | **SYSTEM STUDY** | **4** |
| **2.1** | **INTRODUCTION** | **4** |
| **2.2** | **EXISTING SYSTEM** | **6** |
| **2.3** | **DRAWBACKS OF EXISTING SYSTEM** | **7** |
| **2.4** | **PROPOSED SYSTEM** | **6** |
| **2.5** | **ADVANTAGES OF PROPOSED SYSTEM** | **6** |
| **3** | **REQUIREMENT ANALYSIS** | **8** |
| **3.1** | **FEASIBILITY STUDY** | **10** |
| **3.1.1** | **ECONOMICAL FEASIBILITY** | **10** |
| **3.1.2** | **TECHNICAL FEASIBILITY** | **11** |
| **3.1.3** | **BEHAVIORAL FEASIBILITY** | **11** |
| **3.2** | **SYSTEM SPECIFICATION** | **12** |
| **3.2.1** | **HARDWARE SPECIFICATION** | **12** |
| **3.2.2** | **SOFTWARE SPECIFICATION** | **12** |
| **3.3** | **SOFTWARE DESCRIPTION** | **12** |
| **3.3.1** | **PHP** | **12** |
| **3.3.2** | **MYSQL** | **13** |
| **4** | **SYSTEM DESIGN** | **15** |
| **4.1** | **INTRODUCTION** | **16** |
| **4.2** | **UML DIAGRAM** | **16** |
| **4.2.1** | **USE CASE DIAGRAM** | **17** |
| **4.2.2** | **SEQUENCE DIAGRAM** | **20** |
| **4.2.3** | **CLASS DIAGRAM** | **21** |
| **4.2.4** | **DEPLOYEMENT DIAGRAM** | **22** |
| **4.2.5** | **ACTIVITY DIAGRAM** | **24** |
| **4.2.6** | **OBJECT DIAGRAM** | **26** |
| **4.2.7** | **STATE DIAGARAM** | **27** |
| **4.5** | **USER INTERFACE DESIGN** | **30** |
| **4.6** | **DATA BASE DESIGN** | **33** |
| **5** | **SYSTEM TESTING** | **41** |
| **5.1** | **INTRODUCTION** | **42** |
| **5.2** | **TEST PLAN** | **43** |
| **5.2.1** | **UNIT TESTING** | **43** |
| **5.2.2** | **INTEGRATION TESTING** | **44** |
| **5.2.3** | **VALIDATION TESTING** | **44** |
| **5.2.4** | **USER ACCEPTANCE TASTING** | **45** |
| **6** | **IMPLEMENTATION** | **46** |
| **6.1** | **INTRODUCTION** | **47** |
| **6.2** | **IMPLEMENTATION PROCEDURE** | **47** |
| **6.2.1** | **USER TRAINING** | **48** |
| **6.2.2** | **TRAINING ON APPLICATION SOFTWARE** | **48** |
| **6.2.3** | **SYSTEM MAINTENANCE** | **48** |
| **7** | **CONCLUSION & FUTURE SCOPE** | **49** |
| **7.1** | **CONCLUSION** | **50** |
| **8** | **BIBLIOGRAPHY** | **51** |
| **9** | **APPENDIX** | **53** |
| **9.1** | **SAMPLE CODE** | **54** |
| **9.2** | **SCREEN SHOTS** | **60** |

**List of Abbreviation**

|  |  |  |
| --- | --- | --- |
| IDE | - | Integrated Development Environment |
| HTML | - | Hyper Text Markup Language. |
| CSS | - | Cascading Style Sheet |
| SQL | - | Structured Query Language |
| UML | - | Unified Modeling Language |

**CHAPTER 1 INTRODUCTION**

* 1. **PROJECT OVERVIEW**

Kerala state youth festival is one of the biggest cultural event in Asia, Keralites are very proud to conduct this festival. Participants from all over Kerala attend the programs. But the existing system of the conducting of youth festival is not so fair. Everything from start to end is being done manually, from declaration to result publishing everything is done as paper and pen form. All the records are stored just in ledger notes which is not a safe way of storing files, because damage of these ledger books may clear all datas of the concerened youth festivals. As the technology has made a vast grow in the world, use of these technologies can be implemented for state youth festival also. My project proposes such a system to make everything form registration to result publication automated.

* 1. **PROJECT SPECIFICATION**

The proposed system is a website for making the state youth festival automated. This system proposes a way of registration to result publication everything can be done online with help of few users. The proposed system contains mainly 4 users whose functionalities have mentioned below:

The system includes 4 modules. They are:

1. **Admin Module**

Admin performs a central control over whole the system. Admin performs all the following functions and took a central control over the whole system.

* Admin login.
* Add/edit staff.
* Add/edit representative.
* Add judge
* Add/edit programs.
* Add/edit districts.
* View schedule.
* View of users.
* View results
* View of all programs registrations.

1. **Staff Module**

Staffs are here to perform the following functions:

* Staff login.
* Schedule programs.
* Enter results.
* View of schedules.
* View of event registrations
* View of results.

1. **Representative module**

Representatives perfoms following functions:

* Rep login
* View/remove registered participant
* View/edit event registrations
* Register for group events
* View schedule
* View result

1. **Participant module**

Participants can do the following functions:

* User login
* Register for events
* View/edit event registrations
* View schedule
* View result

**CHAPTER 2 SYSTEM STUDY**

* 1. **INTRODUCTION**

Critical information is being managed and maintained manually by the state youth festival. The existing method has data stores dispersed throughout the event infrastructure and necessitates several paper forms. On forms, information is typically wrong or doesn't follow management guidelines. Records frequently vanish during computation, so a rigorous auditing system is required to ensure that no important data is lost. There are many versions of the same information in the data for the current state youth festival, which could lead to data conflicts between various data repositories. The timely collection, storage, and retrieval of vast amounts of data is essential to the success of youth festivals. This information typically includes participant personal information as well as a variety of additional details about the individuals working on making the festival happen. So that the organization's resources may be used successfully, all of this information must be managed effectively and economically.

DAKSHA- STATE YOUTH FESTIVAL AUTOMATION SYSTEM's goal is to automate state youth festival management, making it more effective and error-free. By using a highly automated process that is stress-free, dependable, and quick, it aims to standardise data, consolidate data, ensure data integrity, and reduce inconsistencies. It does this by using the PHP programming language and application SQL database for both all users and the staff who are in charge of it. methods for registration and others. While the SQL database will be at the back-end to manage the data storage process, HTML will be at the front-end and give the graphical user interface that interacts with the user.

* 1. **EXISTING SYSTEM**

When running youth festivals, important information is currently managed and preserved manually. The current system requires a lot of paper forms due to the fact that data repositories are spread out throughout the youth festival's architectural design. On forms, information is typically wrong or doesn't follow management guidelines. In order to ensure that no important information is lost, a thorough auditing procedure is required because forms frequently disappear while being moved between departments. First stage of conducting youth festival is registering the participants from all districts.now in the existing system, the concerened authority of each district transfer the datas to state co-rdinators as ledger format hence scheduling marking system result view everything is manually done as ledger form. Therefore these datas and informations are not at all secured and consistent. Datas can be lost anytime if there happends any damages in the ledgers.

* 1. **DRAWBACKS OF EXISTING SYSTEM**
     + No proper online management of system
     + Human effort is needed.
     + It is difficult to maintain important information in ledgers.
     + More manual hours need to generate required reports.
     + Data is not at all secure.
     + Datas are inconsistent.
     + Future references of informations are difficult.
  2. **PROPOSED SYSTEM**

The DAKSHA-STATE YOUTH FESTIVAL AUTOMATION SYSTEM is intended to replace state government's current manual, paper-based system for conducting state youth festivals. The new system, which controls festival information, event information, participant information, results, timetables, etc., takes the shape of a registration system. In order to shorten the time and resources now needed to hold a state youth festival, these services must be delivered effectively and economically. Due to the time-consuming and laborious administrative processes used by the current system, complete information cannot be preserved for the future. This automated system focuses on enhancing its efficiency and reducing human error. It seeks to standardise data, consolidate data, assure data integrity, and decrease discrepancies through the use of a fully automated procedure that is stress-free, dependable, and rapid.

* 1. **ADVANTAGES OF PROPOSED SYSTEM**

The system is relatively easy to implement and design. The system works in practically all settings and uses very little system resources. First and mostly its advantage is that it is very simple to use even for a common user who has only less technical knowledge. All the datas used in this website is stored permenantly for future references. The datas and other informations are secured means unauthorized access is protected in the website,

users with correct login credentials can only be accessed in this site. So that the datas are protected from various forms of destructions.

Almost all the datas entered in the system gets validated at the time of entering itself so that the system eliminates all the manual errors which makes the data accurate. This system helps to avoid a lot of paper works and use of such resources to store a vast volume of datas and other information regarding conducting state youth festival. While conducting a state youth festival many datas need to be collected and stored, use of the existing system need many resources to collect these informations and to store it.

Therefore it serves a better service which reduces or avoids all such hard copies which is highly difficult to manage as it is in manual form. We can also conserve the time and human resources for doing the same tasks. The data can be maintained for longer period with no loss of data.

**CHAPTER 3 REQUIREMENT ANALYSIS**

* 1. **FEASIBILITY STUDY**

To determine if the project will achieve the goals of the organisation given the amount of work, effort, and time invested in it, a feasibility study is carried out. The developer can predict the project's future and usefulness thanks to the feasibility study. A system proposal's viability is determined by its influence on the organisation, ability to satisfy the needs of its users, and efficient use of resources. As a result, before a new application is accepted for development, it normally undergoes a feasibility assessment.

The document provides the feasibility of the project being designed and lists several areas that were carefully considered during the feasibility study of this project, such as technical, economic and operational feasibility. Following are its features:-

* + 1. **Economical Feasibility**

Cost and benefit analyses are required to support the emerging system. criteria to make sure that focus is placed on the project that will yield the best results the earliest. The price that would be involved in developing a new system is one of the variables.

Some significant financial queries raised during the initial probe include the following:

* + - * The expenses carry out an extensive system investigation.
      * The price of the software and hardware.
      * The advantages in terms of lower expenses or fewer expensive mistakes.

The proposed system was created as part of a project; hence, there are no manual expenses associated with it. Additionally, the fact that all the resources are currently available shows that the system can be developed economically.

The cost of project, DAKSHA-STATE YOUTH FESTIVAL AUTOMATION SYSTEM was divided according to the system used, its development cost and cost for hosting the project. According to all the calculations the project was developed in a low cost. As it is completely developed using open source software.

* + 1. **Technical Feasibility**

First, a technical evaluation of the system is required. For the assessment of this feasibility, a design strategy of the system's requirements in terms of input, output, programmes, and procedures must be used as the basis. Further investigation is required once a system's design has been established in order to suggest the kind of tools, the process required to develop the system, and how to operate the system once it has been built.

Technical issues raised during the investigation are:

* + - * Does the suggested technology work with the current technology?
      * Can the system expand if developed?

The project must be created in a way that achieves the desired features and performance while adhering to the limitations. The project requires a high resolution scanning device and uses cryptographic techniques. The system can still be used even though the technology may eventually become outdated because the current version of the same software is backward compatible. Therefore, this project only has a few limitations. The system was created using PHP for the front end and a MySQL server for the back end; it is technically feasible to complete the project. The system has been developed using PHP in front end and MySQL in server in back end, the project is technically feasible for development. The System used was also of good performance of Processor Intel i3 core; RAM 4GB and, Hard disk 1TB

* + 1. **Behavioral Feasibility**

The proposed system includes the following questions:

* + - * Is there enough support for users?
      * Will the proposed system cause any harm?

When the project is built and installed, it will be advantageous since it achieves the goals. The project is determined to be behaviorally feasible after carefully weighing all behavioural factors.

* 1. **SYSTEM SPECIFICATION**
     1. **Hardware Specification**

Processor - Intel core i5

RAM - 4GB

Hard disk - 1 TB

* + 1. **Software Specification**

Front End - HTML, CSS

Backend - MYSQL

Client on PC - Windows 7 and above.

Technologies used - JS, HTML5, BOOTSTARP, J Query, PHP, CSS

* 1. **SOFTWARE DESCRIPTION**
     1. **PHP**

PHP is a server-side scripting language used for both web development and general-purpose programming. More than 244 million websites and 2.1 million web servers currently use PHP. The PHP group now produces the reference implementation of PHP, which was first developed by Rasmus Ledorf in 1995. PHP, a recursive acronym that once meant for personal Home page, now stands for PHP: HypertextPreprocessor. A web server's PHP processor module interprets PHP code to produce the final web page. Instead of calling an external file, PHP commands can be directly put into an HTML source file to handle data. Due to limitations on the use of the name PHP, it has also developed to include a command-line interface capability and can be used standalone, which makes it incompatible with the GNU General Public License (GPL). Most web servers support the free deployment of PHP, which is also available as a standalone shell on practically all platforms and operating systems.

* + 1. **MySQL**

Oracle Corporation created, distributed, and provided support for MySQL, the most well-known Open Source SQL database management system. The most recent details regarding MySQL software are available on the MySQL website..

* **MySQL is a database management system.**

A database is a planned collection of data. Considerations could range from a simple grocery list to an image gallery or the vast volume of data on a business network. To add, access, and process data that is stored in a computer database, you need a database management system like MySQL Server. Because computers are excellent at processing enormous volumes of data, whether as standalone utilities or as a component of other programmes, database management systems are crucial to computing.

* **MySQL databases are relational.**

Instead of placing all the data in one huge warehouse, a relational database keeps the data in individual tables. Physical files that are optimised for speed contain database structures. The logical model provides a flexible programming environment with objects like databases, tables, views, rows, and columns. Establish guidelines for the one-to-one, one-to-many, unique, required, or optional relationships between various data fields as well as "points" between various tables. With a well-designed database, your application will never see inconsistent, duplicate, orphaned, out-of-date, or missing data since the database enforces these rules. MySQL stands for "Structured Query Language" with the SQL prefix. The most popular standard language for accessing databases is SQL. You can either enter SQL directly (for instance, to generate reports), incorporate SQL statements into code written in another language, or use a language-specific API that conceals SQL syntax, depending on your programming environment. The ANSI/ISO SQL standard specifies SQL. There are various versions of the SQL standard, which has been developing since 1986. The 1992 standard, "SQL92," is referred to in this document.

The 1999 version of the standard is referred to as "SQL: 1999," and the most recent version is referred to as "SQL: 2003." The SQL Standard as it exists at any one time is referred to as "the SQL standard."

* **MySQL software is Open Source*.***

Anyone can use and modify software that is open source. Anyone can use MySQL software for free by downloading it from the Internet. You are free to examine the source code and adjust it as necessary. The GPL (GNU General Public License) is used by MySQL software to specify what you may and cannot do with the programme in various circumstances. You can get a commercially licenced version from us if you don't trust the GPL or need to integrate MySQL code into a business application. For further details, see the MySQL Licensing Overview..

* **The MySQL Database Server is very fast, reliable, scalable, and easy to use.**

You ought to give it a shot if that is what you're after. In addition to your other apps, web servers, and other software, MySQL Server can function smoothly on a desktop or laptop while requiring little to no maintenance. You can modify the settings to utilise all the RAM, CPU power, and I/O capacity if you dedicate an entire machine to MySQL.

* **MySQL Server works in client/server or embedded systems*.***

The client/server system known as the MySQL Database Software is composed of a multi-threaded SQL server, a number of client programmes and libraries, management tools, and a wide range of application programming interfaces (APIs). Additionally, you can combine MySQL Server's built-in multi-threaded library into your software to create a standalone solution that is smaller, quicker, and simpler to use.

**CHAPTER 4 SYSTEM DESIGN**

* 1. **INTRODUCTION**

Any engineered system or product's development process begins with design. A creative process is design. The secret to an efficient system is a decent design. The process of using different methodologies and concepts to specify a process or a system in enough detail to allow for its physical implementation is referred to as "design.". The process of using several approaches and concepts to specify a tool, a procedure, or a system in sufficient detail to enable its physical actuality is one way to put it. Software design is the technical centre of the software engineering process, regardless of the development paradigm used. The architectural detail needed to construct a system or product is developed through the system design. This programme has also through the best possible design phase, fine tuning all efficiency, performance, and accuracy levels, as in the case of any systematic technique. A user-oriented document is converted into a document for programmers or database staff throughout the design phase. There are two stages to the development of a system design: Logical and Physical Design.

* 1. **UML DIAGRAM**

UML is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems. UML was created by the Object Management Group (OMG) and UML 1.0 specification draft was proposed to the OMG in January 1997.

UML stands for **Unified Modeling Language**. UML is different from the other common programming languages such as C++, Java, COBOL, etc. UML is a pictorial language used to make software blueprints. UML can be described as a general purpose visual modeling language to visualize, specify, construct, and document software system. Although UML is generally used to model software systems, it is not limited within this boundary. It is also used to model non-software systems as well. For example, the process flow in a manufacturing unit, etc. UML is not a programming language but tools can be used to generate code in various languages using UML diagrams. UML has a direct relation with object oriented analysis and design. After some standardization, UML has become an OMG standard. A comprehensive UML diagram that depicts a system is made up of all the elements and relationships. The most crucial aspect of the entire procedure is the UML diagram's aesthetic impact. To make it complete, all the additional components are used. UML includes the following nine diagrams.

* Class diagram
* Object diagram
* Use case diagram
* Sequence diagram
* Collaboration diagram
* Activity diagram
* Statechart diagram
* Deployment diagram
* Component diagram

**4.2.1 USE CASE DIAGRAM**

A use case diagram is a visual representation of the interactions between system components. A use case is a method for locating, defining, and organising system needs. The word "system" in this context refers to a thing that is being built or operated, such as a website for mail-order service and product sales. Use case diagrams are used in UML (Unified Modeling Language), a standard language for modelling real-world objects and systems.

The planning of general requirements, the validation of a hardware design, the testing and debugging of a software product in development, the creation of an online help reference, or the completion of a job focused on customer support are all examples of system objectives. For instance, use cases in a product sales context can involve customer service, item ordering, catalogue updating, and payment processing. There are four elements in a use case diagram.

* The boundary, which defines the system of interest in relation to the world around it.
* The actors, usually individuals involved with the system defined according to their roles.
* The use cases, which are the specific roles are played by the actors within and around the system.
* The relationships between and among the actors and the use cases.

Use case diagrams are drawn to capture the functional requirements of a system. After identifying the above items, we have to use the following guidelines to draw an efficient use case diagram

* The name of a use case is very important. The name should be chosen in such a way so that it can identify the functionalities performed.
* Give a suitable name for actors.
* Show relationships and dependencies clearly in the diagram.
* Do not try to include all types of relationships, as the main purpose of the diagram is to identify the requirements.
* Use notes whenever required to clarify some important points.

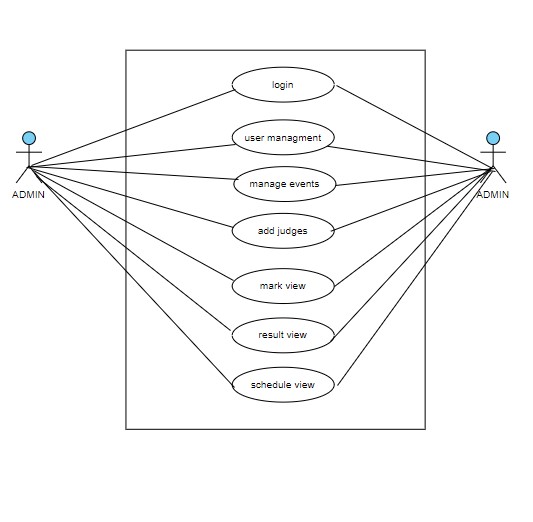


FIG 1.Use case diagram(Admin)

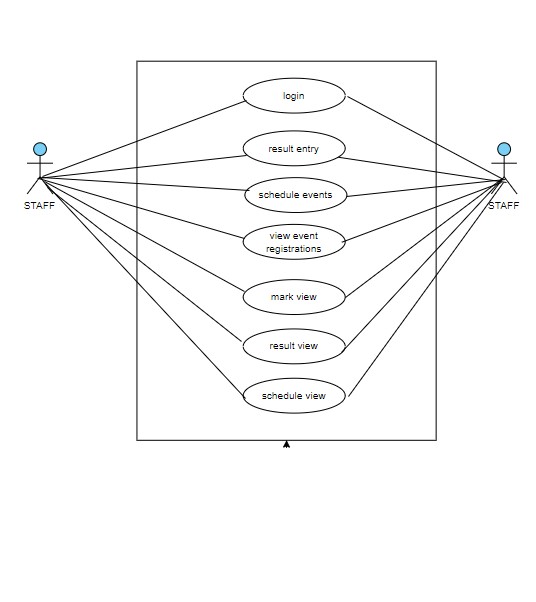


FIG 1.1 Use case diagram(Staff)

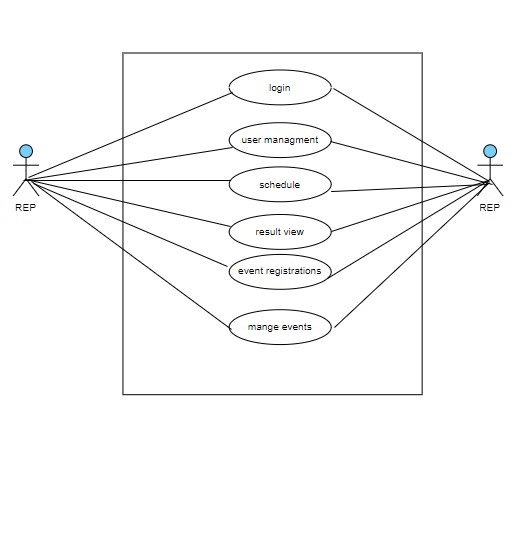


FIG 1.2 Use case diagram(District Rep)

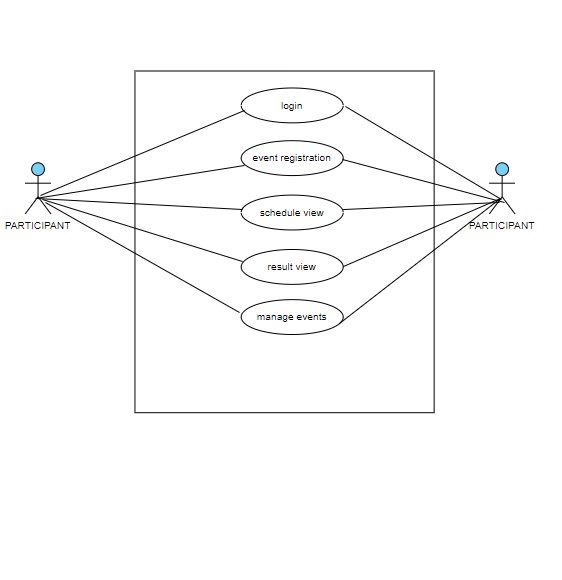


FIG 1.3 Use case diagram(Participant)

**4.2.2 SEQUENCE DIAGRAM**

A sequence diagram essentially shows how things interact with one another sequentially, or the order in which these interactions occur. A sequence diagram can also be referred to as event diagrams or event scenarios. Sequence diagrams show the actions taken by the components of a system in chronological order. These diagrams are widely used by businesspeople and software engineers to document and explain the requirements for new and existing systems.

**Sequence Diagram Notations –**

1. **Actors –** In a UML diagram, an actor represents a particular kind of role in which it communicates with the system's objects. An actor is always beyond the purview of the system that we want to use the UML diagram to represent. We employ actors to portray a variety of roles, including those of human users and other outside subjects. In a UML diagram, an actor is represented using a stick person notation. In a sequence diagram, there might be several actors.
2. **Lifelines –** A lifeline is a named element which depicts an individual participant in a sequence diagram. So basically each instance in a sequence diagram is represented by a lifeline. Lifeline elements are located at the top in a sequence diagram
3. **Messages –** Communication between objects is depicted using messages. The messages appear in a sequential order on the lifeline. We represent messages using arrows. Lifelines and messages form the core of a sequence diagram.

Messages can be broadly classified into the following categories:

* Synchronous messages
* Asynchronous Messages
* Create message
* Delete Message
* Self-Message
* Reply Message
* Found Message
* Lost Message

1. **Guards –** In the UML, guards are used to model circumstances. When we need to limit the flow of messages under the guise of a condition being met, we use them. Software engineers rely on guards to inform them of the limitations imposed by a system or specific process.

**Uses of sequence diagrams :**

* Used to model and visualize the logic behind a sophisticated function, operation or procedure.
* They are also used to show details of UML use case diagrams.
* Used to understand the detailed functionality of current or future systems.
* Visualize how messages and tasks move between objects or components in a system.

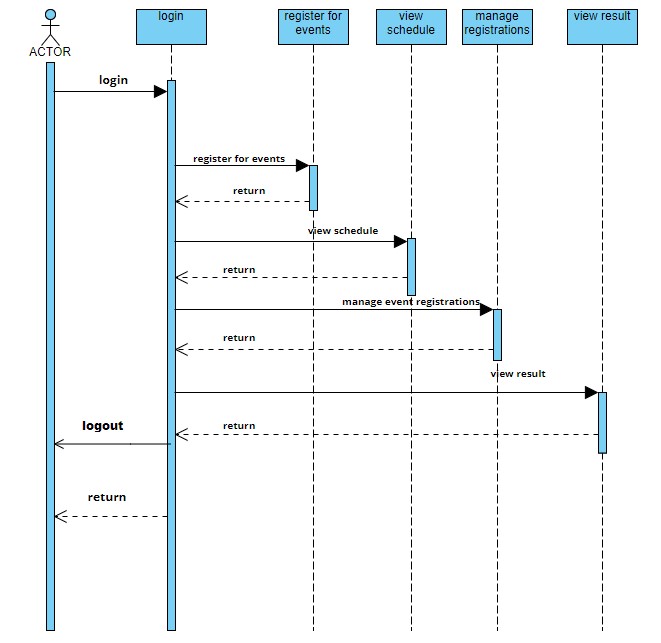


FIG 2.0 Sequence diagram(admin)

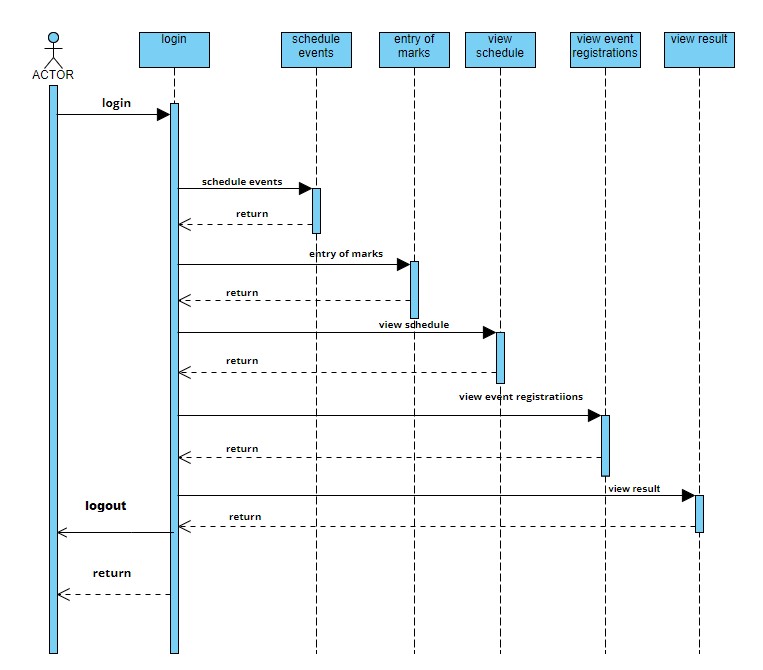


FIG 2.1 Sequence diagram(staff)

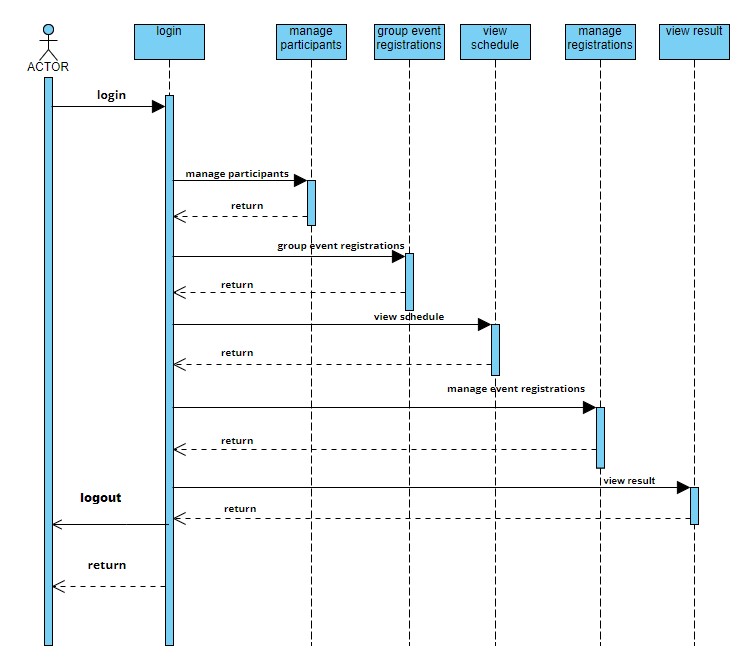


FIG 2.2 Sequence diagram for district representative

**4.2.3 CLASS DIAGRAM**

The fundamental units of every object-oriented methodology are class diagrams. The classes, relationships, interface, affiliation, and collaboration can all be displayed using the class diagram. Class diagrams adhere to UML standards. The class diagram has an ideal structure to depict the classes, inheritance, relationships, and everything else that OOPs have in their context because classes are the foundation of an OOPs-based programme. It describes several object types and their static relationships.

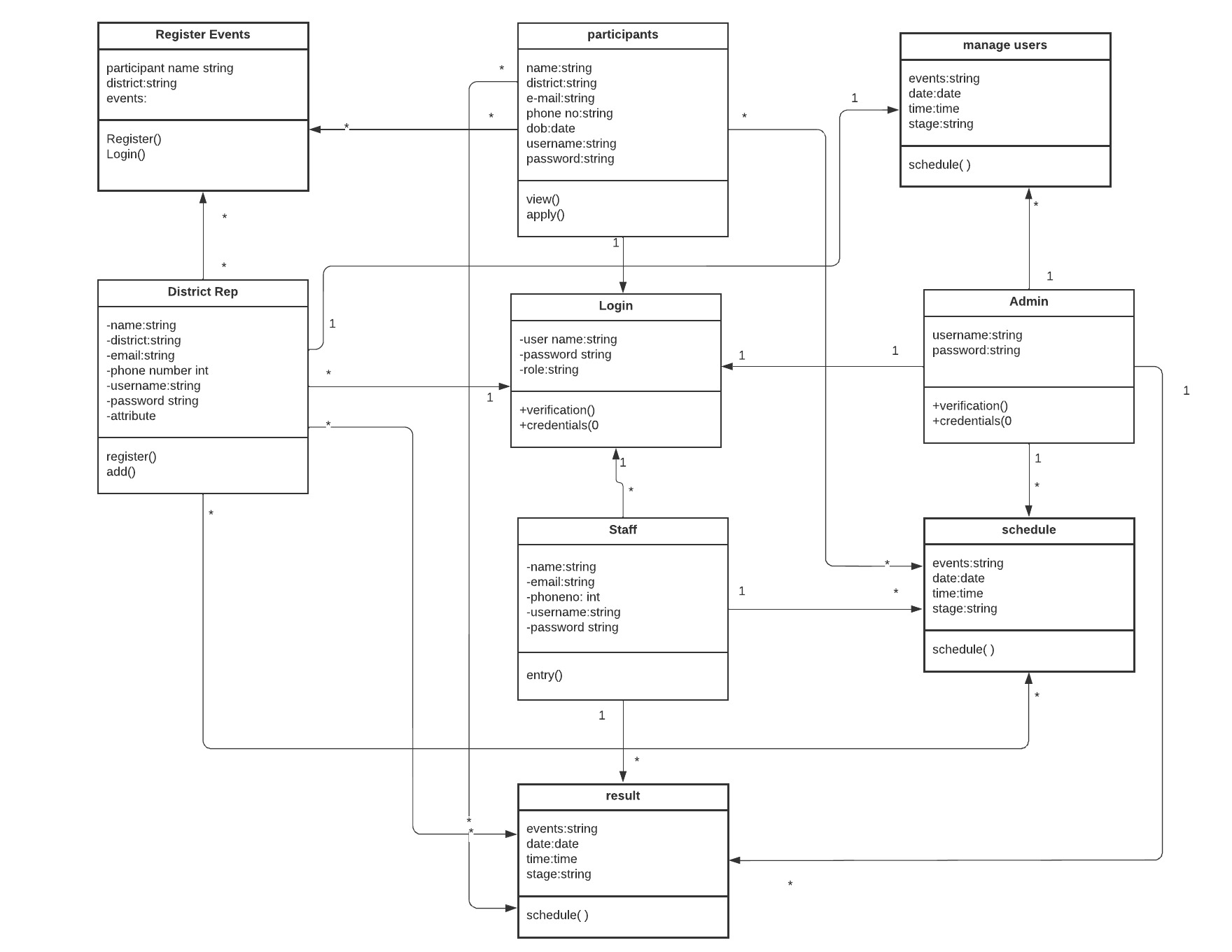


FIG 3. Class diagram for Daksha-State Youth Festival Automation System

**4.2.4 DEPLOYEMENT DIAGRAM**

A deployment diagram, a type of UML diagram, illustrates the execution architecture of a system, which consists of nodes such as hardware or software execution environments and the middleware connecting them. Deployment diagrams are frequently used to show a system's actual hardware and software. You can understand how the hardware will actually provide the system by utilising it. Deployment diagrams help represent the physical structure of a system, in contrast to other UML diagram types that primarily show the logical components of a system.

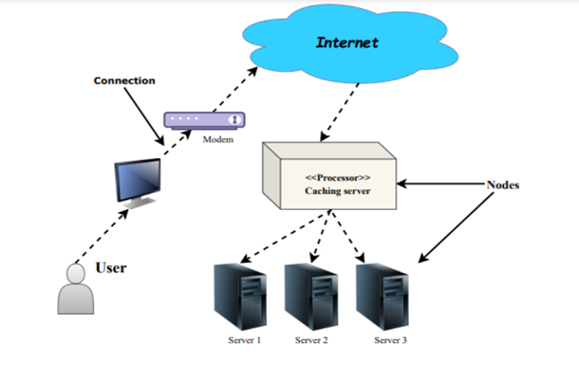


FIG 4. Deployement diagram for Daksha-State Youth Festival Automation System

**4.2.5 ACTIVITY DIAGRAM**

We refer to the stages taken in the execution of a use case and the control flow in a system in Activity Diagrams. Using activity diagrams, we model both sequential and concurrent activities. So, using an activity diagram, we simply visualise workflows. An activity diagram emphasises the state of flow and the order in which it occurs. Using an activity diagram, we can explain or show what leads to a specific event.

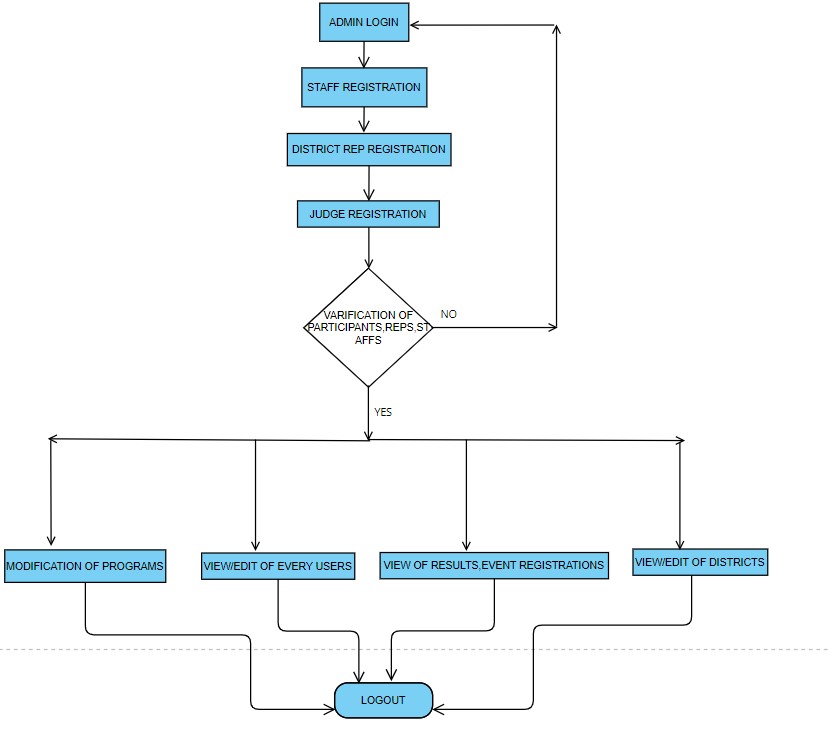
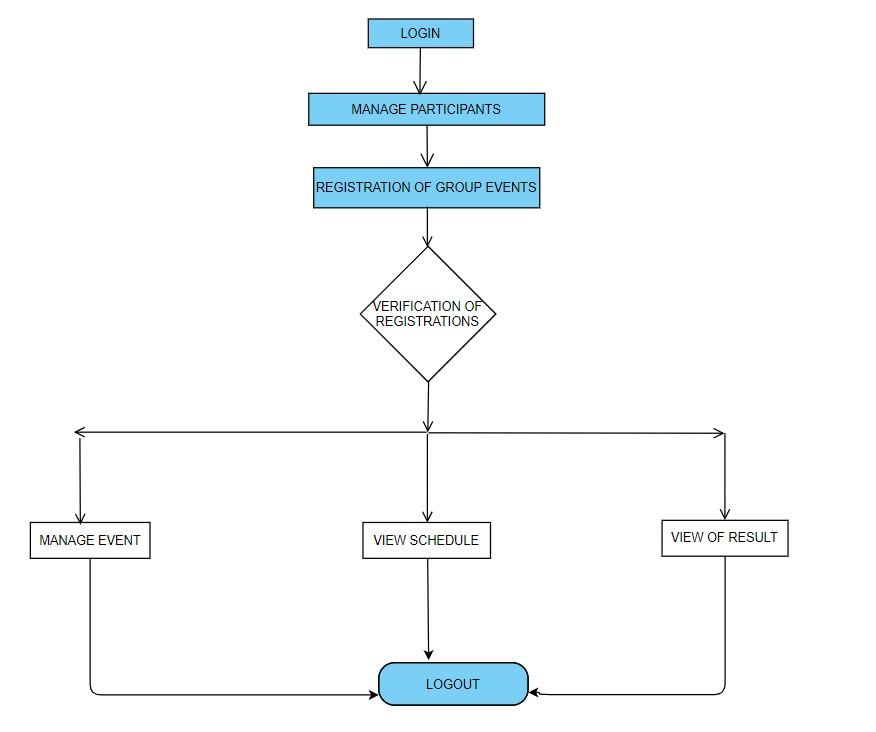


FIG 5.0 Activity diagram for Daksha-State Youth Festival Automation System (Admin)



yes

no

FIG 5.1 Activity diagram for Daksha-State Youth Festival Automation System(representative)

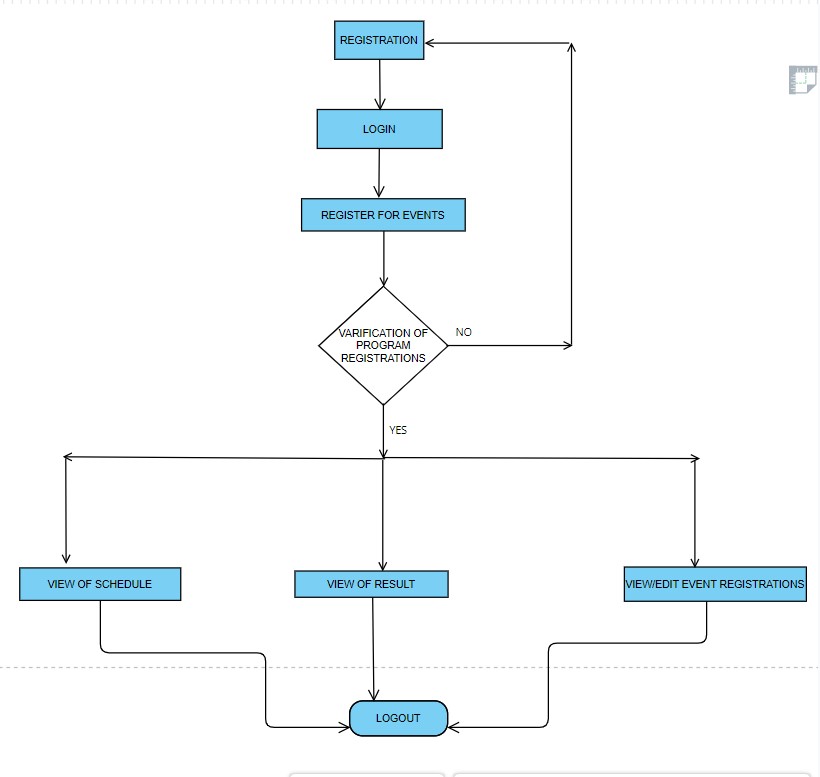
****

FIG 5.2 Activity diagram for Daksha-State Youth Festival Automation System(participant)

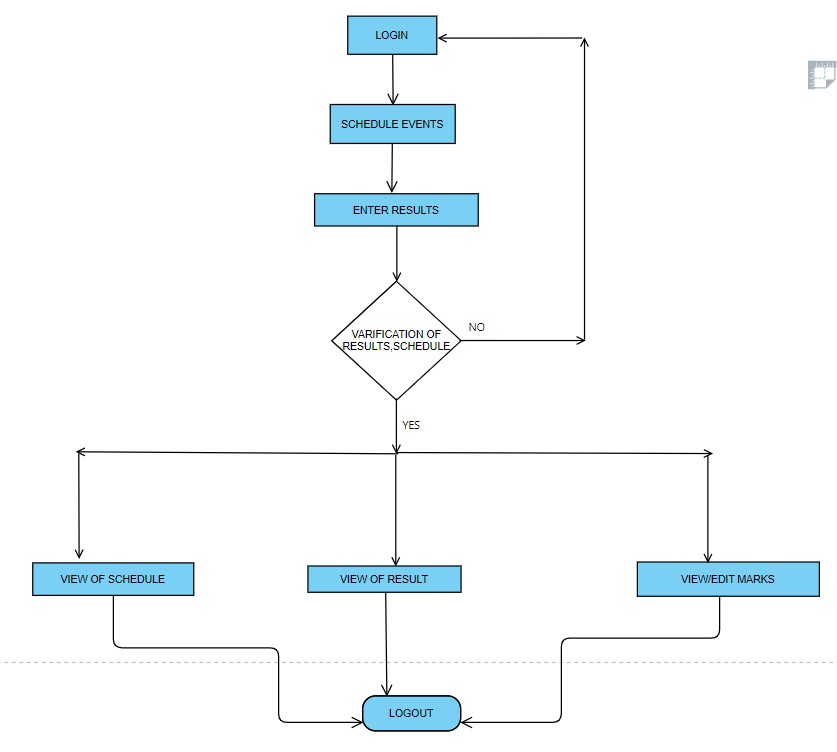
****

FIG 5.3 Activity diagram for Daksha-State Youth Festival Automation System(staff)

**4.2.6 OBJECT DIAGRAM**

Because they are derived from class diagrams, object diagrams are dependent on them. It depicts a particular class diagram instance. The objects aid in presenting a static representation of an object-oriented system at a certain point in time. The main distinction between the object and class diagrams is that the class diagram presents an abstract perspective of a system. It aids in the visualisation of a specific system function..

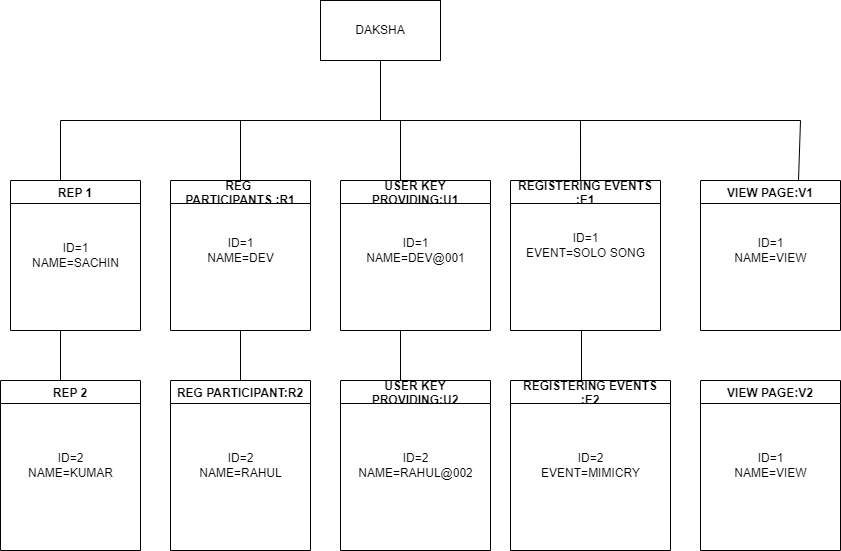


FIG 6. Object diagram for Daksha-State Youth Festival Automation System

**4.2.7 STATE DIAGRAM**

A state diagram is used to depict the status of a system or a component of a system at specific points in time. It is a behavioural diagram that uses finite state transitions to depict the behaviour. State machines and state-chart diagrams are other names for state diagrams. These words are frequently used in the same sentence. Simply said, a state diagram is used to represent how a class will behave dynamically over time and in reaction to changing external stimuli. While we can claim that every class has a state, not all classes are modelled using state diagrams. The states with three or more states are the ones we like to model.

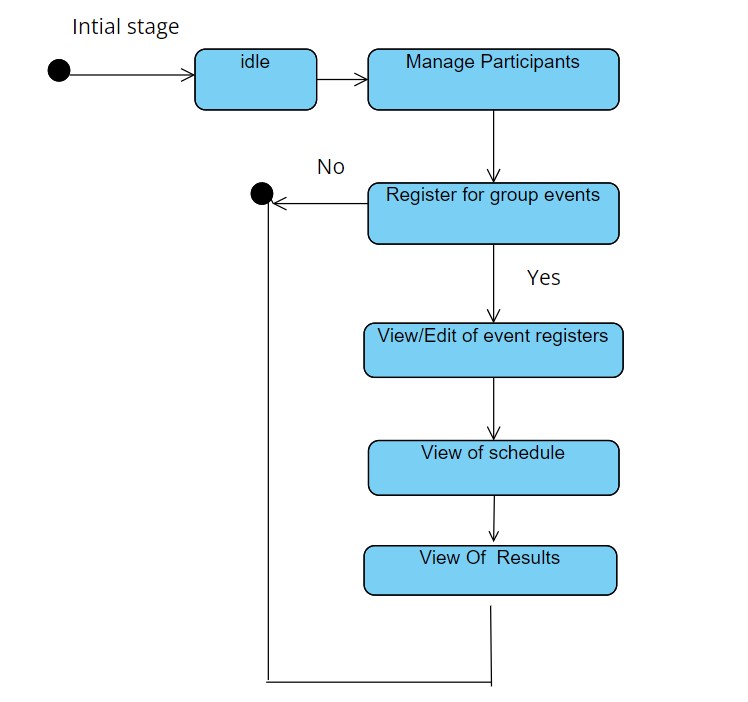


FIG 7.1 Activity diagram (District Representative)

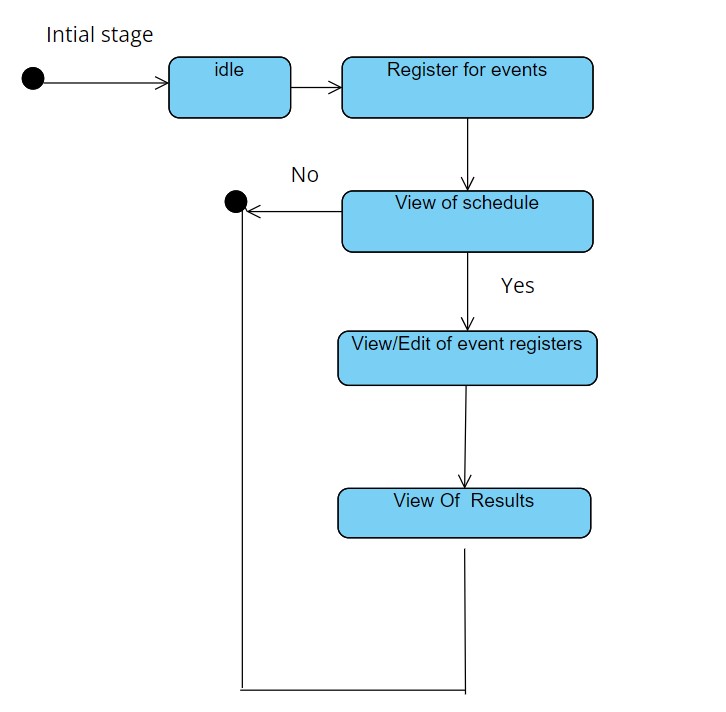


FIG 7.2 Activity diagram (Participant)

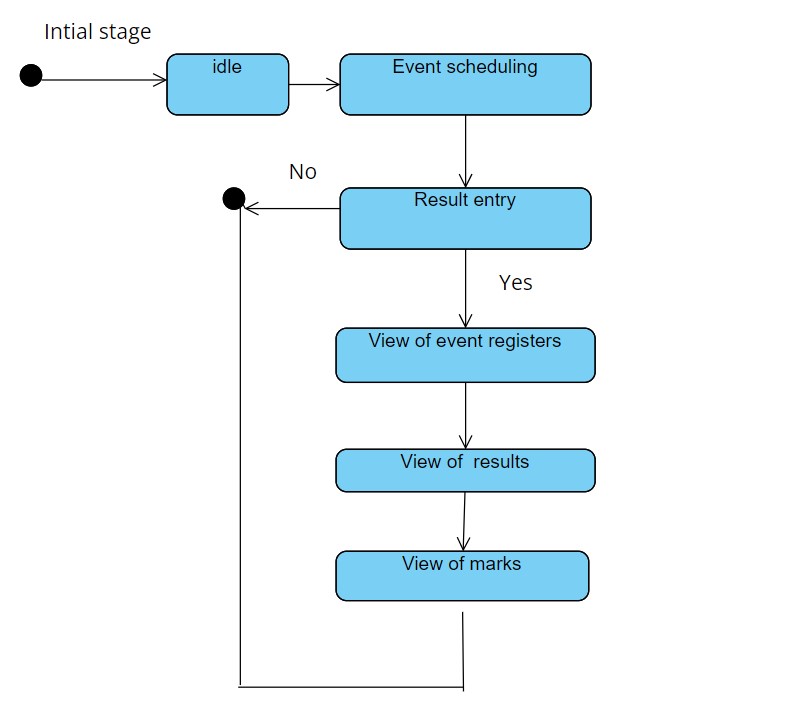
****

FIG 7.3 Activity diagram(staff)

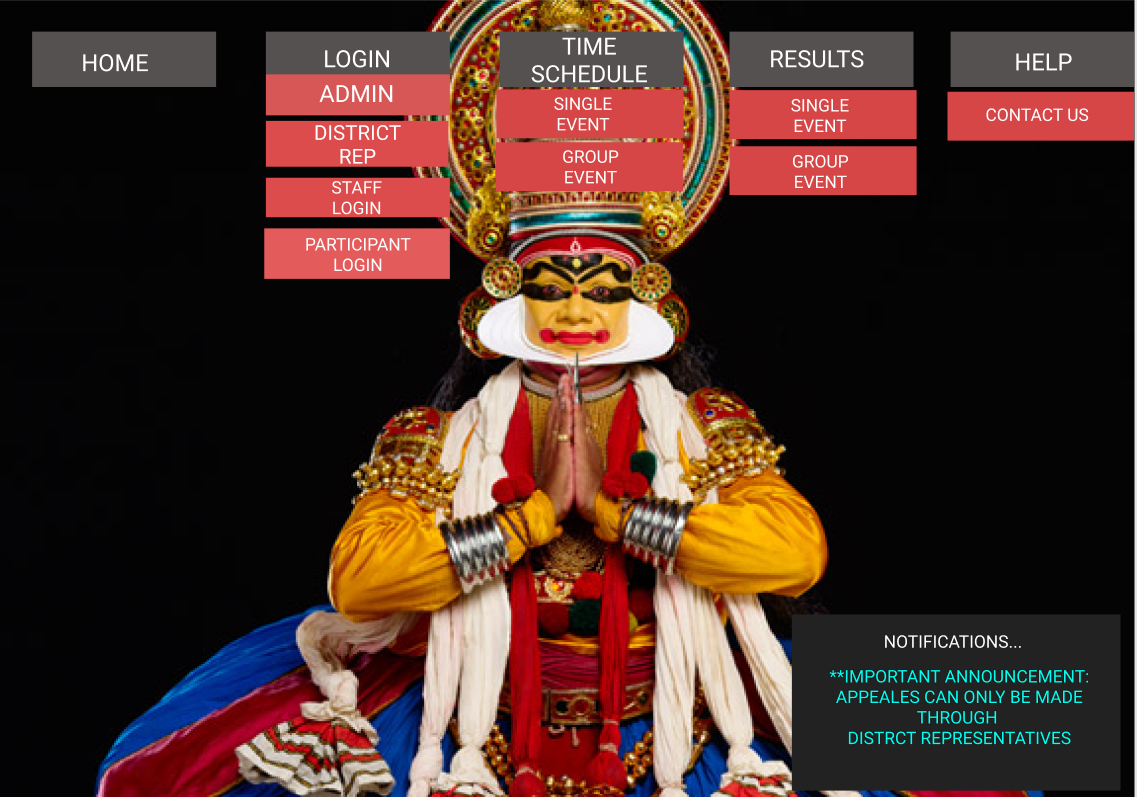
**4.5 USER INTERFACE DESIGN**

**4.5.1 INPUT DESIGN**

**Login**

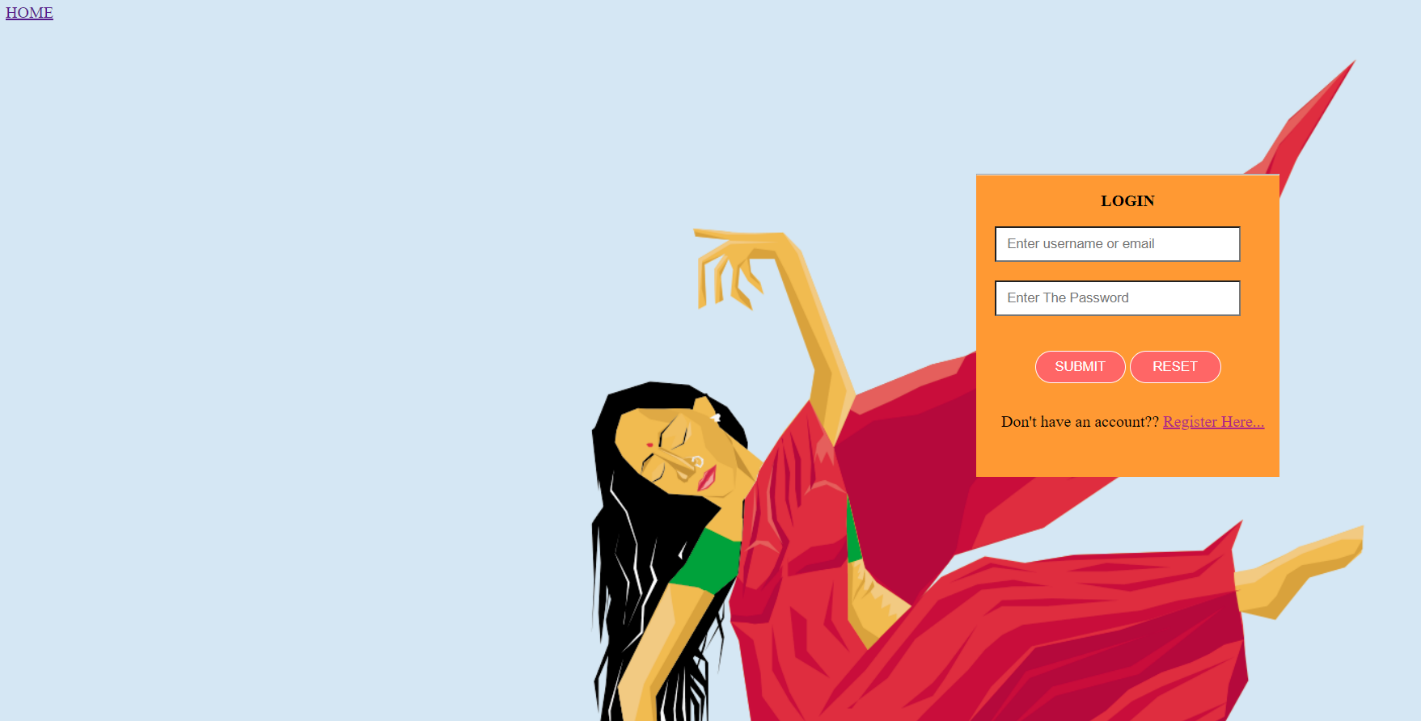
****

**home**

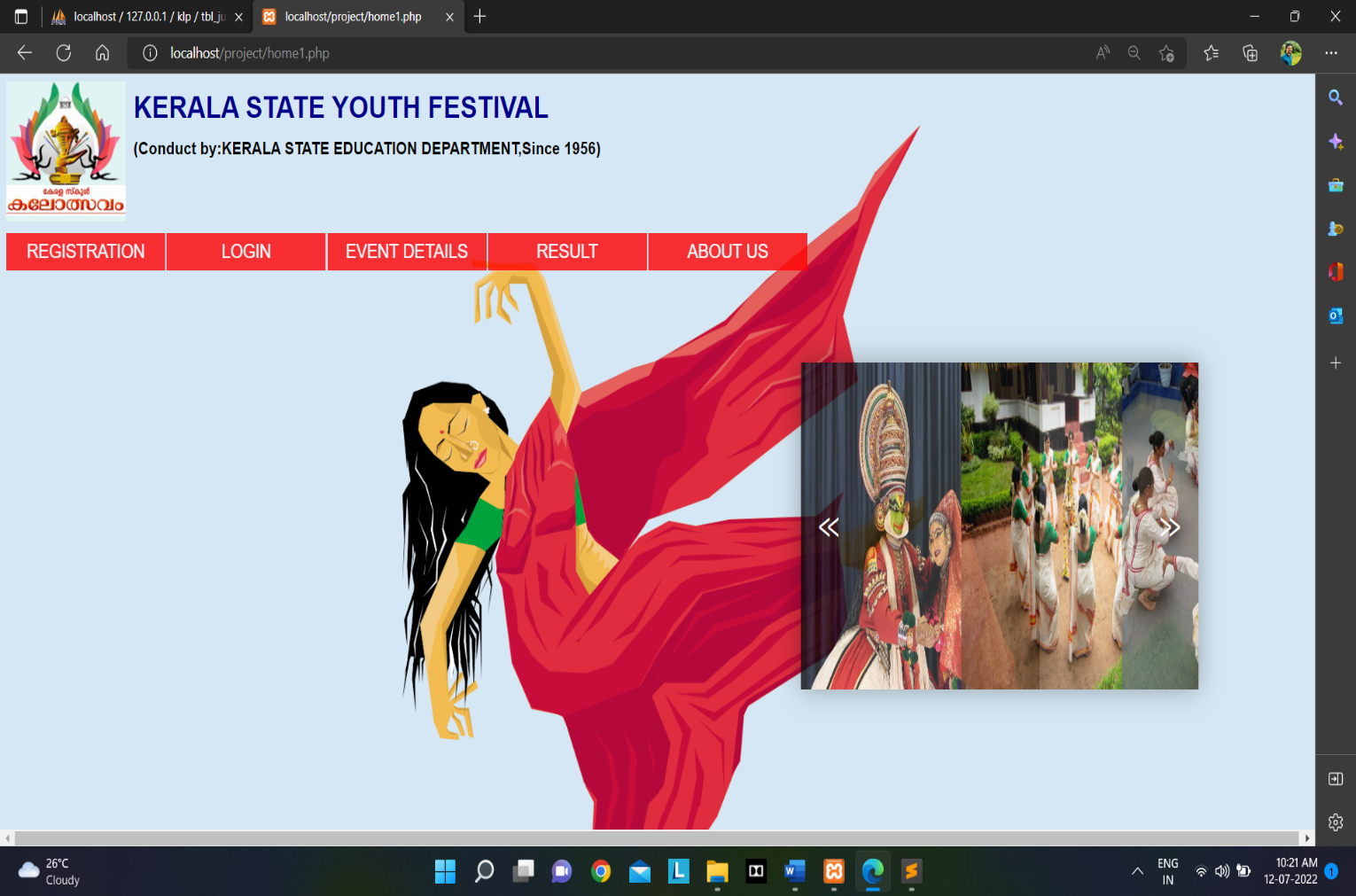
****

**4.5.2 OUTPUT DESIGN**

**User Login**

****

**User Registration**



* 1. **DATABASE DESIGN**

A database is a structured system with the capacity to store information and allows users to retrieve stored information quickly and effectively. The goal of every database is the data, which must be safeguarded.

There are two stages to the database design process. The user needs are obtained in the first step, and a database is created to as clearly as possible meet these criteria. This process, known as information level design, is carried out independently of all DBMSs.

The design for the specific DBMS that will be used to construct the system in issue is converted from an information level design to a design in the second stage. Physical Level Design is the stage where the characteristics of the particular DBMS that will be used are discussed. Parallel to the system design is a database design. The organization of the data in the database is aimed to achieve the following two major objectives.

* + - Data Integrity
    - Data independence
    1. **Relational Database Management System (RDBMS)**

The database is represented as a collection of relations in a relational paradigm. A table or file of records with values can be compared to each relation. In formal relational model terminology, a row is referred to as a tuple, a column heading is referred to as an attribute, and the table is referred to as a relation. Numerous tables, each with a unique name, make up a relational database. A row in a tale represents a collection of related values.

**Relations, Domains & Attributes**

A relation is a table. Tuples are the units of a table's rows. An ordered group of n elements is a tuple. Attributes are referred to as columns. Every table in the database has relationships already established between them. This guarantees the integrity of both referential and entity relationships. A group of atomic values make up a domain D. Choosing a data type from which the domain's data values are derived is a typical way to define a domain. To make it easier to understand the values of the domain, it is also helpful to give it a name.

Every value in a relationship is atomic, meaning it cannot be broken down.

**Relationships**

* Table relationships are established using Key. The two main keys of prime importance are Primary Key & Foreign Key. Entity Integrity and Referential Integrity Relationships can be established with these keys.
* Entity Integrity enforces that no Primary Key can have null values.
* Referential Integrity enforces that no Primary Key can have null values.
* Referential Integrity for each distinct Foreign Key value, there must exist a matching Primary Key value in the same domain. Other key are Super Key and Candidate Keys.
  + 1. **Normalization**

The simplest possible grouping of data is used to put them together so that future changes can be made with little influence on the data structures. The formal process of normalising data structures in a way that reduces duplication and fosters integrity. Using the normalisation technique, superfluous fields are removed and a huge table is divided into several smaller ones. Anomalies in insertion, deletion, and updating are also prevented by using it. Keys and relationships are two notions used in the standard form of data modelling. A row in a table is uniquely identified by a key. Primary keys and foreign keys are the two different kinds of keys. A primary key is an element, or set of components, in a table that serves as a means of distinguishing between records from the same table. A column in a table known as a foreign key is used to uniquely identify records from other tables. Up to the third normal form, all tables have been normalised.

As the name implies, it refers to arranging things in their natural state. The application developer uses normalisation to provide a logical grouping of the data into suitable tables and columns so that users can immediately connect names to the data. Normalization avoids data redundancy, which imposes a severe burden on the computer's resources by removing repeated groupings from the data. These include:

* + - * Normalize the data.
      * Choose proper names for the tables and columns.
      * Choose the proper name for the data.

**First Normal Form**

The First Normal Form states that the domain of an attribute must include only atomic values and that the value of any attribute in a tuple must be a single value from the domain of that attribute. In other words 1NF disallows “relations within relations” or “relations as attribute values within tuples”. The only attribute values permitted by 1NF are single atomic or indivisible values. The first step is to put the data into First Normal Form. This can be donor by moving data into separate tables where the data is of similar type in each table. Each table is given a Primary Key or Foreign Key as per requirement of the project. In this we form new relations for each non-atomic attribute or nested relation. This eliminated repeating groups of data. A relation is said to be in first normal form if only if it satisfies the constraints that contain the primary key only.

**Second Normal Form**

According to Second Normal Form, for relations where primary key contains multiple attributes, no non-key attribute should be functionally dependent on a part of the primary key. In this we decompose and setup a new relation for each partial key with its dependent attributes. Make sure to keep a relation with the original primary key and any attributes that are fully functionally dependent on it. This step helps in taking out data that is only dependent on a part of the key. A relation is said to be in second normal form if and only if it satisfies all the first normal form conditions for the primary key and every non-primary key attributes of the relation is fully dependent on its primary key alone.

**Third Normal Form**

According to Third Normal Form, Relation should not have a non-key attribute functionally determined by another non-key attribute or by a set of non-key attributes. That is, there should be no transitive dependency on the primary key. In this we decompose and set up relation that includes the non-key attributes that functionally determines other non-key attributes. This step is taken to get rid of anything that does not depend entirely on the Primary Key. A relation is said to be in third normal form if only if it is in second normal form and more over the non key attributes of the relation should not be depend on other non-key attribute.

**TABLE DESIGN**

Table NO: 01

Name:-tb\_users

Primary key:-uid

Foreign key:-login\_id references tbl\_login

|  |  |  |
| --- | --- | --- |
| **Field** | **Data type** | **Description** |
| uid | Int(11) | Primary key |
| fname | Varchar(20) | First name of participant |
| Lname | Varchar(20) | Last name of participant |
| Dob | date | Date of birth of participant |
| gender | Varchar(20) | Gender of participant |
| district | Varchar(20) | District of participant |
| phone no | Int(22) | number |
| login\_id | Varchar(20) | Foreign key |

Table NO: 02

Name:-tbl\_login

Primary key:-login\_id

Foreign key:-utype\_id refernces usertype

|  |  |  |
| --- | --- | --- |
| **Field** | **Data type** | **Description** |
| login\_id | Int(11) | Primary key |
| Email | Varchar(20) | Email of users |
| password | Varchar(20) | Password |
| status | Varchar(20) | Status of user |
| utype\_id | varchar(20) | Foreign key |

Table NO: 03

Name:-tbl\_staff

Primary key:-id

Foreign key:login\_id references tbl\_login

|  |  |  |
| --- | --- | --- |
| **Field** | **Data type** | **Description** |
| id | Int(11) | Primary key |
| fname | Varchar(20) | First name of the staff |
| lname | Varchar(20) | Lastname of the staff |
| phone | Int(11) | Phone number of staff |
| login\_id | Int(11) | Foreign key |

Table NO: 04

Name:-tbl\_rep

Primary key:rid

Foreign key:-login\_id references tbl\_login

|  |  |  |
| --- | --- | --- |
| **Field** | **Data type** | **Description** |
| rid | Int(11) | Primary key |
| fname | Varchar(20) | First name of rep |
| lname | Varchar(20) | Last name of the rep |
| d\_id | Varchar(20) | District id of rep |
| phone | Int(11) | Phone number of rep |
| login\_id | Int(11) | Foreign key |

Table NO: 05

Name:tbl\_sch

Primary key:pid

|  |  |  |
| --- | --- | --- |
| **Field** | **Data type** | **Description** |
| pid | Int(11) | Primary key |
| list | Varchar(50) | Foreign key .to store program names |
| Stage | Varchar(20) | To schedule the stage |
| Date1 | date | To schedule date |
| Time | time | To schedule time |

Table NO: 06

Name:-usertype

Primary key:-utype\_id

|  |  |  |
| --- | --- | --- |
| **Field** | **Data type** | **Description** |
| utype\_id | Int(11) | Primary key |
| user\_type | Varchar(20) | Type user |

Table NO: 07

Name:tbl\_group

Primary key:g\_id

Foreign key:-gname references tbl\_prog

|  |  |  |
| --- | --- | --- |
| **Field** | **Data type** | **Description** |
| g\_id | Int(11) | Primary key |
| gname | Varchar(50) | To store registered group event |
| district | Varchar(20) | To store district |

Table NO: 08

Name:-tbl\_reg

Primary key:-eid

Foreign key:-uid references tb\_users

|  |  |  |
| --- | --- | --- |
| **Field** | **Data type** | **Description** |
| eid | Int(11) | Primary key |
| uid | Varchar(20) | To store participant id |
| a | Varchar(50) | To store registered event 1 |
| b | Varchar(20) | To store registered event 2 |
| c | Varchar(20) | To store registered event 3 |

Table NO: 09

Name:-tbl\_point

Primary key:point\_id

Foreing key:-uid references tb\_users

|  |  |  |
| --- | --- | --- |
| **Field** | **Data type** | **Description** |
| Point\_id | Varchar(20) | To primary key |
| event | Varchar(20) | To store event name |
| uid | Varchar(10) | To store participant id |
| mark | Int(11) | To store participant mark |

Table NO: 10

Name:tbl\_district

Primary key:di\_id

|  |  |  |
| --- | --- | --- |
| **Field** | **Data type** | **Description** |
| di\_id | Int(11) | Primary key |
| district | Varchar(50) | To store districts |

Table NO: 11

Name:tbl\_event

Primary key:e\_id

|  |  |  |
| --- | --- | --- |
| **Field** | **Data type** | **Description** |
| e\_id | Int(11) | Primary key |
| Event | Varchar(50) | To store single events |

Table NO: 12

Name:tbl\_prog

Primary key:pr\_id

|  |  |  |
| --- | --- | --- |
| **Field** | **Data type** | **Description** |
| pr\_id | Int(11) | Primary key |
| Gname | Varchar(50) | To store group events |

Table NO: 13

Name:-tbl\_judge

Primary key:-ju\_id

|  |  |  |
| --- | --- | --- |
| **Field** | **Data type** | **Description** |
| Ju\_id | Int(11) | Primary key |
| fname | Varchar(20) | First name of judge |
| lname | Varchar(20) | Last name of judge |
| phno | Int(22) | To store judge’s phone number |
| email | Varchar(20) | To store email of judge |
| Exp | Varchar(20) | To store expertise area of judge |
| event a | Varchar(20) | To store event 1 to judge |
| event b | Varchar(20) | To store event 2 to judge |
| event c | Varchar(20) | To store event 3 to judge |

**CHAPTER 5 SYSTEM TESTING**

* 1. **INTRODUCTION**

The database is represented as a collection of relations in a relational paradigm. A table or file of records with values can be compared to each relation. In formal relational model terminology, a row is referred to as a tuple, a column heading is referred to as an attribute, and the table is referred to as a relation. Numerous tables, each with a unique name, make up a relational database. A row in a tale represents a collection of related values.

The processes of static analysis and dynamic analysis are additional ones that are frequently related to software testing. Static analysis examines the software's source code, searching for issues and obtaining statistics without actually running the code. Dynamic analysis examines how software behaves while it is running in order to offer data like execution traces, timing profiles, and test coverage data.

Testing is a collection of activities that can be planned ahead of time and carried out in a methodical manner. Testing starts with individual modules and progresses to the integration of the full computer-based system. Nothing is complete without testing, as it is essential to the system's testing goals. Several guidelines can be used as testing goals. They are:

Testing is a process of executing a program with the intent of finding an error.

* + - A good test case is one that has high possibility of finding an undiscovered error.
    - A successful test is one that uncovers an undiscovered error.

If a test is successfully carried out in accordance with the aforementioned aims, it will reveal software bugs. Additionally, testing shows that the software functions seem to be functioning in accordance with the specifications and that the performance requirements seem to have been satisfied.

There are three ways to test program.

* + - For correctness
    - For implementation efficiency
    - For computational complexity

Testing for correctness is meant to ensure that a programme performs exactly as it was intended to. This is much harder than it might initially seem, especially for big programmes.

* 1. **TEST PLAN**

A test plan suggests a number of required steps that need be taken in order to complete various testing methodologies. The activity that is to be taken is outlined in the test plan. A computer programme, its documentation, and associated data structures are all created by software developers. It is always the responsibility of the software developers to test each of the program's separate components to make sure it fulfils the purpose for which it was intended. There is an independent test group (ITG) whose purpose is to address the problems that come from allowing the creator of anything to test it. Testing's precise goals should be laid forth in quantifiable language. The test strategy should include information on the mean time to failure, cost to detect and correct problems, remaining defect density or frequency of occurrence, and test work hours per regression test.

The levels of testing include:

* Unit testing
* Integration Testing
* Data validation Testing
* Output Testing
  + 1. **Unit Testing**

Unit testing concentrates verification efforts on the software component or module, which is the smallest unit of software design. The component level design description is used as a guide when testing crucial control paths to find faults inside the module's perimeter. the level of test complexity and the untested area determined for unit testing. Unit testing is white-box focused, and numerous components may be tested simultaneously. To guarantee that data enters and exits the software unit under test properly, the modular interface is tested. To make sure that data temporarily stored retains its integrity during each step of an algorithm's execution, the local data structure is inspected. To confirm that each statement in a module has been executed at least once, boundary conditions are evaluated. Finally, each path for managing errors is examined.

Before starting any other test, tests of data flow over a module interface are necessary. All other tests are irrelevant if data cannot enter and depart the system properly. An important duty during the unit test is the selective examination of execution pathways. Error circumstances must be foreseen in good design, and error handling paths must be put up to cleanly reroute or halt work when an error does arise. The final step of unit testing is boundary testing. Software frequently fails at its limits.

In the Sell-Soft System, unit testing was carried out by treating each module as a distinct entity and subjecting them to a variety of test inputs. The internal logic of the modules had some issues, which were fixed. Each module is tested and run separately after coding. All unused code was eliminated, and it was confirmed that every module was functional and produced the desired outcome.

* + 1. **Integration Testing**

Integration testing is a methodical approach for creating the program's structure while also carrying out tests to find interface issues. The goal is to construct a programme structure that has been determined by design using unit tested components. The programme as a whole is tested. Correction is challenging since the size of the overall programme makes it challenging to isolate the causes. Once these mistakes are fixed, new ones arise, and the process keeps repeating itself in an apparently unending circle. All of the modules were integrated after unit testing was completed in the system to check for any interface inconsistencies. A distinctive programme structure also developed when discrepancies in programme structures were eliminated.

* + 1. **Validation Testing or System Testing**

The testing process comes to an end here. This involved testing the entire system in its entirety, including all forms, code, modules, and class modules. Popular names for this type of testing include system tests and black box testing.

The functional requirements of the software are the main emphasis of the black box testing approach. That example, using Black Box testing, a software engineer can create sets of input conditions that will fully simulate all of a program's functional requirements.

Errors in data structures or external data access, erroneous or missing functions, interface faults, performance issues, initialization issues, and termination issues are all types of errors that black box testing looks for.

* + 1. **Output Testing or User Acceptance Testing**

User approval of the system under consideration is tested; in this case, it must meet the needs of the company. When developing, the programme should stay in touch with the user and perspective system to make modifications as needed. With regard to the following points, this was done:

* + - * Input Screen Designs,
      * Output Screen Designs,

The aforementioned testing is carried out using a variety of test data. The preparation of test data is essential to the system testing process. The system under investigation is then put to the test using the prepared test data. Errors in the system are once again found during testing, fixed using the methods described above, and logged for use in the future.

## Test cases for a Login Page

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Project Name: Daksha-State Youth Festival Automation System** | | | | | |
| **Login Test Case** | | | | | |
| **Test Case ID:** Fun\_1 | | | **Test Designed By: Muhammed Sonu Ribin KA** | | |
| **Test Priority**  **(Low/Medium/High):** High | | | **Test Designed Date:**18-07-2022 | | |
| **Module Name**: Login Screen | | | **Test Executed By: Mrs.Gloriya Mathew** | | |
| **Test Title:** Verify login with valid username and  password | | | **Test Execution Date:** 19-07-2022 | | |
| **Description:** Test the Login Page | | |  | | |
| **Pre-Condition:** User has valid username and password | | | | | |
| **Step** | **Test Step** | **Test Data** | **Expected**  **Result** | **Actual**  **Result** | **Status (Pass/Fail)** |
| 1 | Navigation to Login Page |  | Login Page should be  display ed | Login page displayed | Pass |
| 2 | Provide | Username :rajitha@gmail.com | User should d be able to Login |  | Pass |
| Valid | User Logged |
| username | in and |
| 3 | Provide Valid  Password | Password:123456 | navigated to  User Dashboard |
| 4 | Click on |  |  |
| Sign In |  |
| button |  |
| 5 | Provide Invalid username orpassword | Username:rajitha@gmail.com  Password: 01234 | User should not be | Message for enter valid email id or  password | Pass |
| 6 | Provide | Username : null |
| Null  username | Password: null | able to | displayed |
| or |  | Login |  |
| Password |  |  |  |
| 7 | Click on |  |  |  |
| Sign In |  |  |
| button |  |  |

**Post-Condition:** User is validated with database and successfully login into account.The Account session details are logged in database.

### Code package

package test;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver; public class login {

public static void main(String[] args) {

System.setProperty("webdriver.chrome.driver","C:\\Users\\nibin\\Downloads\\chromedriver\_win32\\ chromedriver.exe" );

WebDriver driver=new ChromeDriver();

driver.ge[t("htt](http://localhost/hostel/login/login.php)p:/[/localhost/hostel/login/login.php");](http://localhost/hostel/login/login.php) driver.findElement(By.id("your\_name")).sendKeys("irin1"); driver.findElement(By.id("your\_pass")).sendKeys("Ib@ll4600"); driver.findElement(By.id("signin")).click();

String actualUr[l="htt](http://localhost/hostel/Hosteller/index.php)p:/[/localhost/hostel/Hosteller/index.php";](http://localhost/hostel/Hosteller/index.php) String expectedUrl= driver.getCurrentUrl(); if(actualUrl.equalsIgnoreCase(expectedUrl)) { System.out.println("Test passed");

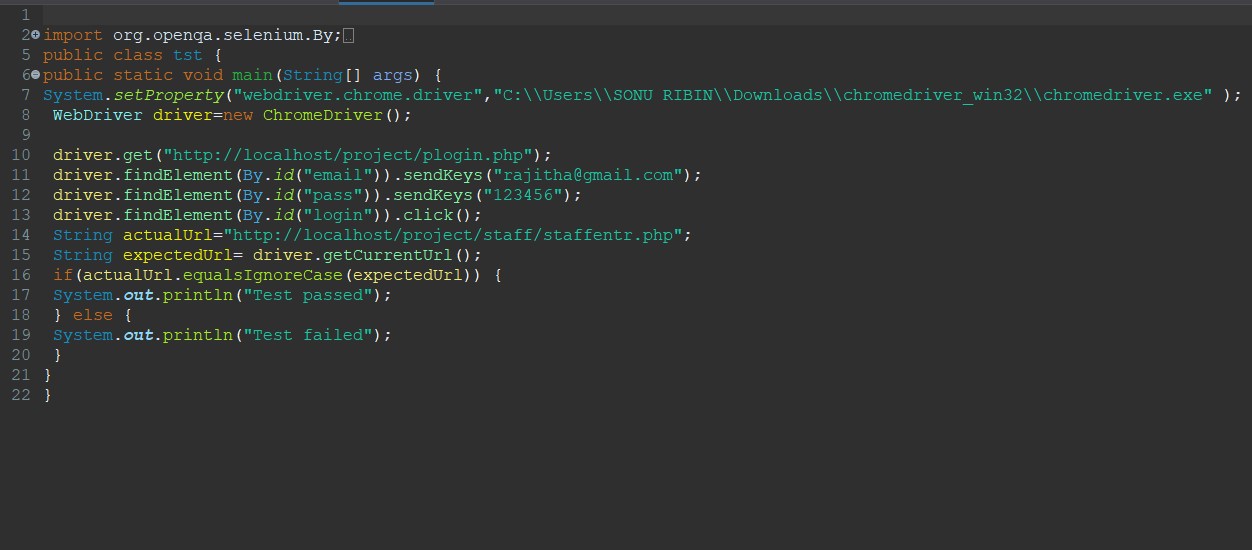
} else {

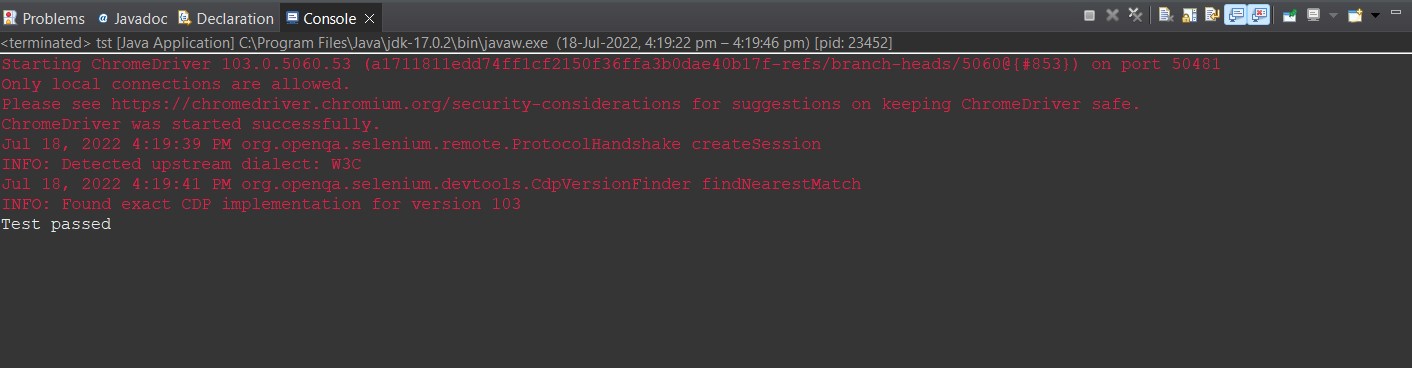
System.out.println("Test failed");

}

}

}





**CHAPTER 6 IMPLEMENTATION**

* 1. **INTRODUCTION**

The project's implementation phase is where the conceptual design is transformed into a functional system. It can be regarded as the most important stage in creating a successful new system since it gives users assurance that the system will operate as intended and be reliable and accurate. User documentation and training are its main concerns. Usually, conversion happens either during or after the user's training. Implementation is the process of turning a newly revised system design into an operational one, and it simply refers to placing a new system design into operation.

The user department now bears the most of the workload, faces the most disruption, and has the biggest influence on the current system. If the implementation is not well thought out or managed, confusion and mayhem may result.

Implementation encompasses all of the steps used to switch from the old system to the new one. The new system could be entirely different, take the place of an existing manual or automated system, or it could be modified to work better. A reliable system that satisfies organisational needs must be implemented properly. System implementation refers to the process of actually using the built system. This comprises all the processes involved in switching from the old to the new system. Only after extensive testing and if it is determined that the system is operating in accordance with the standards can it be put into use. The system personnel assess the system's viability. The system analysis and design work needed to implement the three key components of education and training, system testing, and changeover will increase in complexity as a system is implemented.

The implementation state involves the following tasks:

 Careful planning.

 Investigation of system and constraints.

Design of methods to achieve the changeover.

* 1. **IMPLEMENTATION PROCEDURES**

Software implementation refers to the complete installation of the package in its intended environment, as well as to the system's functionality and satisfaction of its intended applications. The software development project is frequently commissioned by someone who will not be using it. People have early reservations about the software, but we must watch out that they do not become more resistant by making sure that:

 The active user must be aware of the benefits of using the new system.  Their confidence in the software is built up.

 Proper guidance is imparted to the user so that he is comfortable in using the application.

Before examining the system, the user must be aware that the server software needs to be running on the server in order to access the results. The actual process won't happen if the server object is not active and functioning on the server.

* + 1. **User Training**

The purpose of user training is to get the user ready to test and modify the system. The people who will be involved must have faith in their ability to contribute to the goal and benefits anticipated from the computer-based system. Training is more necessary as systems get more complicated. The user learns how to enter data, handle error messages, query the database, call up routines to generate reports, and execute other important tasks through user training.

* + 1. **Training on the Application Software**

The user will need to receive the essential basic training on computer awareness after which the new application software will need to be taught to them. This will explain the fundamental principles of how to use the new system, including how the screens work, what kind of help is displayed on them, what kinds of errors are made while entering data, how each entry is validated, and how to change the date that was entered. Then, while imparting the program's training on the application, it should cover the information required by the particular user or group to operate the system or a certain component of the system. It's possible that this training will vary depending on the user group and the level of hierarchy.

* + 1. **System Maintenance**

Maintenance is the enigma of system development. The maintenance phase of the software cycle is the time in which a software product performs useful work. After a system is successfully implemented, it should be maintained in a proper manner. System maintenance is an important aspect in the software development life cycle. The need for system maintenance is for it to make adaptable to the changes in the system environment. Software maintenance is of course, far more than "Finding Mistakes".

* + 1. **Hosting**

A web hosting service is a sort of Internet hosting service that makes websites for customers visible on the World Wide Web and provides the resources needed for them to construct and maintain a site. Web hosts are businesses that provide web hosting services. Here I use 000webhost.com to host my project which is a free website hosting solution that provides an array of valuable features, including a website builder, WordPress support, and no ads.

**CHAPTER 7 CONCLUSION AND FUTURE SCOPE**

* 1. **CONCLUSION**

DAKSHA-STATE YOUTH FESTIVAL AUTOMATION SYSTEM has to do with making appropriate effort to stop the rising problem to all manual youth festival management operations in order to enhance the operations of conducting youth festival. In this project the software or system that can be used to aid the state youth festival which is still operating manually have been successfully developed. The software has a large memory of storing all the data and also keeping the records in a highly effective and accurate manner. Datas are secure and consistent as everything is validated there no chance of incorrectness of datas entered.There is a vast scope for bringing updations to the system to make the system more effective of conducting state youth festival.

* 1. **FUTURE SCOPE**

DAKSHA-STATE YOUTH FESTIVAL AUTOMATION SYSTEM helps the officials of state youth festival in reducing their time for manual paper works. The project has very vast scope in future. The project can be implemented on to conduct state youth festival in future. such as implementing android app for judges for judging, accepting appeals so on. Project can be updated in near feauture as and when requirement for the same arises, as it is very flexible in terms of expansion.

**CHAPTER 8 BIBLIOGRAPHY**

**REFERENCES:**

* + - Gary B. Shelly, Harry J. Rosenblatt, “*System Analysis and Design*”, 2009.
    - PankajJalote, “So*ftware engineering*: a precise approach”, 2006.
    - IEEE Std 1016 Recommended Practice for Software Design Descriptions.

**WEBSITES:**

* + - [www.w3schools.com](http://www.w3schools.com/)
    - [www.jquery.com](http://www.jquery.com/)
    - https://colorlib.com

**CHAPTER 9 APPENDIX**

* 1. **Code**

**Login.php**

<?php

;

?>

<head>

<style>

body

{

background-image: url("dancer-2349565.png");

text-shadow: hsla(hue, saturation, lightness, alpha);

}

form

{

text-shadow: hsla(hue, saturation, lightness, alpha);

background-color:#ff9933;

letter-spacing: 0pt;

width:300px;

height: 300px;

border:10px ;

padding:40 px;

margin-top:150px;

margin-left: 700px;

text-align: center;

}

.button

{

width: 30%;

background:#ff6666 ;

border: 1px solid;

margin: 35px 0 10px;

height: 32px;

border-radius: 20px;

padding: 0 10px;

box-sizing: border-box;

outline: none;

color: #fff;

cursor: pointer;

}

span{

font-size: 16px;

margin-left: 10px;

font family: sans-serif;

}

.input-box

{

width: 90%;

height: 35px;

border-radius: 00px;

padding: 0 10px;

}

</style>

</head>

<body>

<a href="home1.php">HOME</a>

<center>

<form class="form" action="codes/login\_action.php" method="post">

<div id="validation-message" style="color:red;text-align:center;">

<?php if(isset($\_GET['error']))

echo $\_GET['error'];

?><hr>

</div>

<p><strong>LOGIN</strong></p>

<div class="input-box" data-validate="Enter username or email">

<input name="email" type="email" class="input-box" id="email" onchange="return Validata();" placeholder="Enter username or email" required>

<span class="focus-input100"></span>

</div>

<span id="msg5" style="color:red;"></span>

<script>

function Validata()

{

var val = document.getElementById('email').value;

if (!val.match(/([A-z0-9\_\-\.]){1,}\@([A-z0-9\_\-\.]){1,}\.([A-Za-z]){2,4}$/))

{

document.getElementById('msg5').innerHTML="Enter a Valid Email";

document.getElementById('email').value = "";

return false;

}

document.getElementById('msg5').innerHTML=" ";

return true;

}

</script>

<br><div class="input-box" data-validate = "Enter password">

<input name="password" type="password" class="input-box" placeholder="Enter The Password" required>

<span class="focus-input100"></span>

</div>

<div class="btn">

<button type="submit" class="button">

SUBMIT

</button>

<button type="reset" class="button">

RESET

</button>

</div>

<br>

<div class="text-center"><span>Don't have an account??</span>

<a href="reg.php" class="txt2 hov1">

<font color="#B12687">Register Here...</font>

</a>

</div>

</form>

</div>

</div>

<?php

if (@$\_GET['registered'] == 'true')

echo ("<SCRIPT LANGUAGE='JavaScript'>

window.alert('You are Succesfully Registered')</script>");

?>

</center>

<?php

;

?>

</body>

**Register.php**

<!DOCTYPE html>

<html>

<head>

<title>Register</title>

<meta name="viewport" content="width=device-width, initial-scale=1">

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<script

type="application/x-javascript"> addEventListener("load", function() { setTimeout(hideURLbar, 0); }, false); function hideURLbar(){ window.scrollTo(0,1); } </script>

<!-- Custom Theme files -->

<link href="css/style.css" rel="stylesheet" type="text/css" media="all" />

<!-- //Custom Theme files -->

<!-- web font -->

<link href="//fonts.googleapis.com/css?family=Roboto:300,300i,400,400i,700,700i" rel="stylesheet">

<!-- //web font -->

<script>

function registration() {

var name = document.getElementById("name").value;

var email = document.getElementById("email").value;

var number = document.getElementById("phone").value;

var address = document.getElementById("address").value;

var password = document.getElementById("password").value;

var cpassword = document.getElementById("cpassword").value;

//email id expression code

var pwd\_expression = /^(?=.\*?[A-Z])(?=.\*?[a-z])(?=.\*?[0-9])(?=.\*?[#?!@$%^&\*-])/;

var letters = /^[A-Za-z]+$/;

var filter = /^([a-zA-Z0-9\_\.\-])+\@(([a-zA-Z0-9\-])+\.)+([a-zA-Z0-9]{2,4})+$/;

var phoneno = /^\d{10}$/;

if (name == '') {

alert('Please enter your name');

}

else if (!letters.test(name)) {

alert('Name field required only alphabet characters');

}

else if (email == '') {

alert('Please enter your user email id');

}

else if (!filter.test(email)) {

alert('Enter the e-mail format correctly');

}

else if (number == '') {

alert('Please enter your number');

}

else if (!phoneno.test(number)) {

alert('Enter number correctly');

}

else if (password == '') {

alert('Please enter Password');

}

else if (address == '') {

alert('Please enter Address');

}

else if (cpassword == '') {

alert('Enter Confirm Password');

}

else if (!pwd\_expression.test(password)) {

alert('Upper case, Lower case, Special character and Numeric letter are required in Password filed');

}

else if (password != cpassword) {

alert('Password not Matched');

}

else if (document.getElementById("password").value.length < 6) {

alert('Password minimum length is 6');

}

else if (document.getElementById("cpassword").value.length > 12) {

alert('Password max length is 12');

}

else {

alert('Successfully Registered');

}

}

</script>

</head>

<body>

<!-- main -->

<div class="main-w3layouts wrapper">

<h1>Register Here</h1>

<div class="main-agileinfo">

<div class="agileits-top">

<form action="users/hm/registration.php" method="POST">

<input class="text" type="text" id="name" name="name" placeholder="Name" required="">

<input class="text email" type="email" id="email" name="email" placeholder="Email" required="">

<input class="text" type="text" name="phone" id="phone" placeholder="Phone" required="">

<input class="text" type="text" name="address" id="address" placeholder="Address" required="">

<input class="text" type="password" name="password" id="password" placeholder="Password"

required="">

<input class="text w3lpass" type="password" name="cpassword" id="cpassword"

placeholder="Confirm Password" required="">

<div class="wthree-text">

<div class="clear"> </div>

</div>

<input type="submit" onclick="registration()" name="submit" value="SIGNUP">

</form>

<p>Go to Home <a href="index.html">Click</a></p>

<p>Login <a href="login.html">Click</a></p>

</div>

</div>

</div>

<!-- //main -->

</body>

</html>

**home.html**

<html>

<link rel="stylesheet" type="text/css" href="home2.css">

<form>

</form>

<ul>

<li><a href="home1.php">HOME</a></li>

<li><a href="reg.php">REGISTRATION</a>

</li>

<li><a href="plogin.php">LOGIN</a>

</li>

<li><a>EVENT DETAILS</a>

<ul>

<li><a href="details.php">EVENT DETAILS</a></li>

<li><a href="scview.php"> SCHEDULE</a></li>

</ul>

</li>

<li><a href="fresult.php">RESULT</a></li>

<!--<ul>-->

<!--<li><a href="plogin.php">PARTICIPANTS VIEW</a></li>-->

<!--<li><a href="publicresultview.php">PUBLIC VIEW</a></li>-->

<!--</ul>-->

<li><a>CONTACT</a>

<ul>

<li><a href="home1.php">About As</a></li>

<li><a>Email</a></li>

<li><a>phone number</a></li>

</ul>

</li>

</li>

</ul>

<br><br><br><br><br><br><br><br>

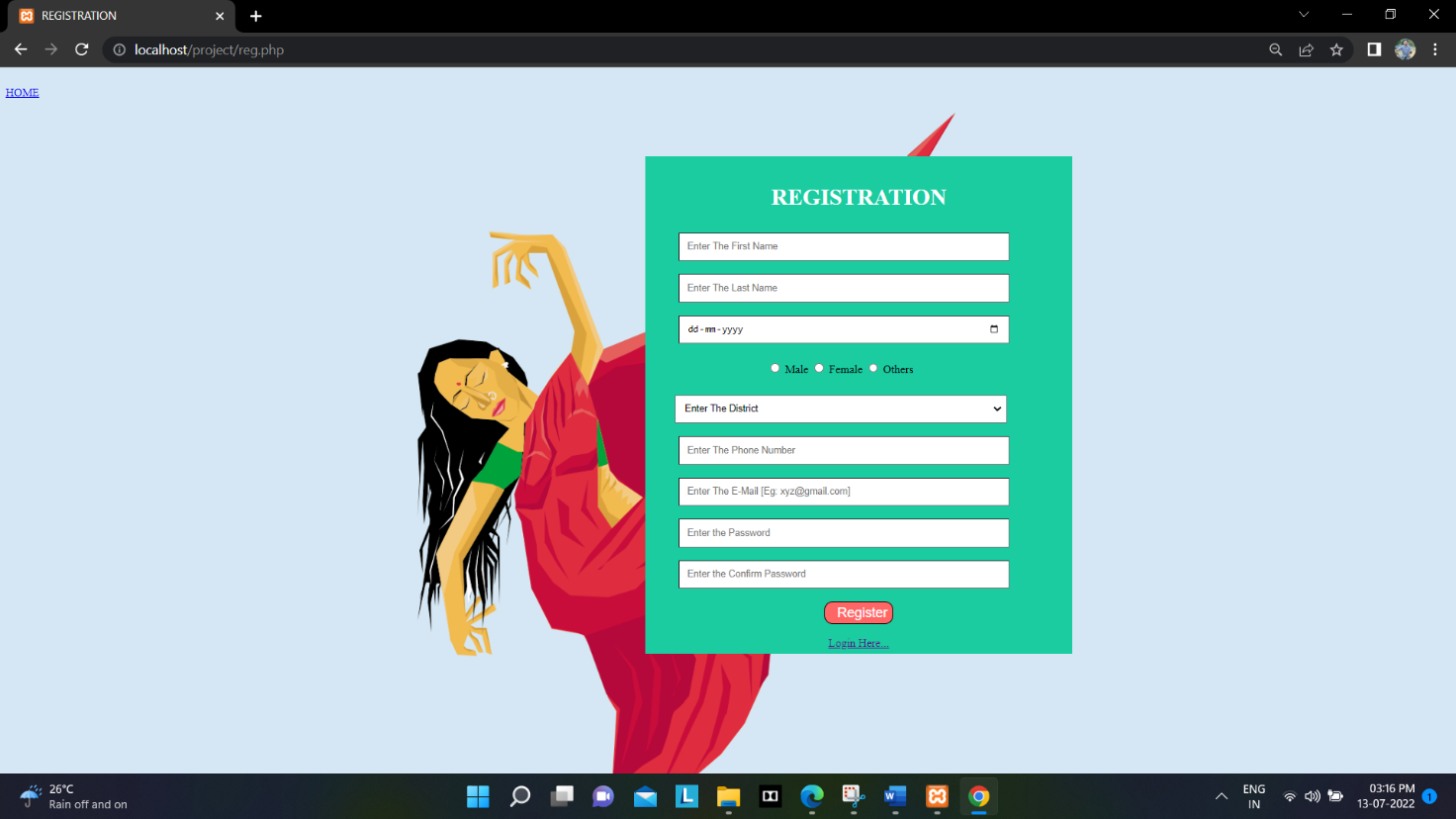
</html>

* 1. **Screen Shots**

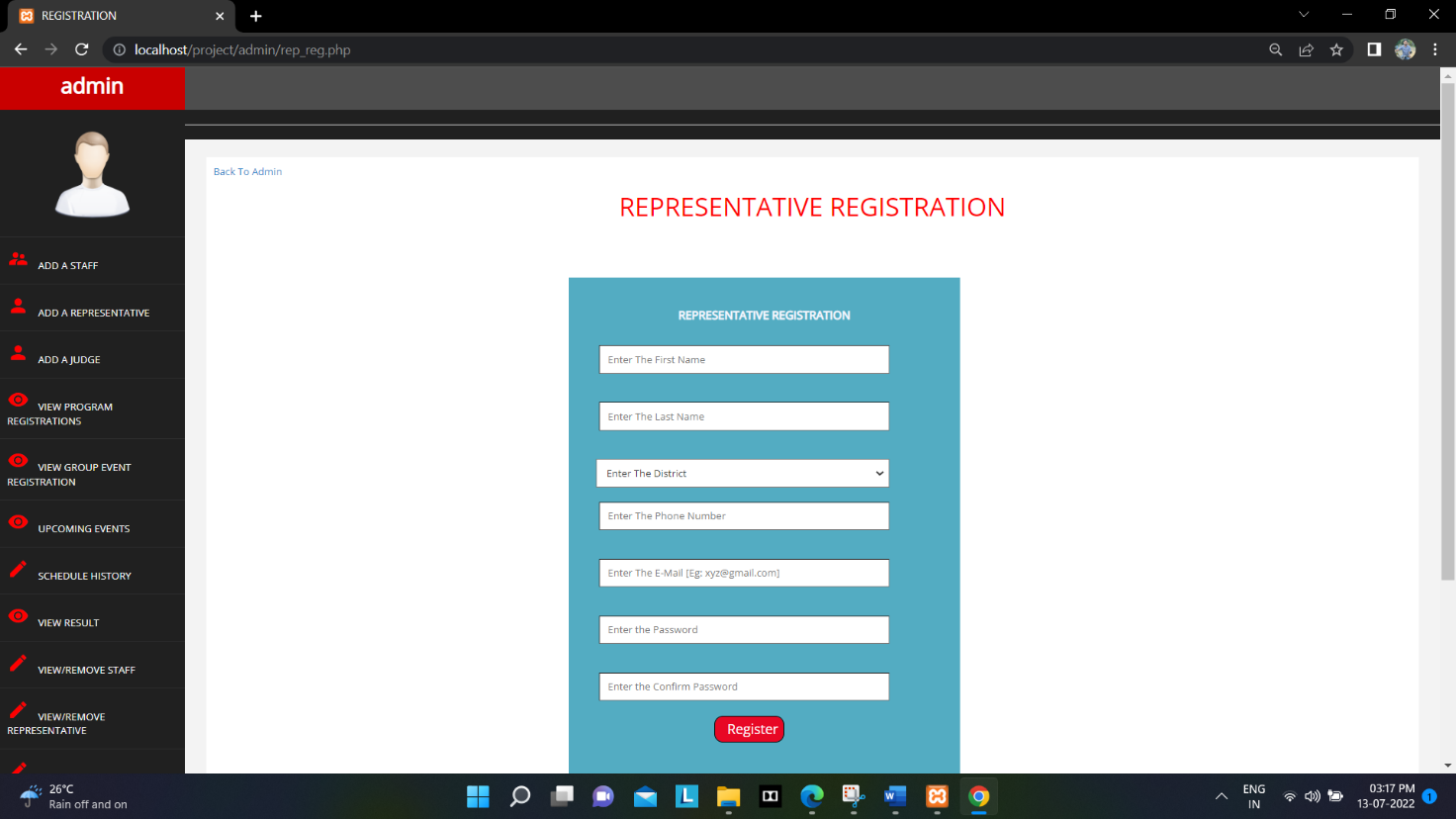
**Home page**



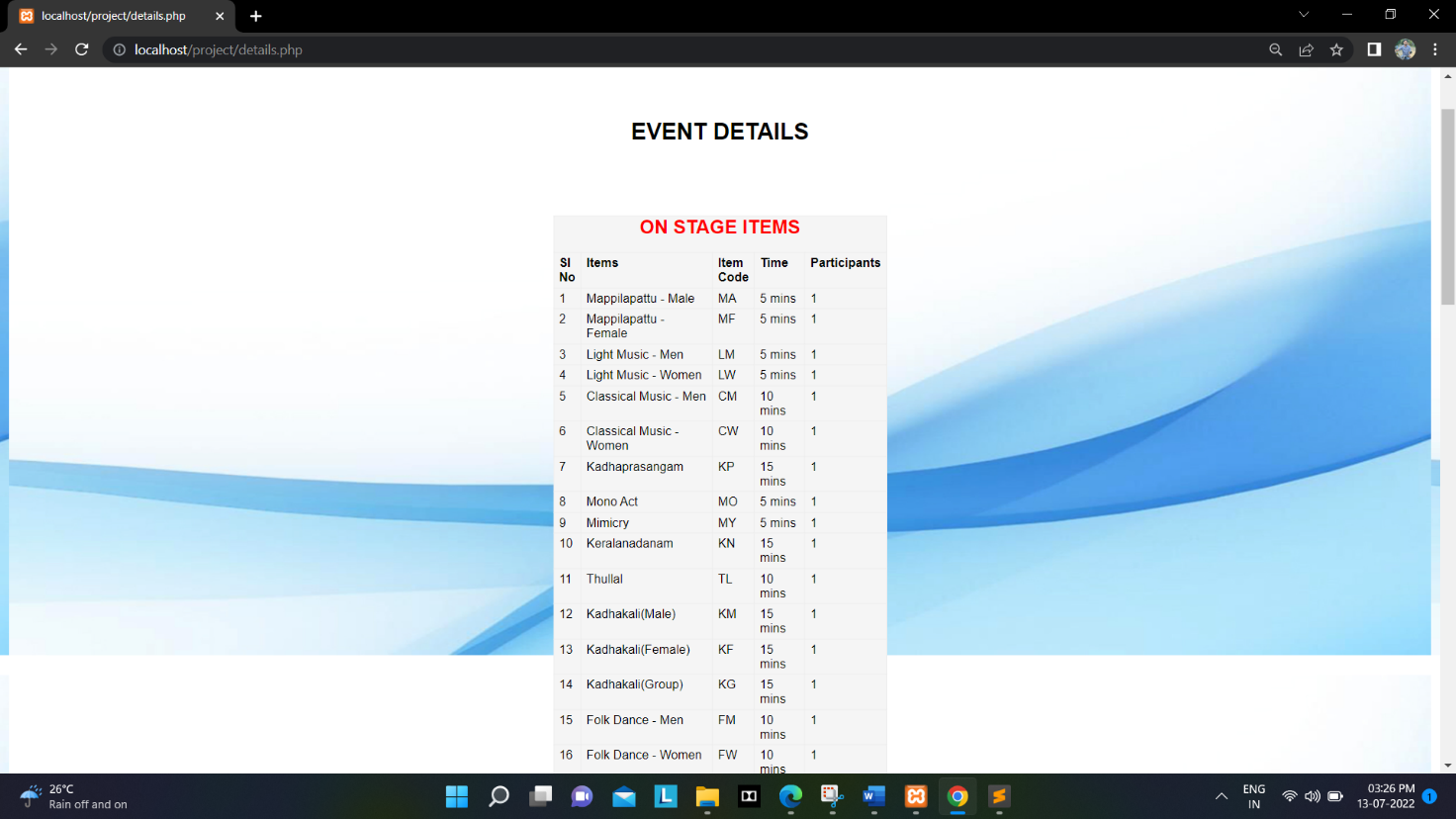
**Registration page**

****

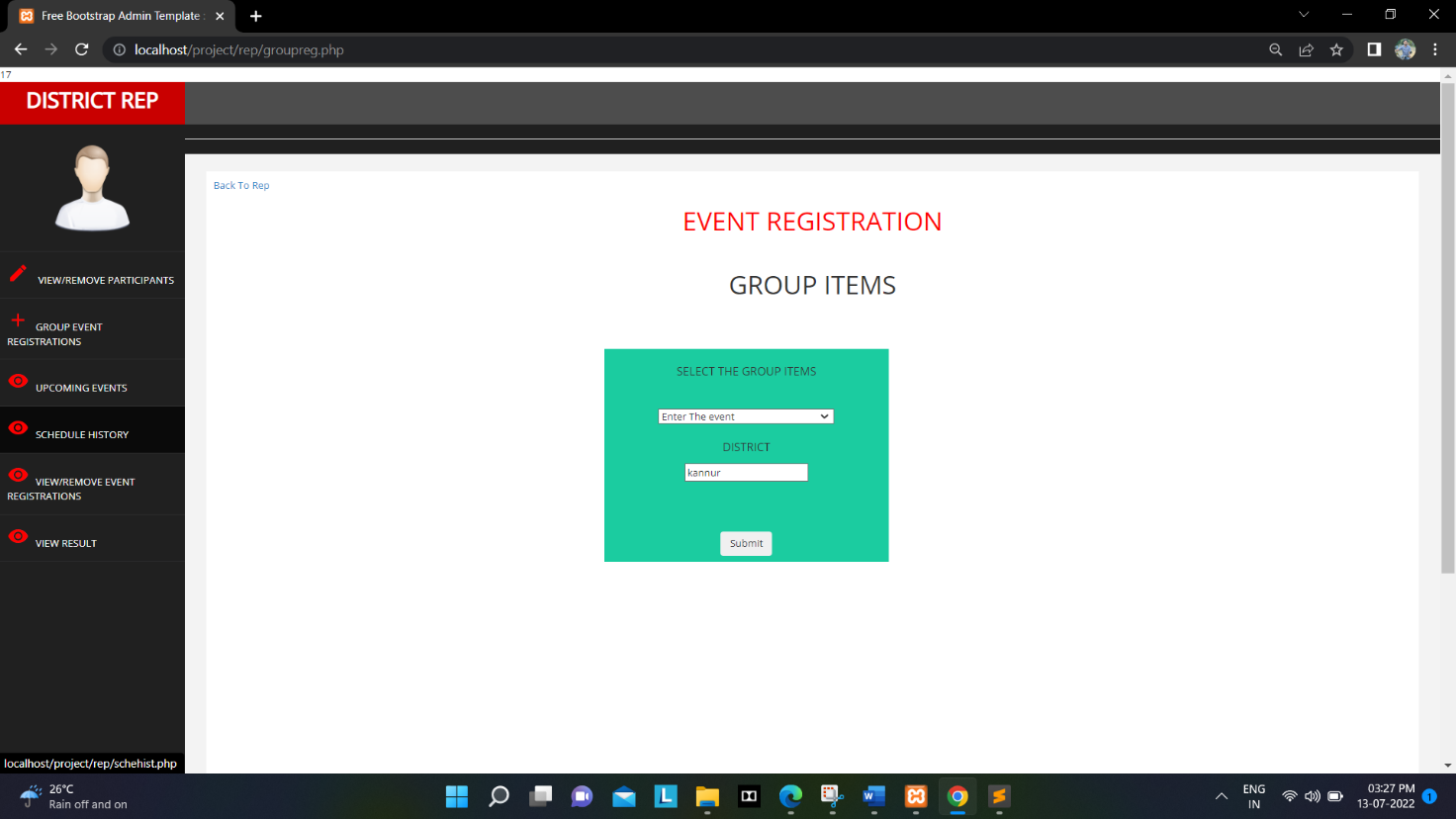
**Admin page**



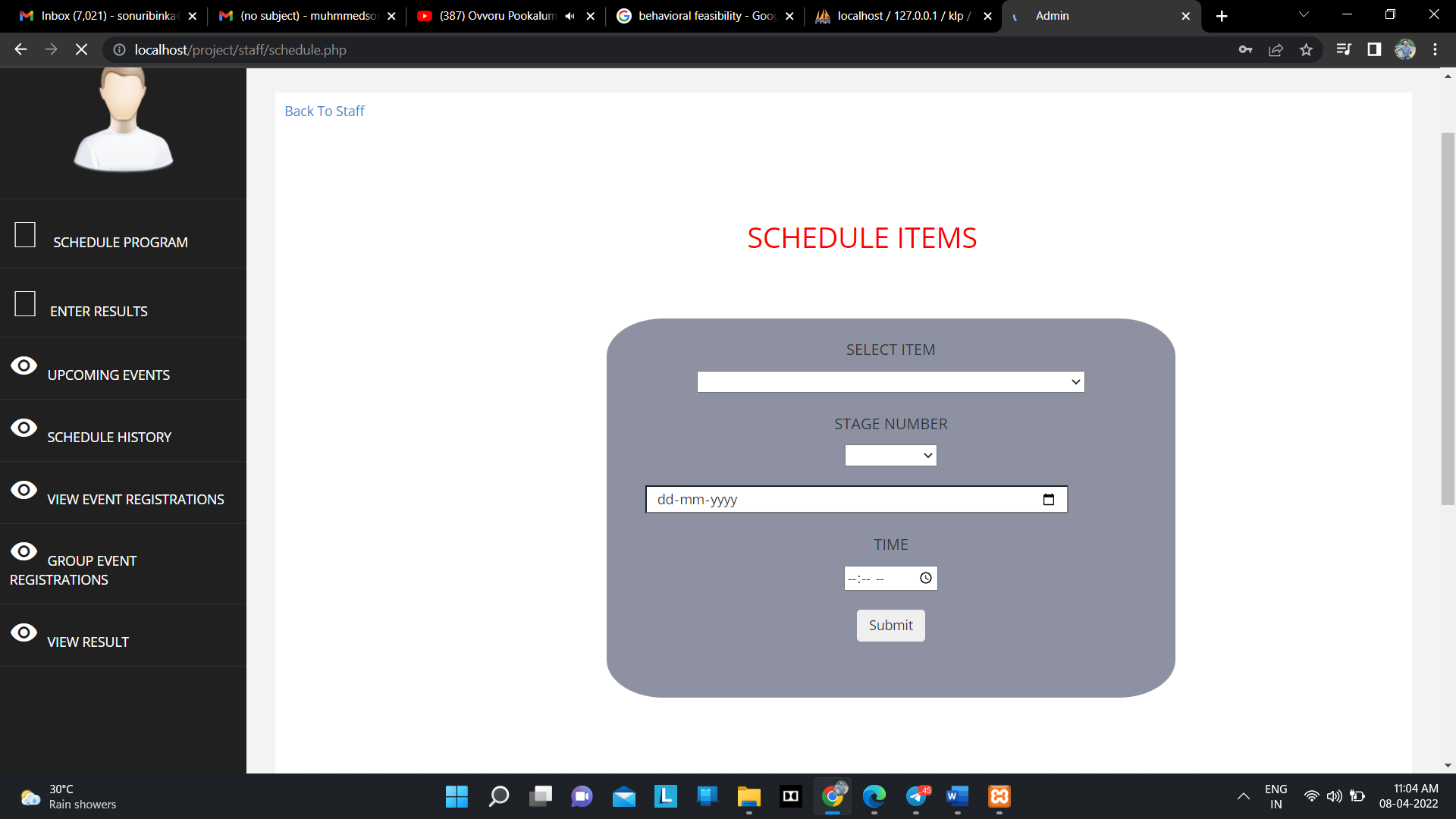
**Event details**

****

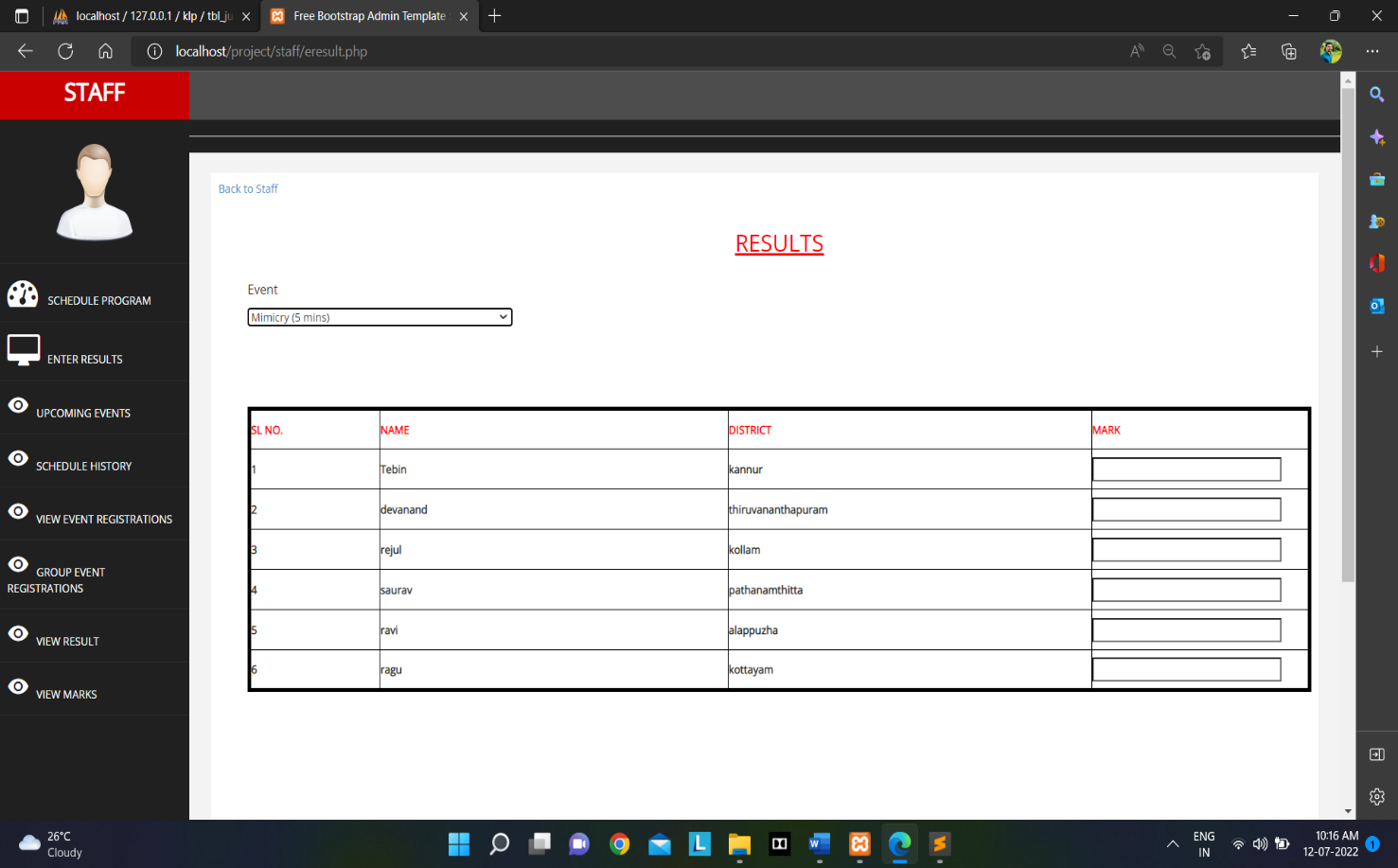
**Group event registration**

****

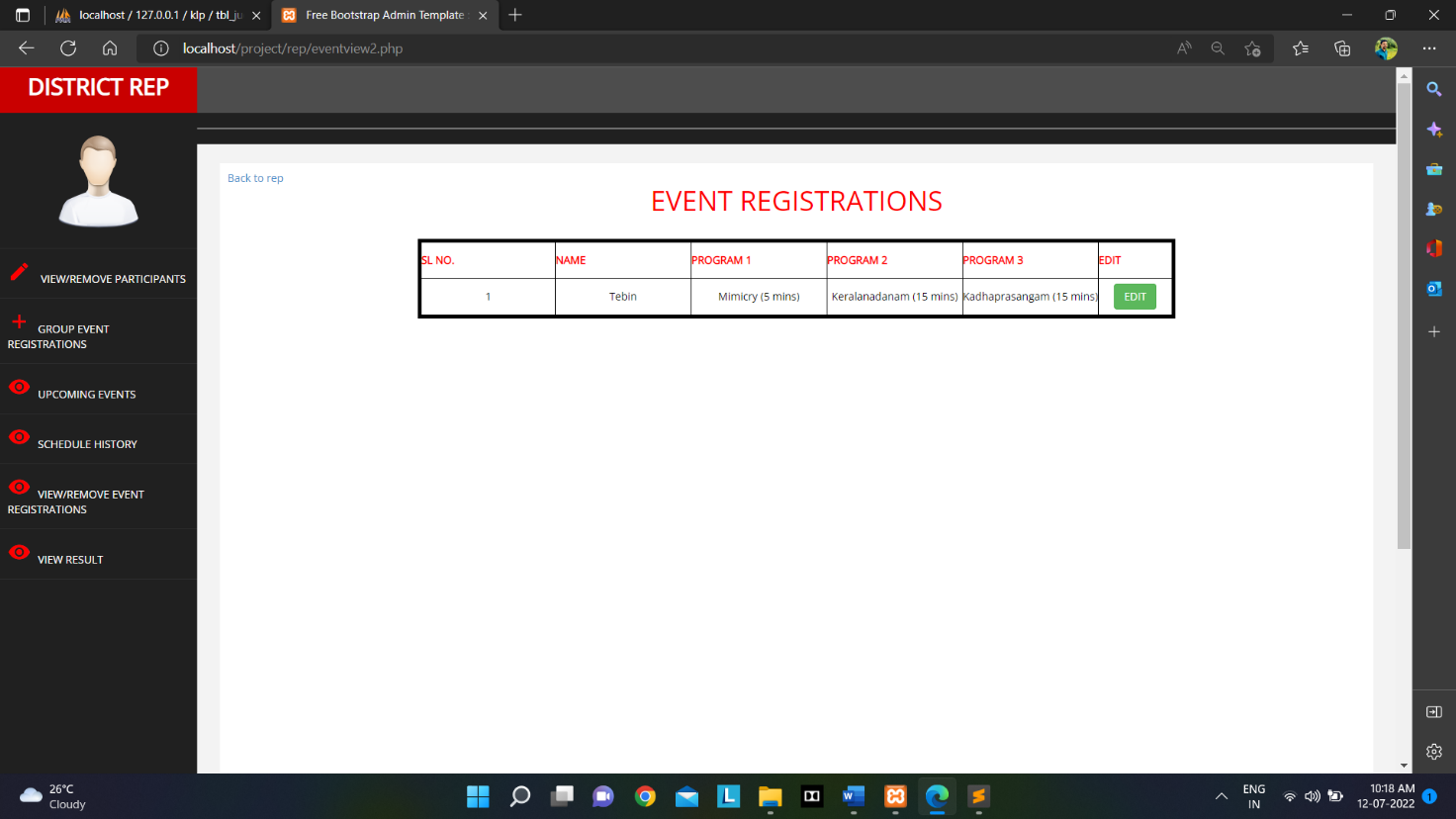
**Item scheduling**



**Mark entering**

****

**Event view/editing option**

****