**

**TRIBHUVAN UNIVERSITY**

**Prime College**

**Nayabazar, Kathmandu, Nepal**

**A Report**

**On**

**“Karyakram:**

**An Online Event Management System”**

**Submitted By:**

**Dipesh Shrestha (15316/074)**

**Kilesh Maharjan (15319/074)**

**Rijan Maharjan (15329/074)**

**A Report Submitted in partial fulfillment of the requirement of Bachelor of Science in Computer Science & Information Technology (BSc.CSIT) 7th Semester of Tribhuvan University, Nepal**

Supervisor’s Recommendations

I hereby recommend that the report prepared under my supervision by Dipesh Shrestha (TU Exam Roll No. 15316/074), Kilesh Maharjan (TU Exam Roll No. 15319/074) and Rijan Maharjan (TU Exam Roll No. 15329/074) entitled “Karyakram” in partial fulfillment of the requirements for the degree of B.Sc. in Computer Science and Information Technology be processed for evaluation.

---------------------------------

**Mr. Bipin Timalsina**

Project Supervisor

Certificate of Approval

This is to certify that this project prepared by Dipesh Shrestha (TU Exam Roll No. 15316/074), Kilesh Maharjan (TU Exam Roll No. 15319/074) and Rijan Maharjan (TU Exam Roll No. 15329/074) entitled “Karyakram” in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science and Information Technology has been well studied. In our opinion, it is satisfactory in the scope and quality as a project for the required degree.

-------------------------------------

**Mr. Narayan Prasad Sharma**

Principle

-------------------------------------

**Mr. Bipin Timalsina**

Project Supervisor

-------------------------------------

**Ms. Rolisha Sthapit**

Program Coordinator

-------------------------------------

**External Examiner**

Tribhuvan University

Kirtipur, Nepal

Acknowledgement

We would like to express our deepest appreciation to all those who provided us the possibility to complete this report. A special gratitude we would give to our project supervisor, **Mr. Bipin Timalsina**, whose contribution in stimulating suggestions and encouragement, helped us to coordinate our project especially in writing this report.

Furthermore, we would also like to acknowledge with much appreciation the crucial role of the staff of B.Sc. CSIT department, who gave permission to use all required equipment and the necessary materials to complete the task. A special thanks goes to constant support from our seniors and every teaching staff of B.Sc.CSIT department which helped us successfully complete our project. Last but not least, many thanks go to the head of the project, Mr. Bipin Timalsina who has invested his full effort in guiding the team in achieving the goal. We have to appreciate the guidance given by other supervisors as well as the panels especially in our project presentation that has improved our presentation skills thanks to their comment and advice.

With Respect,

Dipesh Shrestha (15316/074)

Kilesh Mahrajan (15319/074)

Rijan Maharjan (15329/074)

Abstract

Events have become more crucial for organizations as a means of advertising communication. The management of events is an interdisciplinary task, addressed withinside the most diverse fields in practice and in studies establishments. Because cautious initial planning and particular execution are extraordinarily crucial for events.

“Karyakram” is an Event Management System to create and develop large scale events. It involves gathering the necessary information about the target audience, developing the event concept, and organizing the logistical aspects of the event. This system also involves identifying the right people and processes to execute the event. This might include managing the teams of people responsible for each function, the budget, and overseeing the execution of the event. It involves overseeing all logistics leading up to and during an event, whether a conference, workshop, or any organized gathering.

**Keywords**

Events, Event management, project management, registration, activities, participants, tasks and assignments, schedules, calendar, notifications, access levels, reporting, dashboards, open source.

Table of Contents

[Supervisor’s Recommendations I](#_Toc97138220)

[Certificate of Approval II](#_Toc97138221)

[Acknowledgement II](#_Toc97138222)

[Abstract III](#_Toc97138223)

[Table of Contents IV](#_Toc97138224)

[List of Figures VI](#_Toc97138225)

[List of Tables VII](#_Toc97138226)

[List of Abbreviations VIII](#_Toc97138227)

[Chapter 1: Introduction 1](#_Toc97138228)

[1.1 Introduction 1](#_Toc97138229)

[1.2 Problem Statement 1](#_Toc97138230)

[1.3 Objectives 2](#_Toc97138231)

[1.4 Scopes and Limitations 2](#_Toc97138232)

[1.4.1 Scope 2](#_Toc97138233)

[1.4.2 Limitations 2](#_Toc97138234)

[1.5 Development Methodology 2](#_Toc97138235)

[1.6 Report Organization 4](#_Toc97138236)

[Chapter 2: Background And Literature Review 6](#_Toc97138237)

[2.1 Background Study 6](#_Toc97138238)

[2.2 Literature Review 6](#_Toc97138239)

[Chapter 3: System Analysis 7](#_Toc97138240)

[3.1 Requirement Analysis 7](#_Toc97138241)

[3.1.1 Functional Requirement 7](#_Toc97138242)

[3.1.2 Non-Functional Requirements 7](#_Toc97138243)

[3.2 Feasibility Analysis 8](#_Toc97138244)

[3.2.1 Technical Feasibility 8](#_Toc97138245)

[3.2.2 Operational Feasibility 8](#_Toc97138246)

[3.2.3 Economic Feasibility 8](#_Toc97138247)

[3.2.4 Scheduling Feasibility 9](#_Toc97138248)

[Chapter 4: System Design 11](#_Toc97138249)

[4.1 System Architecture and Overview 11](#_Toc97138250)

[4.1.1 System Architecture 11](#_Toc97138251)

[4.1.2 System Overview 12](#_Toc97138252)

[4.2 Algorithm Used 13](#_Toc97138253)

[4.3 UML Diagrams 15](#_Toc97138254)

[4.3.1 Use Case Diagram 15](#_Toc97138255)

[4.3.2 Sequence Diagram 17](#_Toc97138256)

[4.3.3 Activity Diagram 18](#_Toc97138257)

[4.3.4 Class Diagram 20](#_Toc97138258)

[4.3.5 Deployment Diagram 24](#_Toc97138259)

[Chapter 5: Implementation And Testing 25](#_Toc97138260)

[5.1 Implementations 25](#_Toc97138261)

[5.1.1 Tools Used 25](#_Toc97138262)

[5.1.2 Implementation details of modules 28](#_Toc97138263)

[5.2 Testing 31](#_Toc97138264)

[5.2.1 Unit testing and Test cases 31](#_Toc97138265)

[5.2.2 Test cases for Regression testing 32](#_Toc97138266)

[Chapter 6: Conclusion and Recommendations 40](#_Toc97138267)

[6.1 Conclusion 40](#_Toc97138268)

[6.2 Future Recommendations 40](#_Toc97138269)

[References 41](#_Toc97138270)

List of Figures

[Figure 1 SCRUM process 3](file:///C:\Users\97798\Desktop\latest-EMS-Report.docx#_Toc97130800)

[Figure 2 Gantt chart (Tabular) 9](#_Toc97130801)

[Figure 3 Gantt Chart 10](#_Toc97130802)

[Figure 4 Higher level System Architecture 11](#_Toc97130803)

[Figure 5 Lower-level System Architecture 11](#_Toc97130804)

[Figure 6 Low-level Use case diagram 15](#_Toc97130805)

[Figure 7 High-level Use case Diagram 16](#_Toc97130806)

[Figure 8 Sequence Diagram 17](#_Toc97130807)

[Figure 9 Client-side Activity Diagram 18](#_Toc97130808)

[Figure 10 Admin-side Activity Diagram 19](#_Toc97130809)

[Figure 11 Class Diagram for user module 20](#_Toc97130810)

[Figure 12 Class diagram for event module 21](#_Toc97130811)

[Figure 13 Class diagram for venue module 22](#_Toc97130812)

[Figure 14 Deployment diagram 24](#_Toc97130813)

[Figure 15 Result of unit testing 31](#_Toc97130814)

List of Tables

[Table 1 Test case for Registration 32](#_Toc97130815)

[Table 2 Test cases for Login 33](#_Toc97130816)

[Table 3 Test case for Role 34](#_Toc97130817)

[Table 4 Test Case for Vendor 35](#_Toc97130818)

[Table 5 Test Case for Event 36](#_Toc97130819)

[Table 6 Test case for Venue 37](#_Toc97130820)

[Table 7 Test case for Space 38](#_Toc97130821)

[Table 8 Test case for Facility 38](#_Toc97130822)

[Table 9 Test Case for Attendee 39](#_Toc97130823)

List of Abbreviations

EMS : Event Management System

JSON: Javascript Object Notation

JOSE: JSON Web Signature and Encryption

API: Application Programmable Interface

REST: Representational State Transfer

GraphQL: Graph Query Language

JWT: JSON Web Token

CMS: Content Management System

MINIO: AWS Object storage

HTML: HyperText Markup Language

CSS: Cascading Style Sheets

JS: JavaScript

AWS: Amazon Web Services

# Introduction

## Introduction

In management terms, event management is the process of planning and executing an event. Events are managed in a variety of ways, and different approaches are taken to handle them in general. To make an event a success, efficient management is necessary in all instances. For this purpose, an effective system of handling events through online solutions helps organizations to get better outcomes and efficient management. The system is named “Karyakram”. It has been developed using Node Js and React Js. Other Technologies used are CSS, HTML5, PostgreSQL and Redis.

“Karyakram” is an Event Management System aimed to provide a better and more effective method of working with the existing manual system. It eliminates the various issues that the system commonly encounters and helps in reducing them. The system tries to minimize the errors while entering the data and provides error messages while doing so.

Event management is a broad field that involves handling various aspects of a large-scale event. It can include planning, organizing, and managing the event's various components. The event management system is used to keep track of all event-related activities. In any event, multiple service providers are working at the same time, making management difficult. It's also crucial for event planners that he has all of these service providers' contact information so that he can Contact them at any time to schedule an event. So this software was developed to keep track of all these activities. To be successful in the competition in order to manage a business, the user must have a strong network of service contacts. provider. These individuals are essentially service providers who supply certain services such as Sound systems services, Lighting providers, Canteen services, stage construction and so on.

In the current system, Event Company is required to perform all management tasks manually. All payment information is kept on paper. There is no system in place to check previous event expenses. To do so, they must check the payment register, which is a time-consuming and tedious task. This system was created with this entire problem in mind. This solution allows event management companies to maintain their paperwork online and access reports from the most recent event they hosted.

## Problem Statement

Every Organization, whether big or small, has challenges to overcome and manage every event. To schedule, book, and organize an event such as a birthday party, wedding, reception, or ring ceremony, conference, class, training, or workshop the client must visit the company office. Clients must pay in cash to make a reservation. Customers waste a lot of time looking for and contacting event organizers. So, use of an online event management system helps customers make booking, schedule events online at any preferred time. Event Company is required to perform all management tasks manually. All payment information is kept on paper. There is no system in place to check previous event expenses. To do so, they must check the payment register, which is a time-consuming and exhausting operation.

## Objectives

* The objectives of the system are:
* To develop a CMS based online portal.
* To enhance event scheduling.
* To develop role-based user management.
* To enhance selecting and reserving venues.
* To develop a centralized system which stores data and is available to all the event managers.
* To develop a system that allows participants to register for any happening event from anywhere.
* To enhance monitoring the event by keeping the track of all sessions and attendees. To enhance notification delivery to the user.

## Scopes and Limitations

### Scope

The scopes of the system are:

* To create a system that successfully maintains all of the data associated with the numerous events that occur within an organization.
* To have a centralized database with all event-related data.
* To support numerous operations and procedures that are required to efficiently handle data.

### Limitations

The limitations of the system are:

* There is an inability to collect detailed data and information from the customer as compared to in person meetings as we get only a limited amount of information from the forms.
* A skilled manpower is required to use the system.
* An active Internet connection is required.
* Immature processes and practices followed by vendors and organizers.

## Development Methodology

The Agile methodology is a style of project management that divides a project into phases. It's a project management method that entails ongoing collaboration and iteration, allowing us to deliver value to clients more quickly and with fewer headaches. Work is produced in small, digestible amounts with agile. Requirement plans and results are evaluated on a regular basis. Changes are progressive and evolutionary rather than revolutionary with user engagement, thus it can be useful in promoting change that is important to the success of most project transitions. As a result, the agile development methodology is a suitable decision.

**Scrum:**

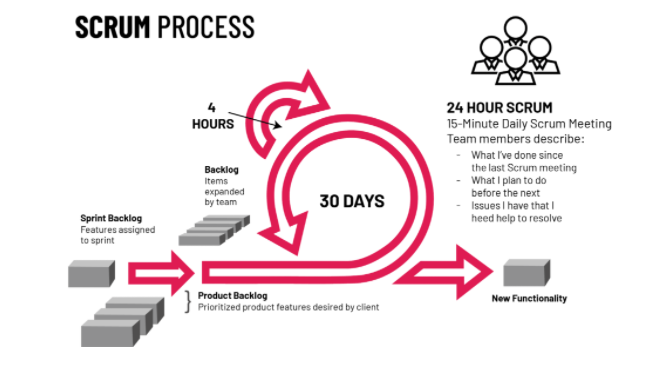
****A Scrum process is distinguished from other agile processes by specific concepts and practices, divided into the three categories of Roles, Artifacts, and Time Boxes. These and other terms used in Scrum are defined below. Scrum is most often used to manage complex software and product development, using iterative and incremental practices.

Figure 1 SCRUM process

Scrum significantly increases productivity and reduces time to benefits relative to classic “waterfall” processes. Scrum processes enable organizations to adjust smoothly to rapidly-changing requirements, and produce a product that meets evolving business goals. An agile Scrum process benefits the organization by helping it to

* Increase the quality of the deliverables
* Cope better with change (and expect the changes)
* Provide better estimates while spending less time creating them
* Be more in control of the project schedule and state

**User Story**

A User Story describes a desired feature (functional requirement) in narrative form. User Stories are usually written by the Product Owner, and are the Product Owner’s responsibility. The format is not standardized, but typically has a name, some descriptive text, references to external documents (such as screen shots), and information about how the implementation will be tested. For example, a Story might resemble the following:

The elements in this User Story are:

**Name:** The Name is a descriptive phrase or sentence. The example uses a basic “Role-Action-Reason” organization. Another common style, popularized by Mike Cohn, follows the template “As a <type of user>, I want <some goal> so that <some reason>.” The choice of template is less important than having a workable standard of some kind.

**Description:** This is a high-level (low-detail) description of the need to be met. For functional (user-facing) requirements, the description is put in narrative form. For non-functional requirements, the description can be worded in any form that is easy to understand. In both cases, the key is that the level of detail is modest, because the fine details are worked out during the implementation phase, in discussions between team members, product owners, and anyone else who is involved. (This is one of the core concepts of Scrum: Requirements are specified at a level that allows rough estimation of the work required to implement them, not in detail.)

**Screens and External Documents:** If the Story requires user-interface changes (especially non-trivial ones), the Story should contain or link to a prototype of the changes. Any external documents required to implement the Story should also be listed.

**How to test:** The implementation of a Story is defined to be complete if, and only if, it passes all acceptance tests developed for it. This section provides a brief description of how the story will be tested. As for the feature itself, the description of testing methods is short, with the details to be worked out during implementation, but at least a summary is needed to guide the estimation process.

## Report Organization

The report is based on six chapters. The project's resources, requirements, and diagrammatic representations are divided into six chapters. Each chapter follows the creation of our project from start to finish.

The first chapter gives a general summary of our project. It foreshadows and includes key concepts that will be covered in the following chapters.

The project's background study and literature evaluation are detailed in Chapter 2, which follows the introduction chapter. It enables you to distinguish between different hypotheses, strategies, and problems in the current investigation.

The third chapter covers a lot of ground when it comes to system analysis. It includes a requirement analysis, a feasibility analysis, and object-oriented analysis. It outlines each experiment's goals and addresses the difficulties presented in Chapter 3.

Depending on the method chosen in Chapter 4, the system design in Chapter 4 takes either a structured or an object-oriented approach. A structured strategy includes database design, forms and report design, as well as interface and dialogue design. However, an object-oriented technique is employed, which involves the refinement of class, state, object, sequence, and activity diagrams. Diagrams of components and deployment are also supplied. This helps you understand the system design and algorithm that was used to create it.

In Chapter 5, a number of CASE tools, as well as computer languages and database platforms, are briefly discussed. In this sub-topic, the implementation details of modules are defined. During the testing phase, unit and system testing are performed to assure accuracy. This chapter discusses the results of the tests.

In Chapter 6, the project's findings and future recommendations are offered. This gives an indication of how the project will be evaluated as a success and what improvements might be made.

# Background And Literature Review

## Background Study

Events are basically gatherings of people at a certain time in a certain place to create, operate and participate in an experience and relay a message to the targeted group. The supervision and assistance of well-trained and experienced professionals are required to organize a perfectly synchronized, well-planned, well-conducted, and memorable event. Budgeting, scheduling, location selection, obtaining appropriate permits, coordinating transportation and parking, arranging for entertainment, arranging decorations, catering, and emergency preparations are all included in the process of planning an event [1]. Communication may now take place at any time of day, from anywhere in the world, thanks to the use of these electronic devices. Individuals can now use any portable device to establish, locate, register, and document an event.

Event management is the application of project management to the creation and development of small and/or large-scale personal or corporate events such as festivals, conferences, ceremonies, weddings, formal parties, concerts, or conventions. It involves studying the brand, identifying its target audience, devising the event concept, and coordinating the technical aspects before actually launching the event. Prior to 2000’s, all documents were physically mailed, and event organizers had to use handwritten registration processes. It could take days for mail to arrive at its final destination, and documents were easily lost. But now documents can be easily emailed or uploaded through the Internet and the registration can be performed online. With online registration implemented in an event, the event planner/host can virtually see the people who are interested or are attending the event. Before the internet, people had to carry a physical map if they didn't know the direction of the venue. But now one simply needs to type in the address of the venue in google maps and it provides the direction to the venue.

## Literature Review

As mentioned above, events are the public assembly to gather people for the celebrations, education, marketing or reunion. Events can be of many types based on their size, type and context. In the current situation, the existing system has numerous problems that make it ineffective to continue using. [2] The event's execution may be delayed due to an uncontrollable planning stumbling block. In terms of quality, it's adequate but not exceptional, compared to when handled through a computerized system. In the existing system the customer contacts the company for event management. Then the customer provides the details of the event and its requirements. The customer explains its aims, how long it will last, its format (Presentation/Workshop and/or Exhibition etc.), expected number of delegates/guests, equipment and furniture required, whether any delegate pack or promotional material is to be distributed, and other facilities required. The Event Manager studies the requirements of the event carefully and using the event management system The company offers some readymade packages to choose from. If the customer agrees, the event is booked and the advance deposit is taken by the company. According to the requirements of the event, different bookings are made. A strategic schedule is prepared for smooth conduct of the event. The Event Management System helps the manager in different tasks of planning, scheduling and conducting the event. This system provides instant access to event-related information.

# System Analysis

## Requirement Analysis

The process of precisely locating, describing, and documenting various requirements connected with a certain business objective is known as requirement analysis. Gathering requirements aids in determining customer demands, determining project size, and determining the timeframes and resources required to finish the project.

### Functional Requirement

1. **Registration**

In order to access this site, the user must first register. First and last names, user names, email addresses, passwords, and confirm passwords are all required for registration.

1. **User Login**

The system allows users to log in to the system.

Username and password are required for the User Profile page.

1. **Choosing Event**

The user can choose an event as well as a payment method.

1. **Venue Selection**

These are time slots for the availability of a location or venue where an event will be hosted.

1. **Forgot Password**

The user can send a password reset link to their email address.

Input: e-mail address

Output: Reset link sent to email id.

1. **Logout**

The mechanism allows you to log out of the website.

### Non-Functional Requirements

1. **System Reliability Requirements:**

* The system is dependable
* If the request cannot be processed, an appropriate error message is displayed
* Web pages load in a matter of seconds

1. **Requirements for safety**

* It is necessary to keep track of its specifics.
* Authentications of users is required
* Database is backed up on a regular basis

1. **Requirement for safety**

* The user can view his profile after providing his password and user id.
* The information of the users is kept safe and secure.
* Details are not shared.

## Feasibility Analysis

### Technical Feasibility

This includes analyzing whether the system's technology is available, how difficult it will be to construct, and whether the company has enough experience with it. An outline of the system's input, processes, output, fields, programs, and procedures is used to make the assessment. This could be qualified in terms of volume of data, trends, and frequency of updating in order to provide an introduction to the technical system. The application requires a minimum of Windows XP platform with a high configuration of 1GB RAM and an Intel Pentium Dual Core CPU. This is technically doable. The purpose of the technical feasibility evaluation is to learn about the organization's current technology resources and how well they match the needs of the proposed system. It's a test of the hardware and software to assess how well they meet the system's requirements.

### Operational Feasibility

The proposed system follows PIECES framework which helps in identifying problems to be solved and their urgency:

**Performance:**

With the implementation of image compression database normalization, the system will provide adequate throughput and response time.

**Information:**

The system includes optimized form to get maximum information which will provide end users and managers with timely, pertinent, accurate, usefully formatted information.

**Economy:**

Initially this system will be hosted in a free hosting system to provide cost effective information to the service.

**Control:**

To offer effective controls to protect against fraud and guarantee security of data and information the system collects user details for user identification and verification.

**Efficiency:**

This system maximizes the use of available data and resources Services: The proposed system will be initially in testing phase with a limited area and later will grow globally. So, the system will provide reliable service and will be flexible and expandable for the future works

### Economic Feasibility

Economic feasibility analyses the project’s costs and revenue in an effort to determine whether it is possible to complete or not. Technologies that are being used to develop the project are freely available and initially the web site is hosted in a free web hosting space. So, the system is economically feasible.

### Scheduling Feasibility



Figure 2 Gantt chart (Tabular)

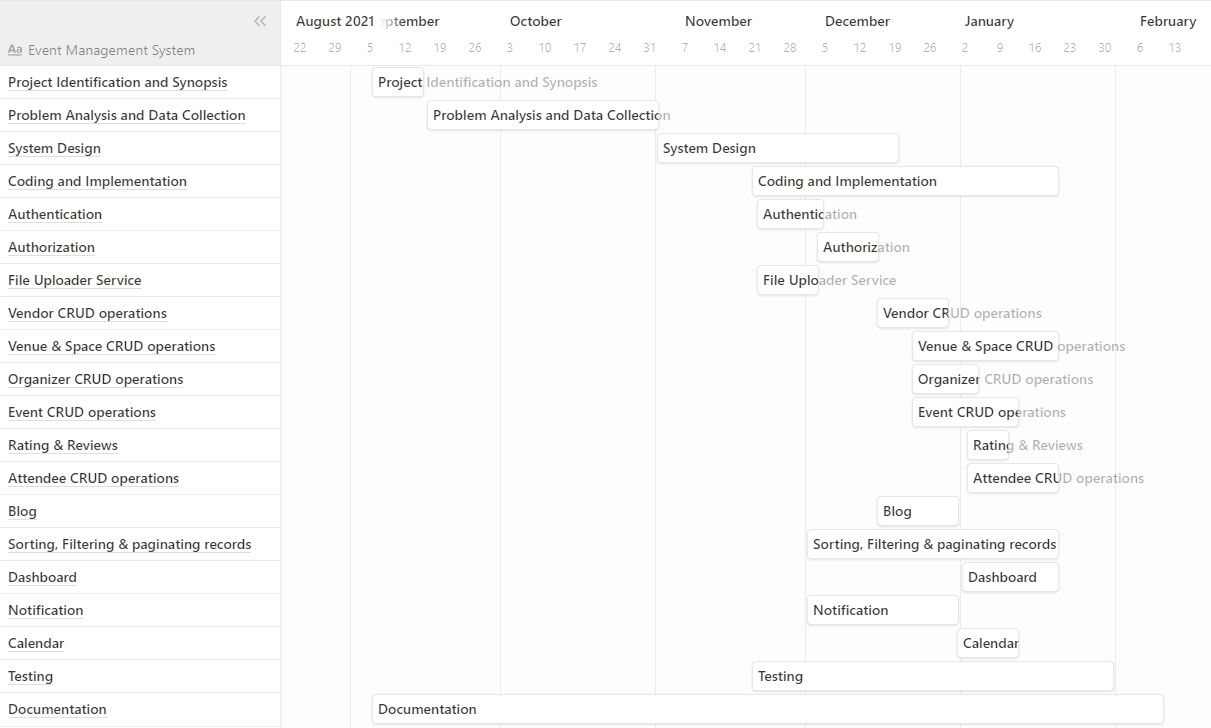


Figure 3 Gantt Chart

The Gantt chart for the development plan is given above. The plan explains the tasks versus the time that will take to complete.

The project was officially started on September 5, 2021 with the problem identification and synopsis. The system design was started on November 1, 2021, the coding and implementation activities were carried out parallely with system design from November 20, 2021 for generic modules like user authentication & authorization. The testing were parallely conducted as TDD approach was preferred. However different testing methods were used and were conducted for a longer period till January 31, 2021. The documentation phase was carried for the entire period. The project was concluded on February 10, 2021.

# System Design

## System Architecture and Overview

### System Architecture

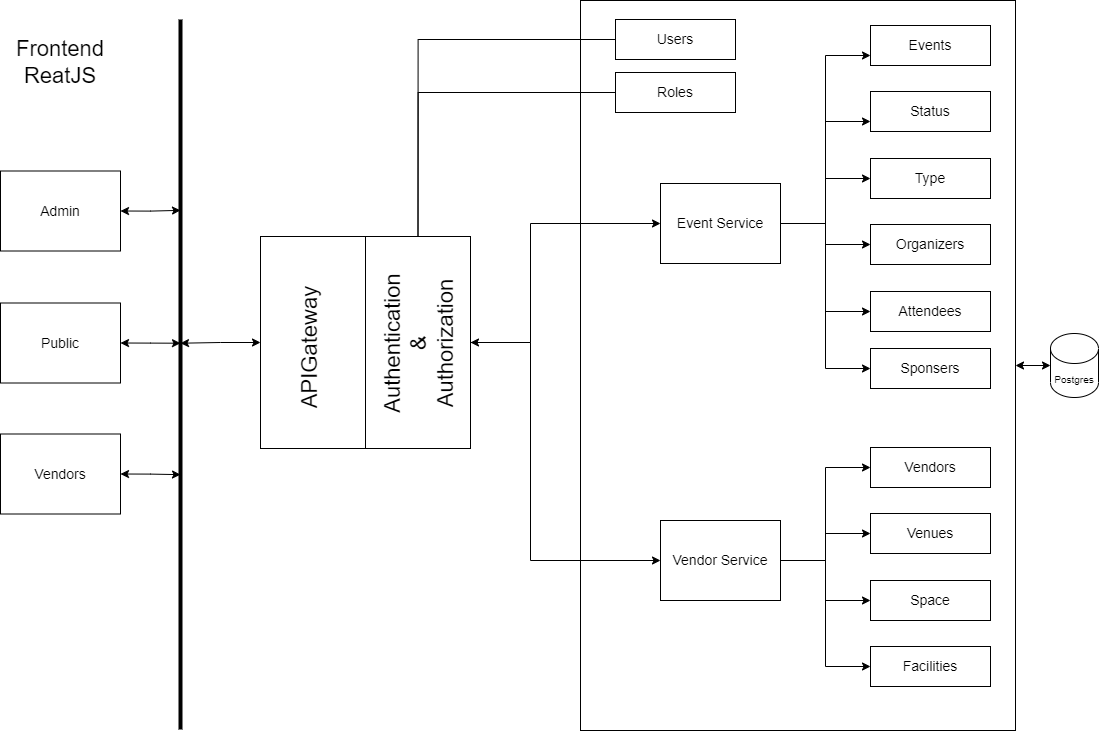


Figure 4 Higher level System Architecture

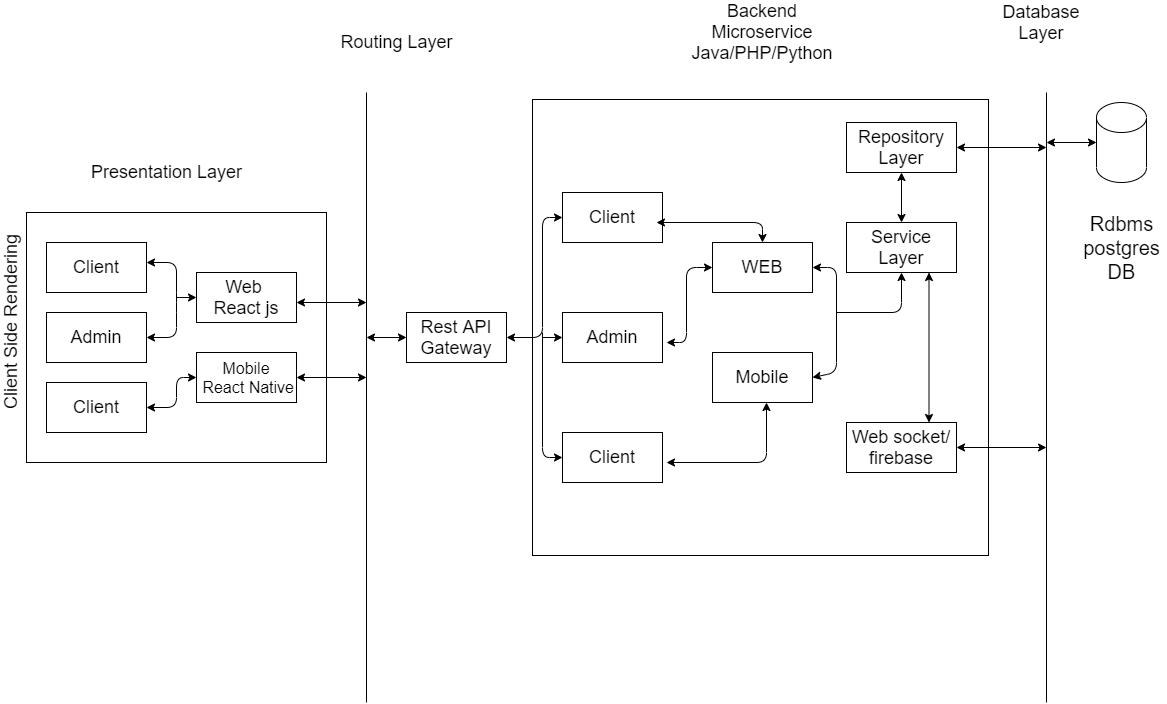


Figure 5 Lower-level System Architecture

### System Overview

The system consists of 4 layers namely:

* **Presentation layer (Frontend)**

This layer consists of client, admin and a vendor interface designed using HTML, CSS, JS and ReactJs. The presentation layer contains user interfaces which can be viewed and navigated by users. The actions performed by the users are passed to the Routing layer.

* **Routing layer (API Gateway)**

This layer handles the incoming requests from the presentation layer and proxies it to the backend. In this layer a load balancer like NGINX is used.

However, it also consists of user authorization and authentication. The routing layer again forwards the response received from the backend layer to the presentation layer.

* **Backend layer**

It receives all the requests from the presentation layer via the routing layer. In this layer different micro-services are present like event-service, vendor-service, billing-service and more. All the services are connected to a single database from which data is fetched and updated. The service layer handles the business logic of the application and passes the response to the routing layer.

* **Database layer**

Postgres Database is used for storing the system's data. The features provided by this database are suitable for the current system like generation of UUID, storing JSON data and querying them. Provides the term schema which acts as a virtual database for further abstraction and modularity.

## Algorithm Used

**JSON Web Token:**

A JSON Web Token encodes a series of claims in a JSON object. Some of these claims have specific meaning, while others are left to be interpreted by the users. Common claims are:

* Issuer (iss)
* Subject (sub)
* Audience (aud)
* Expiration time (exp)
* Not before (nbf)
* Issued at (iat)
* JWT ID (jti)

A JWT is usually complemented with a signature or encryption. A signature allows a JWT to be validated against modifications. Encryption, on the other hand, makes sure the content of the JWT is only readable by certain parties.

**JOSE header**

Signed and encrypted JWTs carry a header known as the JOSE header (JSON Object Signing and Encryption). This header describes what algorithm (signing or encryption) is used to process the data contained in the JWT. The JOSE header typically defines two attributes: alg and typ.

alg: the algorithm used to sign or encrypt the JWT.

typ: the content that is being signed or encrypted (usually 'JWT').

**Compact Representation**

The compact representation is basically the concatenation of the JOSE header, the JWT and the details of the signature. Each component is BASE64 encoded and separated by a single dot ('.').

**Example:**

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiYWRtaW4iOnRydWV9.TJVA95OrM7E2cBab30RMHrHDcEfxjoYZgeFONFh7HgQ

**Common JWT Signing Algorithms**

Most JWTs in the wild are just signed. The most common algorithms are:

* HMAC + SHA256
* RSASSA-PKCS1-v1\_5 + SHA256
* ECDSA + P-256 + SHA256

JWTs are a convenient way of representing authentication and authorization claims for your application. They are easy to parse, human readable and compact. Solutions such as session-ids and server-side tokens seem old and cumbersome when compared to the power of JWTs.

## UML Diagrams

### Use Case Diagram

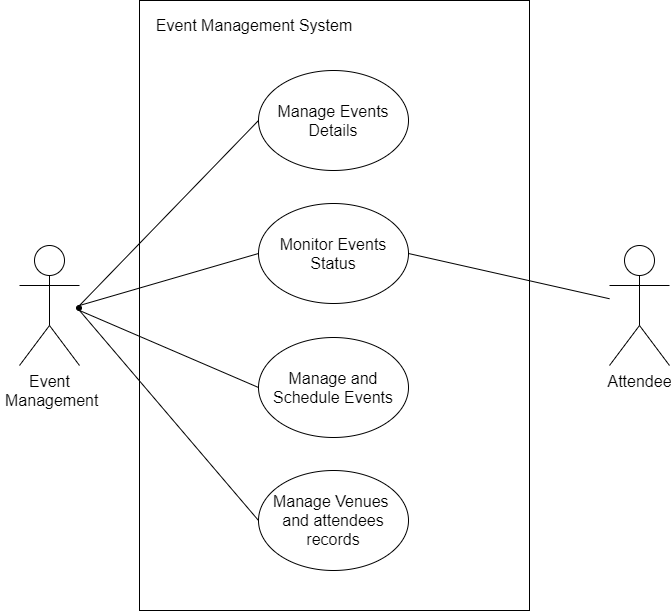


Figure 6 Low-level Use case diagram

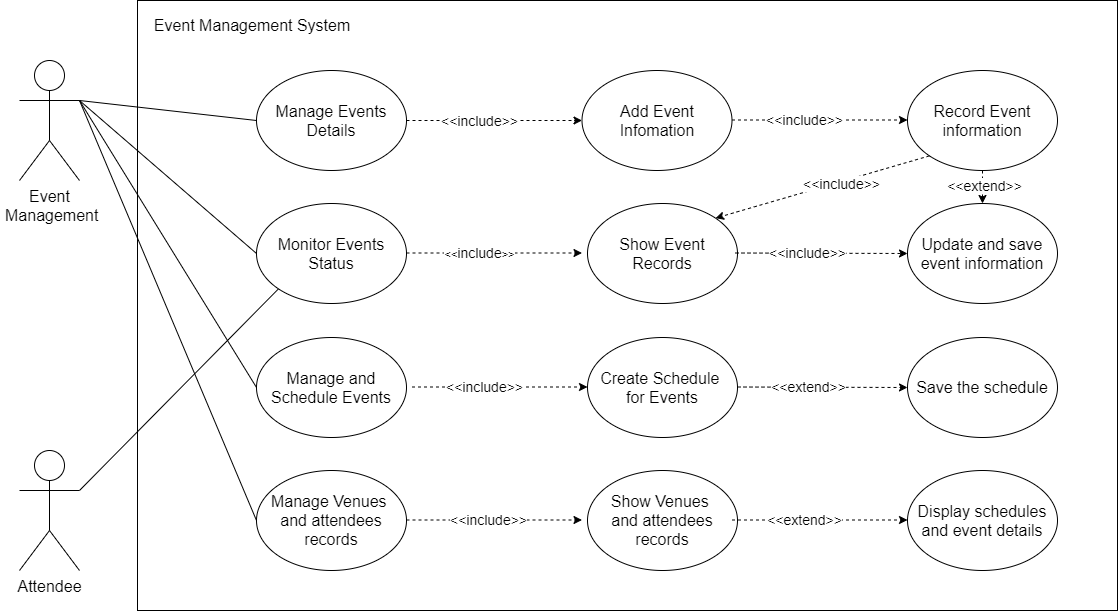


Figure 7 High-level Use case Diagram

This use case diagram is a visual representation of the process required to create and publish an event. An event manager initially creates an event with the scheduled date and time, selecting the desired venue. An attendee freely views the event and browse through its details, and decides his registration. Once an attendee requests to join an event the event manager views it and decides to accept/reject.

### Sequence Diagram

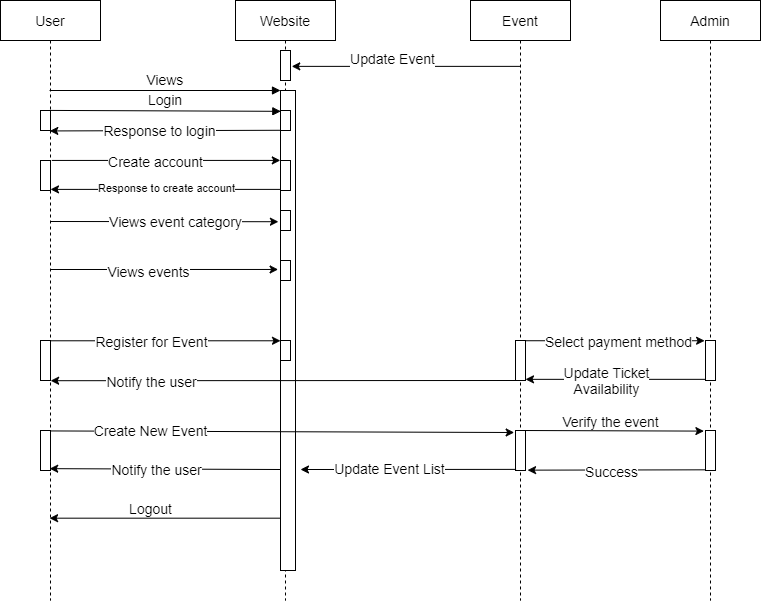


Figure 8 Sequence Diagram

The system allows users to view the event categories and respective events. Once the user tries to register for an event, the system checks for authentication and redirects to the registration page. The registration request is sent to the system from which the data is validated and appropriate response is sent back. The user is then redirected to the login page, from which they can login to the system and browse all the event categories, event details and more. A notification is sent to the event organizer whenever a user registers for an event.

### Activity Diagram

#### Client-side activity diagram

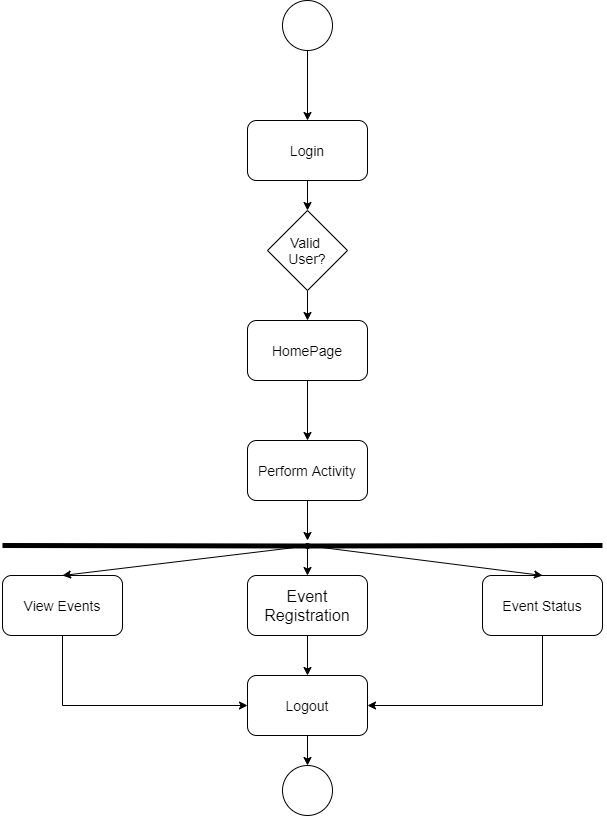


Figure 9 Client-side Activity Diagram

This diagram shows the activities clients can perform in the system. If the user is valid then the user can perform various activities such as viewing events, registering to an event.

#### Admin-side activity diagram

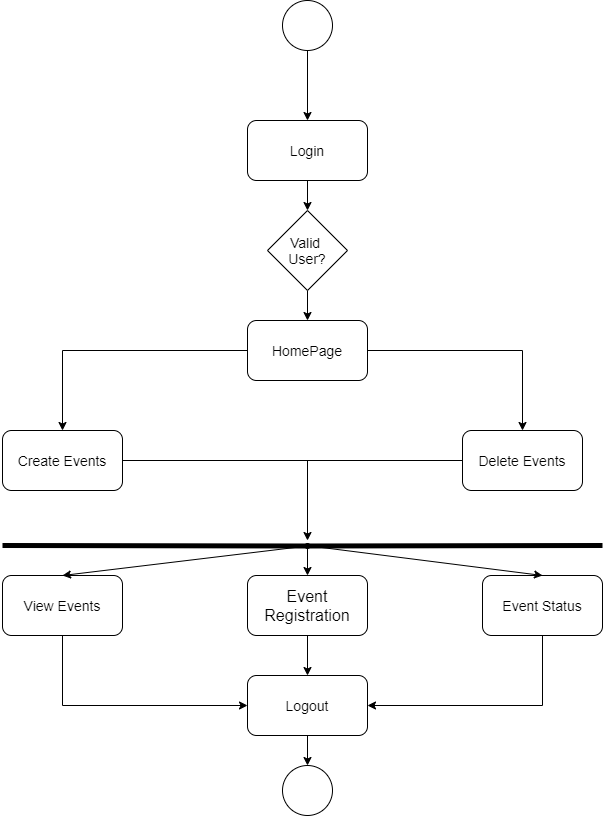


Figure 10 Admin-side Activity Diagram

This diagram shows the activities admin users can perform in the system. If the user is authenticated and authorized then the user can create or delete events and view the event and its status.

### Class Diagram

#### User Module

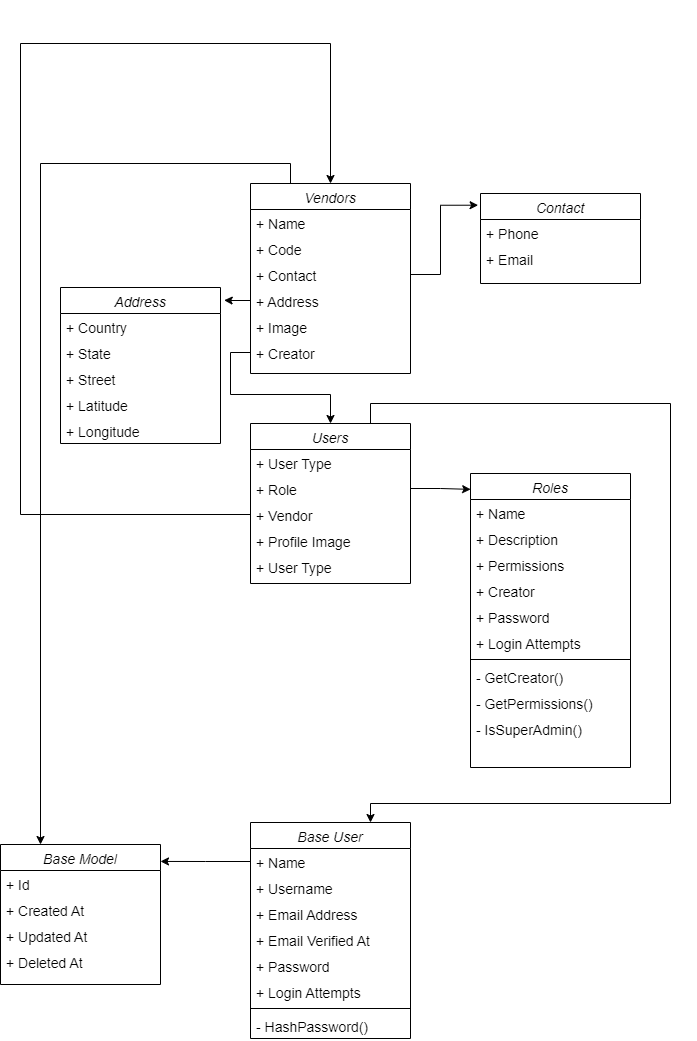


Figure 11 Class Diagram for user module

#### Event Module

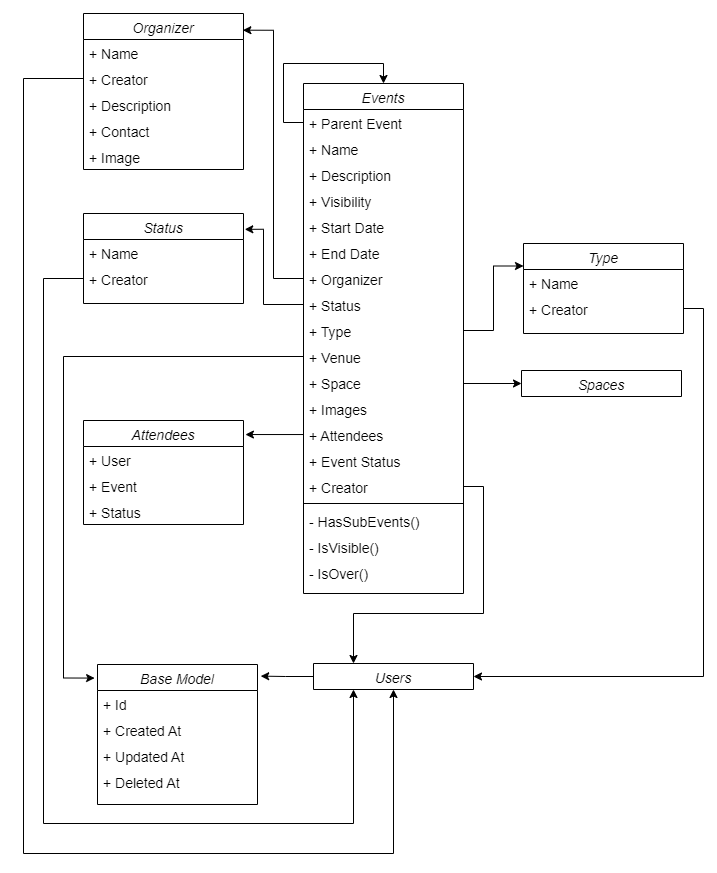


Figure 12 Class diagram for event module

#### Venue Module

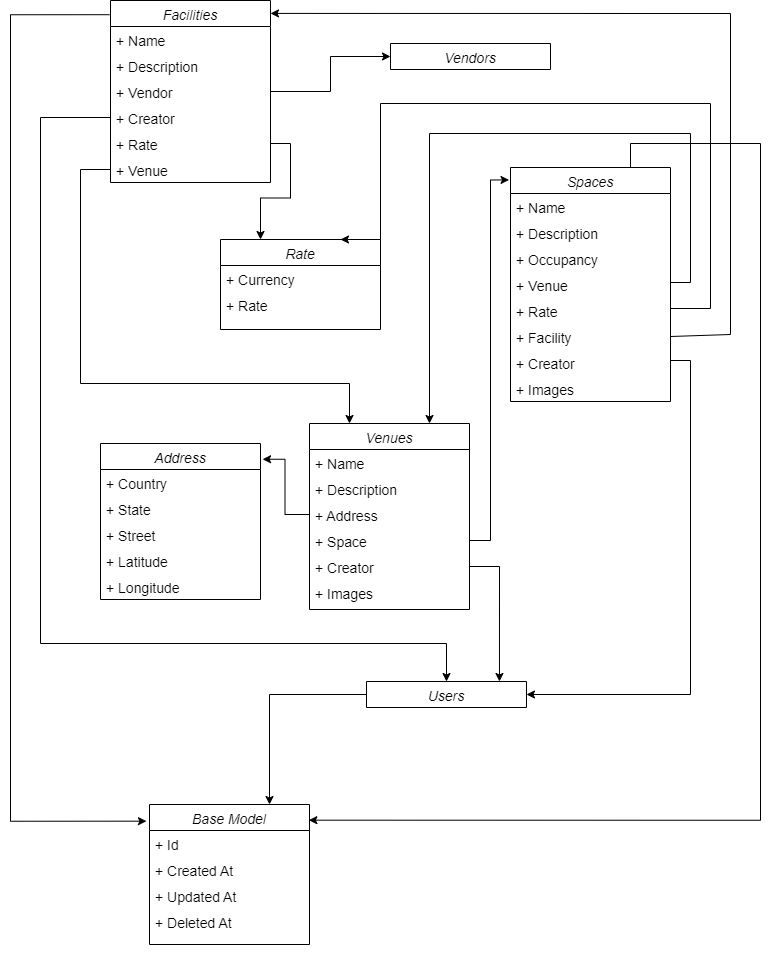


Figure 13 Class diagram for venue module

The above class diagrams show the relationship between each model. Each model extends the Base Model which consists of basic properties like id, createdAt, updatedAt, deletedAt. The user model is categorized into three types, Admin, Vendor and Customer which are indicated by the **userType** property. Similarly, the admin and vendor users are attached with roles consisting of various permissions. The permissions are stored in json type and are used as an authorization. Each vendor user is attached with their associated vendors. A single vendor can contain many users, which are allocated for managing the vendor and its respective venues and spaces. A single vendor can add multiple venues which are capable of hosting their events. Similarly, each venue can have multiple spaces and facilities, from which cost can be calculated.

A customer can either join an event or organize an event by choosing their nearby venues. Once the event is registered, the vendor receives a notification for confirmation. Similarly, the attendees are associated with multiple events, from which their history can be generated.

### Deployment Diagram

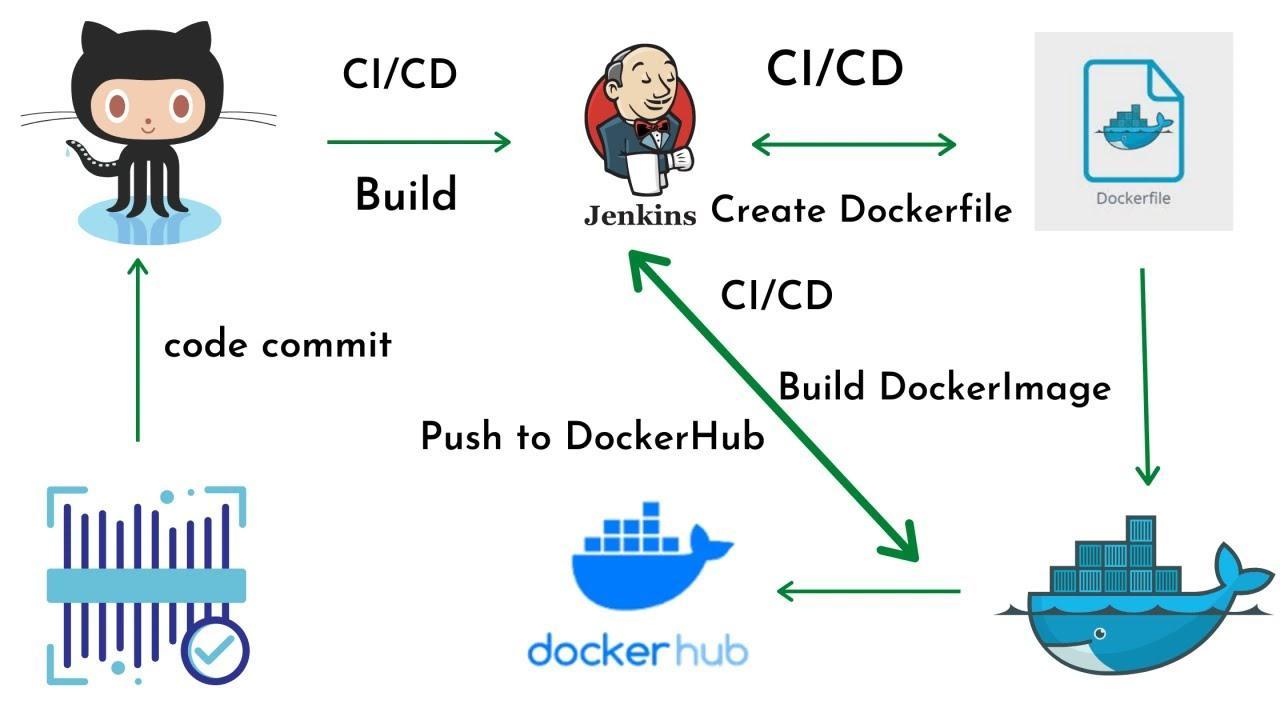


Figure 14 Deployment diagram

Problems which people face in IT Industry is long release cycle, tedious management of artifacts, rollbacks, patching software, versioning, etc. Below innovative solution describes methodologies which resolve these issues at a great extent.

**Continuous Integration (CI)** — It is a development practice that requires developers to integrate code into a shared repository several times a day. Each check-in is then verified by an automated build, allowing teams to detect problems early.

**Continuous Deployment (CD)** — It is a strategy for software releases wherein any code commit that passes the automated testing phase is automatically released into the production environment, making changes that are visible to the software’ users.

NodeJs was used to design and develop the micro-service. As soon as, developer commits his/her code in Git repository, Jenkins will compile the code using NodeJS, etc. and create a Docker image of the micro-service. After creation of the image, Jenkins stage will tag and push the image to Docker Hub or Kubernetes. So far, we’ve seen working of continuous integration pipeline. Deployment job will be triggered after successful completion of integration pipeline. This job will pull the newly created Docker image from the registry and deploy it on Kubernetes cluster using kubeconfig file.

# Implementation And Testing

## Implementations

### Tools Used

#### Frontend Tools

**HTML5:**

HTML or Hyper-Text Markup Language can be referred to as the Worldwide Web’s primary language. Most of the web pages hosted on the internet are written in some variation of HTML. HTML has seen many updates over time, and currently, the newest HTML version is HTML5. HTML5 provides full support for JavaScript to run in the background; this is possible courtesy to the JS web worker API of HTML5. There is no use to employ any JS-based or Flash work-around because there are elements inherently present in HTML5 that provide all the functionalities.

**CSS3:**

CSS3 is the advanced level of CSS2.0. In CSS3, new properties - border radius, box shadow, text shadow, multiple background images and much more. CS3 supported by all new browsers. The CSS3 version supports many more browsers than CSS2.

**JS:**

JavaScript is most commonly used as a client-side scripting language. JavaScript code is written into an HTML page. When a user requests an HTML page with JavaScript in it, the script is sent to the browser and the browser responds with the appropriate actions.

**ReactJS:**

React is a JavaScript library for building user interfaces. It is maintained by Facebook and a community of individual developers and companies. React can be used as a base in the development of single-page or mobile applications.

#### Backend Tools

**Java (Quarkus):**

Java is a general-purpose, class-based, object-oriented programming language designed for having lesser implementation dependencies. It is a computing platform for application development. Java is fast, secure, and reliable, therefore. It is widely used for developing Java applications in laptops, data centers, game consoles, scientific supercomputers, cell phones, etc.

Traditional Java stacks were engineered for monolithic applications with long startup times and large memory requirements in a world where the cloud, containers, and Kubernetes did not exist. Java frameworks needed to evolve to meet the needs of this new world.

Quarkus was created to enable Java developers to create applications for a modern, cloud-native world. Quarkus is a Kubernetes-native Java framework tailored for GraalVM and HotSpot, crafted from best-of-breed Java libraries and standards. The goal is to make Java the leading platform in Kubernetes and serverless environments while offering developers a framework to address a wider range of distributed application architectures.

**Node:**

As an asynchronous event-driven JavaScript runtime, Node.js is designed to build scalable network applications. In the following "hello world" example, many connections can be handled concurrently. Upon each connection, the callback is fired, but if there is no work to be done, Node.js will sleep.

This is in contrast to today's more common concurrency model, in which OS threads are employed. Thread-based networking is relatively inefficient and very difficult to use. Furthermore, users of Node.js are free from worries of dead-locking the process, since there are no locks. Almost no function in Node.js directly performs I/O, so the process never blocks except when the I/O is performed using synchronous methods of Node.js standard library. Because nothing blocks, scalable systems are very reasonable to develop in Node.js.

**AWS s3 (Minio):**

Amazon Simple Storage Service (Amazon S3) is an object storage service that offers industry-leading scalability, data availability, security, and performance. Customers of all sizes and industries can use Amazon S3 to store and protect any amount of data for a range of use cases, such as data lakes, websites, mobile applications, backup and restore, archive, enterprise applications, IoT devices, and big data analytics. Amazon S3 provides management features so that you can optimize, organize, and configure access to your data to meet your specific business, organizational, and compliance requirements.

Amazon S3 offers a range of storage classes designed for different use cases. For example, you can store mission-critical production data in S3 Standard for frequent access, save costs by storing infrequently accessed data in S3 Standard-IA or S3 One Zone-IA, and archive data at the lowest costs in S3 Glacier Instant Retrieval, S3 Glacier Flexible Retrieval, and S3 Glacier Deep Archive.

You can store data with changing or unknown access patterns in S3 Intelligent-Tiering, which optimizes storage costs by automatically moving your data between four access tiers when your access patterns change. These four access tiers include two low-latency access tiers optimized for frequent and infrequent access, and two opt-in archive access tiers designed for asynchronous access for rarely accessed data.

#### Database

**Postgres:**

PostgreSQL is a powerful, open-source object-relational database system that uses and extends the SQL language combined with many features that safely store and scale the most complicated data workloads. PostgreSQL has earned a strong reputation for its proven architecture, reliability, data integrity, robust feature set, extensibility, and the dedication of the open-source community behind the software to consistently deliver performant and innovative solutions. PostgreSQL runs on all major operating systems, has been ACID-compliant since 2001, and has powerful add-ons such as the popular PostGIS geospatial database extender. It is no surprise that PostgreSQL has become the open-source relational database of choice for many people and organizations.

**Redis:**

Redis is an open source (BSD licensed), in-memory data structure store, used as a database, cache, and message broker. Redis provides data structures such as strings, hashes, lists, sets, sorted sets with range queries, bitmaps, hyper logs, geospatial indexes, and streams. Redis has built-in replication, Lua scripting, LRU eviction, transactions, and different levels of on-disk persistence, and provides high availability via Redis Sentinel and automatic partitioning with Redis Cluster.

You can run atomic operations on these types, like appending to a string; incrementing the value in a hash; pushing an element to a list; computing set intersection, union and difference; or getting the member with highest ranking in a sorted set.

To achieve top performance, Redis works with an in-memory dataset. Depending on your use case, you can persist your data either by periodically dumping the dataset to disk or by appending each command to a disk-based log. You can also disable persistence if you just need a feature-rich, networked, in-memory cache.

#### Others

**Docker:**

Docker, a subset of the Moby project, is a software framework for building, running, and managing containers on servers and the cloud. The term "docker" may refer to either the tools (the commands and a daemon) or to the Docker file format.

It used to be that when you wanted to run a web application, you bought a server, installed Linux, set up a LAMP stack, and ran the app. If your app got popular, you practiced good load balancing by setting up a second server to ensure the application wouldn't crash from too much traffic.

**GraphQL:**

GraphQL is a query language for APIs and a runtime for fulfilling those queries with your existing data. GraphQL provides a complete and understandable description of the data in your API, gives clients the power to ask for exactly what they need and nothing more, makes it easier to evolve APIs over time, and enables powerful developer tools.

Send a GraphQL query to your API and get exactly what you need, nothing more and nothing less. GraphQL queries always return predictable results. Apps using GraphQL are fast and stable because they control the data they get, not the server.

GraphQL queries access not just the properties of one resource but also smoothly follow references between them. While typical REST APIs require loading from multiple URLs, GraphQL APIs get all the data your app needs in a single request. Apps using GraphQL can be quick even on slow mobile network connections.

#### Testing Tools

**Jest:**

Jest is a JavaScript testing framework maintained by Facebook, Inc., designed and built by Christoph Nakazawa with a focus on simplicity and support for large web applications. It works with projects using Babel, TypeScript, Node.js, React, Angular, Vue.js and Svelte.

**Mockito:**

Mockito is a mocking framework, JAVA-based library that is used for effective unit testing of JAVA applications. Mockito is used to mock interfaces so that a dummy functionality can be added to a mock interface that can be used in unit testing. This tutorial should help you learn how to create unit tests with Mockito as well as how to use its APIs in a simple and intuitive way.

**JUnit:**

JUnit is a unit testing framework for the Java programming language. JUnit has been important in the development of test-driven development, and is one of a family of unit testing frameworks which is collectively known as xUnit that originated with SUnit. JUnit is linked as a JAR at compile-time.

### Implementation details of modules

The proposed system is designed following a microservice architecture consisting of various services like events, blogs, user authentication and authorization programmed using Node and Java as a backend. Modern application consists of many users which directly results in many requests and responses. Handling such requests in production is very difficult as it requires more hardware resources. Replicating the same application for handling a certain traffic can be done, but replication of the whole application will require more hardware resources. But in an application only a certain feature will be overused i.e. many users will be using a certain part of the application more often. Hence while designing a microservice a clear idea of the proposed system must be done. It is very difficult to choose the granularity of the services. In the application each module is separated so that it can be easily changed into various services during production.

However, in the frontend a mono-repo architecture is used where all the Vendor application, Customer application and Admin application. The Node backend produces a GraphQL API where the response data is queried by the frontend. There are three methods provided by GraphQL namely query, mutation and subscription.

Query is used for fetching data from the backend, it uses Http GET requests.

Mutation is used for creating, updating and deleting an object, it uses Http Post requests.

Subscription is used for handling real time events like creation of an object, its deletion and more, which internally uses a WebSocket. E.g., handling notifications, chat systems and more.

In the following way:

Sample Code:

query FetchCurrentUser {

FetchCurrentUser {

id

name

username

email

image

createdAt

updatedAt

}

}

The above query is sent to the backend server, from where the requested data is wrapped inside the response. By using a Graphql API one can query required data for a certain view which will optimize the response time and response size of a request. Graphql is also useful for a highly nested/related database from which one can query the related data e.g.

query AdminFetchEvents {

AdminFetchEvents {

id

name

createdAt

updatedAt

description

visibility

startDate

endDate

image

eventStatus

organizer {

id

name

}

status {

id

name

}

type {

id

name

}

}

}

From the above query when fetching an event detail, one can also fetch details regarding organizer, status, and type. However, in REST API, 4 requests need to be made each for fetching their respective data mapping their corresponding id. This will significantly reduce the amount of server load. However, a new problem is faced in the database named N+1 problem. This problem can be solved using batching and loading.

For the storage of objects such as (photos, videos, log files, backups and container/VM images), MinIO has been implemented. It is a high-performance Object Storage API which is the most popular open-source database classified as a tool in the Cloud storage category.

Files are organized in buckets, which are logical separators for the stored data and this is passed into our application along with the access key, secret key, and the HTTP address of the Minio instance. As the MinIO process starts in the foreground by pulling the latest MinIO release from Docker, an access key and secret key is printed out which will be used in the client for authentication.

## Testing

Software testing is the process or an activity to check whether the actual results match the expected results and to ensure that the software system is free of errors. Unit Testing, Integration Testing (Blackbox and Whitebox testing), Regression Testing are carried out during the development of the software.

In this testing phase our system as a whole was tested. Every individual component was integrated and tested against user and hardware compatibility. Sometime in this testing process dependency error was found due to different local server environments and dependency conflict. To overcome this problem virtual environment and docker container were used.

### Unit testing and Test cases

Unit testing refers to the testing of every small modular component of the system, keeping them isolated from other modules. Here testing results of the various parts of the system were mentioned. The entire system is designed in a modularized form, and each module is tested via unit testing. The outcome of each module is checked. Every unitary module of the system is used.

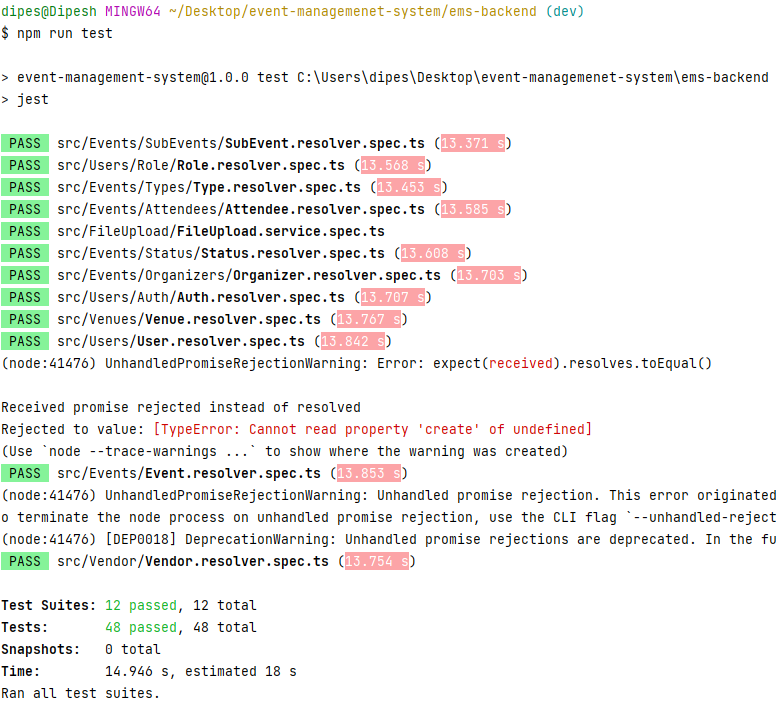


Figure 15 Result of unit testing

### Test cases for Regression testing

Table 1 Test case for Registration

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SN** | **Test Case ID** | **Test Description** | **Test Data** | **Expected Result** |
| 1 | TC-REG-01 | Registering a user in the application | Fullname:  John Doe  Username:  john\_doe  Email Address:  [johndoe@gmail.com](mailto:johndoe@gmail.com)  Password:  Qwerty@123  Confirm Password:  Qwerty@123 | Registered Successfully |
| 2 | TC-REG-02 | Registering a user with empty input fields | Fullname:  Username:  Email Address:  Password:  Confirm Password: | Fields are required error message should be displayed |
| 3 | TC-REG-03 | Registering a user with invalid Email | Fullname:  John Doe  Username:  john\_doe  Email Address:  [johndoe.com](mailto:johndoe@gmail.com)  Password:  Qwerty@123  Confirm Password:  Qwerty@123 | Invalid email error message should be displayed |
| 4 | TC-REG-04 | Registering a user with weak password | Fullname:  John Doe  Username:  john\_doe  Email Address:  [johndoe@gmail.com](mailto:johndoe@gmail.com)  Password:  Qwerty  Confirm Password:  Qwerty | Weak password error message should be displayed |
| 5 | TC-REG-05 | Registering a user with a different Password and Confirm Password. | Fullname:  John Doe  Username:  john\_doe  Email Address:  [johndoe@gmail.com](mailto:johndoe@gmail.com)  Password:  Qwerty@123  Confirm Password:  Qwerty@1233 | Password doesn’t match error should be displayed |
| 6 | TC-REG-06 | Registering a user with a username that is already registered to the application | Fullname:  John Doe  Username:  john\_doe  Email Address:  [johndoe@gmail.com](mailto:johndoe@gmail.com)  Password:  Qwerty@123  Confirm Password:  Qwerty@123 | User exists error should be displayed. |

Table 2 Test cases for Login

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SN** | **Test Case ID** | **Test Description** | **Test Data** | **Expected Result** |
| 1 | TC-LOG-01 | Logging user into the application | **Username**:  john\_doe  **Password**:  Qwerty@123 | Should be logged in successfully |
| 2 | TC-LOG-02 | Logging user into the application with invalid credentials | **Username**:  john\_doe  **Password**:  Qwerty | Invalid user credentials error should be displayed |
| 3 | TC-LOG-03 | Logging user into the application with username that does not exist in the application | **Username**:  john\_doe123  **Password**:  Qwerty | Authentication error message should be displayed |
| 4 | TC-LOG-04 | Logging user as a vendor | **Username**:  john\_doe  **Password**:  Qwerty@123  **Vendor code:**  ven\_1234 | Should log in successfully as Vendor |
| 5 | TC-LOG-05 | Logging in as aVendor with invalid vendor code | **Username**:  john\_doe  **Password**:  Qwerty@123  **Vendor code:**  ven\_0000 | Invalid Credential error should be displayed |
| 6 | TC-LOG-06 | Resetting password | **Email:**  johndoe@gmail.com | Email should be sent to the valid user with reset information |
| 7 | TC-LOG-07 | Logging in as a basic user | **Username**:  john\_doe  **Password**:  Qwerty@123 | Should be logged in successfully and should be redirected to Homepage |
| 8 | TC-LOG-08 | Logging in as an admin | **Username**:  super\_admin  **Password**:  Qwerty@123 | Should be logged in successfully and should be redirected to Admin Dashboard |
| 9 | TC-LOG-09 | Logging in as a vendor | **Username**:  vendor\_123  **Password**:  Qwerty@123 | Should be logged in successfully and should be redirected to Vendor Dashboard |

Table 3 Test case for Role

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SN** | **Test Case ID** | **Test Description** | **Test Data** | **Expected Result** |
| 1 | ADD\_ROLE | Adding a new user role. | **Name**: Admin  **Description**: Admin user.  **Permissions**: [“show-role”, “update-role”, “add-role”] | A new role should be created. |
| 2 | SHOW\_ROLE | Viewing an existing role. | **ROLE\_ID:** “asd-asdas-asd” | The selected role details are shown. |
| 3 | DELETE\_ROLE | Deleting an existing user role. | **ROLE\_ID:** “asd-asdas-asd” | The selected role should be deleted. |
| 4 | UPDATE\_ROLE | Updating an existing user role. | **ROLE\_ID:** “asd-asdas-asd”  **Name**: Admin Updated  **Description**: Admin user updated.  **Permissions**: [“show-role”, “update-role”, “add-role”] | The selected role should be updated. |
| 5 | SHOW\_ALL\_ROLE | Viewing all the created roles. |  | All the created roles should be visible. |
| 6 | GATE\_ADMIN | All the admin users can perform all the actions. |  | Admin users should be able to perform any actions. |

Table 4 Test Case for Vendor

| **SN** | **Test Case ID** | **Test Description** | **Test Data** | **Expected Result** |
| --- | --- | --- | --- | --- |
| 1 | ADD\_VENDOR | Adding a new vendor. | **Name**: Venue I  **Description**: Venue I description.  **Code:** newvendor89 | A new vendor should be created. |
| 2 | SHOW\_VENDOR | Viewing an existing vendor. | **VENDOR\_ID**: “asd-asdas-asd” | The selected vendor details are shown. |
| 3 | DELETE\_VENDOR | Deleting an existing vendor. | **VENDOR\_ID**: “asd-asdas-asd” | The selected vendor should be deleted. |
| 4 | UPDATE\_VENDOR | Updating an existing vendor. | **VENDOR\_ID**: “asd-asdas-asd”  **Name**: Venue I  **Description**: Venue I description. | The selected vendor should be updated. |
| 5 | SHOW\_ALL\_VENDOR | Viewing all the created vendors. |  | All the created vendors should be visible. |
| 6 | UPDATE\_VENDOR\_CODE | A vendor code shouldn’t be able to be updated. | **VENDOR\_ID**: “asd-asdas-asd”  **Name**: Venue I  **Description**: Venue I description.  **Code:** newvendor89 | A validation error stating vendor code cannot be updated. |

Table 5 Test Case for Event

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SN** | **Test Case ID** | **Test Description** | **Test Data** | **Expected Result** |
| 1 | ADD\_EVENT | Adding a new event. | **Name**: Event I  **Description**: Event I description.  **Start Date**: 2021/08/12  **End Date**:  2021/12/02  **Status**: “1asd-12ds32-zxd”  **Type**:  “1asd-12ds32-zxd” | A new event should be created. |
| 2 | SHOW\_EVENT | Viewing an existing event. | **EVENT\_ID**: “asd-asdas-asd” | The selected event details are shown. |
| 3 | DELETE\_EVENT | Deleting an existing event. | **EVENT\_ID**: “asd-asdas-asd” | The selected event should be deleted. |
| 4 | UPDATE\_EVENT | Updating an existing event. | **EVENT\_ID**: “asd-asdas-asd”  **Name**: Event I  **Description**: Event I description.  **Start Date**: 2021/08/12  **End Date**:  2021/12/02  **Status**: “1asd-12ds32-zxd”  **Type**:  “1asd-12ds32-zxd” | The selected event should be updated. |
| 5 | SHOW\_ALL\_EVENT | Viewing all the created events. |  | All the created events should be visible. |

Table 6 Test case for Venue

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SN** | **Test Case ID** | **Test Description** | **Test Data** | **Expected Result** |
| 1 | ADD\_VENUE | Adding a new event. | **Name**: Venue I  **Description**: Venue I description. | A new venue should be created. |
| 2 | SHOW\_VENUE | Viewing an existing event. | **VENUE\_ID**: “asd-asdas-asd” | The selected venue details are shown. |
| 3 | DELETE\_VENUE | Deleting an existing event. | **VENUE\_ID**: “asd-asdas-asd” | The selected venue should be deleted. |
| 4 | UPDATE\_VENUE | Updating an existing event. | **VENUE\_ID**: “asd-asdas-asd”  **Name**: Venue I  **Description**: Venue I description. | The selected venue should be updated. |
| 5 | SHOW\_ALL\_VENUE | Viewing all the created events. |  | All the created venues should be visible. |

Table 7 Test case for Space

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SN** | **Test Case ID** | **Test Description** | **Test Data** | **Expected Result** |
| 1 | ADD\_SPACE | Adding a new event. | **Name:** Space I  **Description:** Space I description.  **Facilities:**  [“asd-asd-asd”, “asd213-asd32”] | A new space should be created. |
| 2 | SHOW\_SPACE | Viewing an existing event. | **SPACE\_ID:** “asd-asdas-asd” | The selected space details are shown. |
| 3 | DELETE\_SPACE | Deleting an existing event. | **SPACE\_ID:** “asd-asdas-asd” | The selected space should be deleted. |
| 4 | UPDATE\_SPACE | Updating an existing event. | **SPACE\_ID:** “asd-asdas-asd”  **Name**: Space I  **Description**: Space I description.  **Facilities**:  [“asd-asd-asd”, “asd213-asd32”] | The selected space should be updated. |
| 5 | SHOW\_ALL\_SPACE | Viewing all the created events. |  | All the created spaces should be visible. |

Table 8 Test case for Facility

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SN** | **Test Case ID** | **Test Description** | **Test Data** | **Expected Result** |
| 1 | ADD\_FACILITY | Adding a new event. | **Name:** FacilityI  **Description:** Facility I description.  **Rate:** $50 | A new facility should be created. |
| 2 | SHOW\_FACILITY | Viewing an existing event. | **FACILITY\_ID:** “asd-asdas-asd” | The selected facility details are shown. |
| 3 | DELETE\_FACILITY | Deleting an existing event. | **FACILITY\_ID:** “asd-asdas-asd” | The selected facility should be deleted. |
| 4 | UPDATE\_FACILITY | Updating an existing event. | **FACILITY\_ID:** “asd-asdas-asd”  **Name:** FacilityI  **Description:** Facility I description.  **Rate:** $50 | The selected facility should be updated. |
| 5 | SHOW\_ALL\_FACILITY | Viewing all the created events. |  | All the created facilities should be visible. |

Table 9 Test Case for Attendee

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SN** | **Test Case ID** | **Test Description** | **Test Data** | **Expected Result** |
| 1 | ATTENDEE\_PUBLIC\_JOIN | An attendee can join a public event. | **USER\_ID:** “1asd-2asd”  **EVENT\_ID:**  “2sad-3asda” | The attendee should be joined in the event, but must be verified by the organizer. |
| 2 | ATTENDEE\_PRIVATE\_JOIN | An attendee cannot join a private event. | **USER\_ID:** “1asd-2asd”  **EVENT\_ID:**  “2sad-3asda” | The attendee should not be able to join in the event. |

# Conclusion and Recommendations

## Conclusion

The manual technique of planning resulted in most planners going over budget because much of the resources were focused into traveling around looking for objects and services rather than purchasing items. Traditional techniques are also inconvenient because planners must travel long distances to check on products. This is a list of functions that the system is attempting to automate. The purpose of this project was to design and develop an online event management system that would help with online event scheduling and booking. In order to attain reliability, effectiveness, and efficiency, the organization will need to use the intended system.

## Future Recommendations

The future recommendation of the project circles around maintaining information regarding:

* More advanced software for event management systems including more facilities can be given.
* The platform will be hosted on online servers to make it accessible worldwide
* Integrate multiple load balancer to distribute the loads of the system

Options were left open so any other future requirement in the system by the client for the enhancement of the system is easily implemented. In the end we would like to thank all the persons involved in the development of the system directly or indirectly.

References

|  |  |  |
| --- | --- | --- |
| [1] | Huseyin Kose, Metin Argan, Mehpare Tokay Argan, "Research Gate," 2011. [Online]. Available: https://www.researchgate.net/publication/256615203\_Special\_event\_management\_and\_marketing\_A\_case\_study\_of\_TKBL\_all\_star\_2011\_in\_Turkey. [Accessed 2021]. | |
| [2] | "LocalHop," 2021. [Online]. Available: https://getlocalhop.com/a-brief-history-of-event-management-event-planning-then-and-now/ [Accessed 2021]. | |
| [3] | V. Prajapati, "Event Management System," 2017. | |
| [4] | K. McCabe, "G2," 17 May 2018. [Online]. Available: https://www.g2.com/articles/types-event-marketing. [Accessed 2021]. | |
| [5] | "10times," [Online]. Available: https://10times.com/. [Accessed 2021]. | |
| [6] | "Nevon Projects," [Online]. Available: https://nevonprojects.com/online-event-management-system/. [Accessed 2021]. | |
| [7] | Oliver Thomas, Bettina Hermes, Peter Loos, "REFERENCE MODEL-BASED EVENT MANAGEMENT". | |
| [8] | "Wikipedia," [Online]. Available: https://en.wikipedia.org/wiki/Event\_management. [Accessed 2021]. | |
| [9] | Eventogy, "Medium," 25 July 2019. [Online]. Available: https://medium.com/eventogy/10-things-to-remember-when-planning-a-high-profile-event-9a57f84e2879. [Accessed 2021]. | |
| [10] | | A. Kumar, "PHP Gurukul," [Online]. Available: https://phpgurukul.com/event-management-system-using-php-and-mysql/. [Accessed 2021]. |

## 

## Appendix

**Sign Up and Sign in Page**

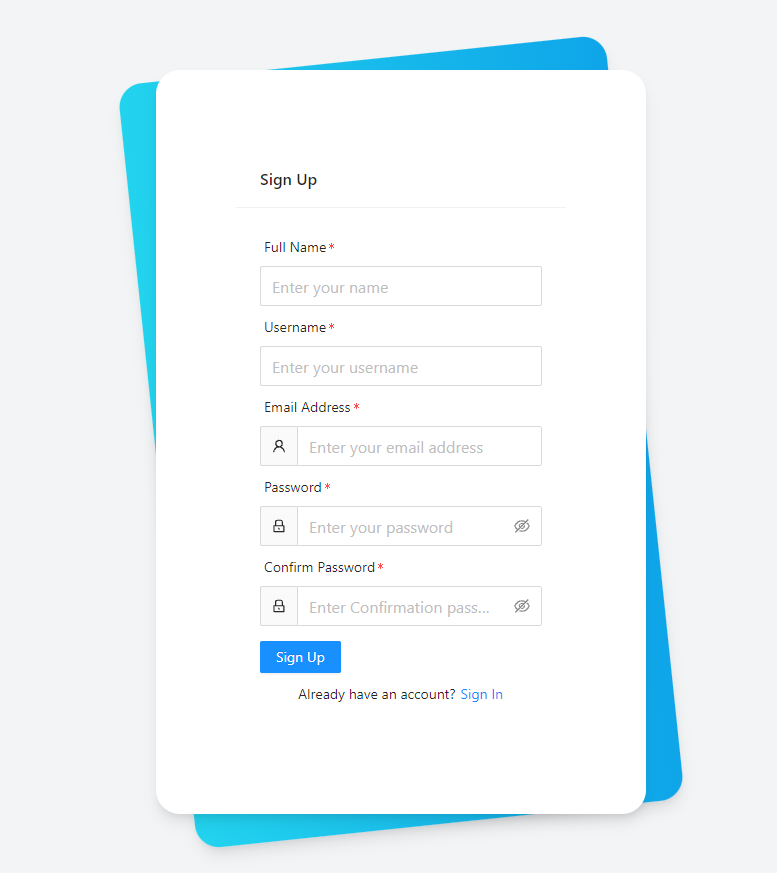


Figure 16 Signup page

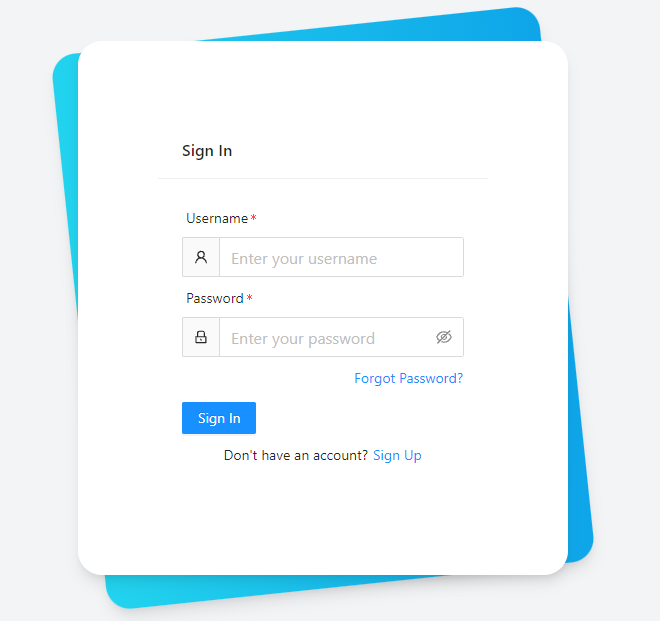


Figure 17 Sign in Page

**Admin Dashboard**

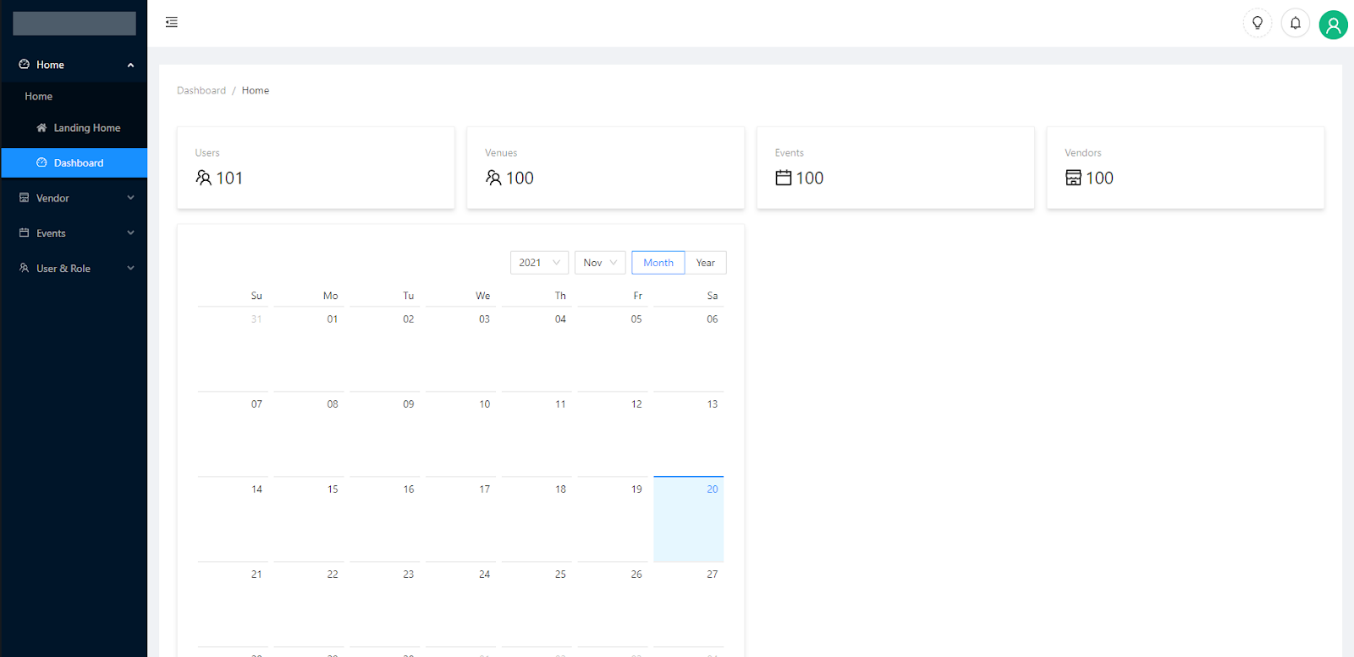
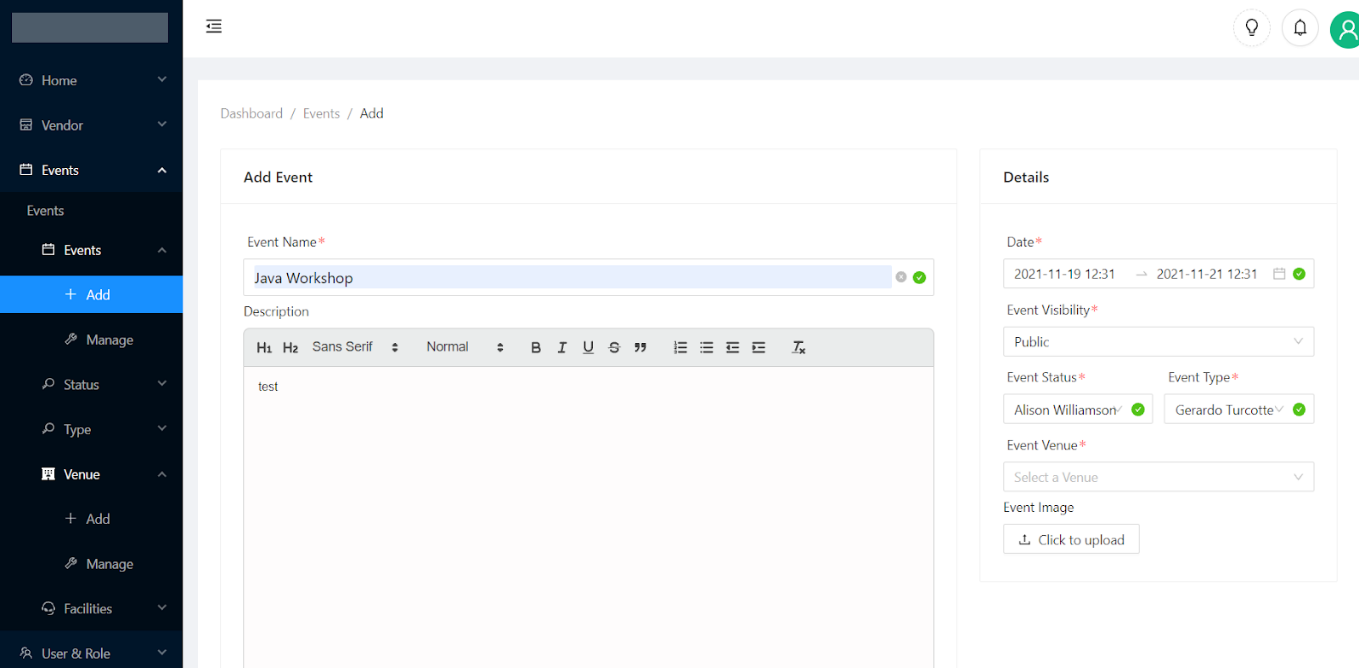


Figure 18 Admin Dashboard

**Adding an Event**



*Figure 17. Adding Events*

**Adding Venues**

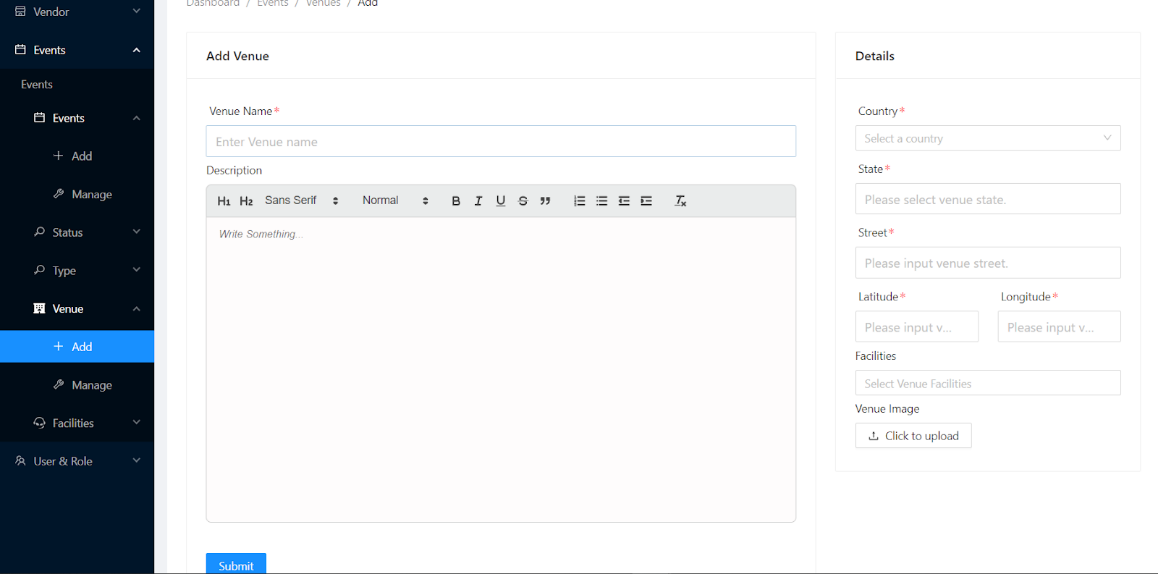


Figure 19 Adding Venues

**Adding Vendors**

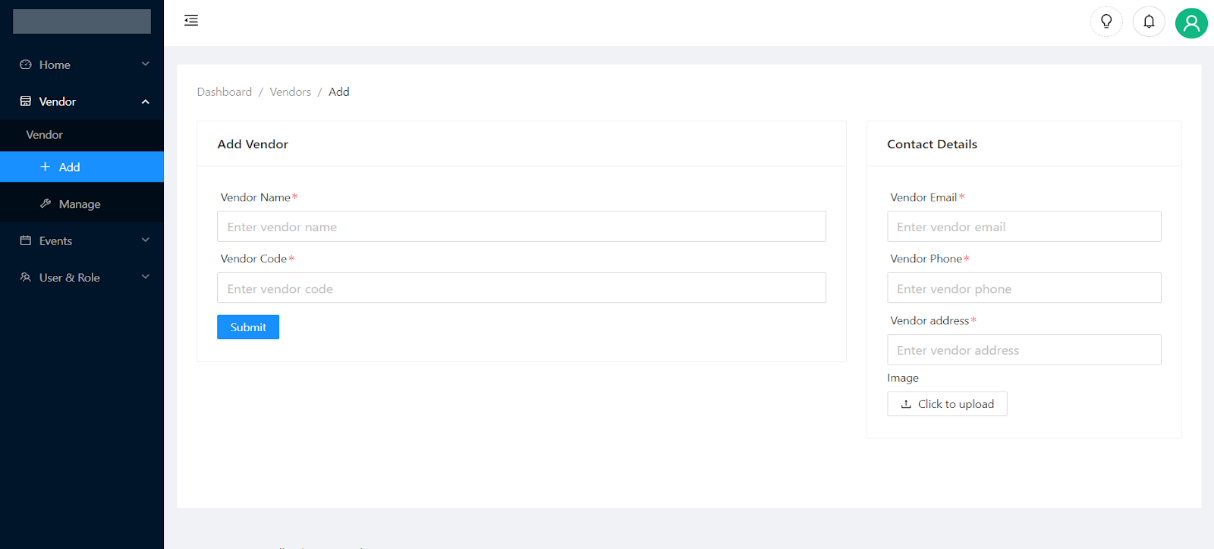


Figure 20 Adding Vendors

**Roles Management**

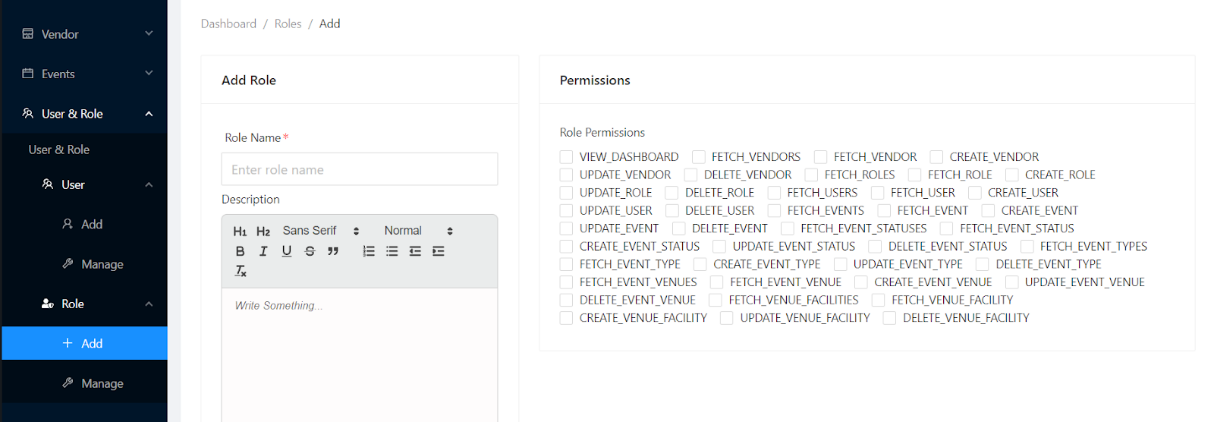


Figure 21 Roles Management

**Landing Page**

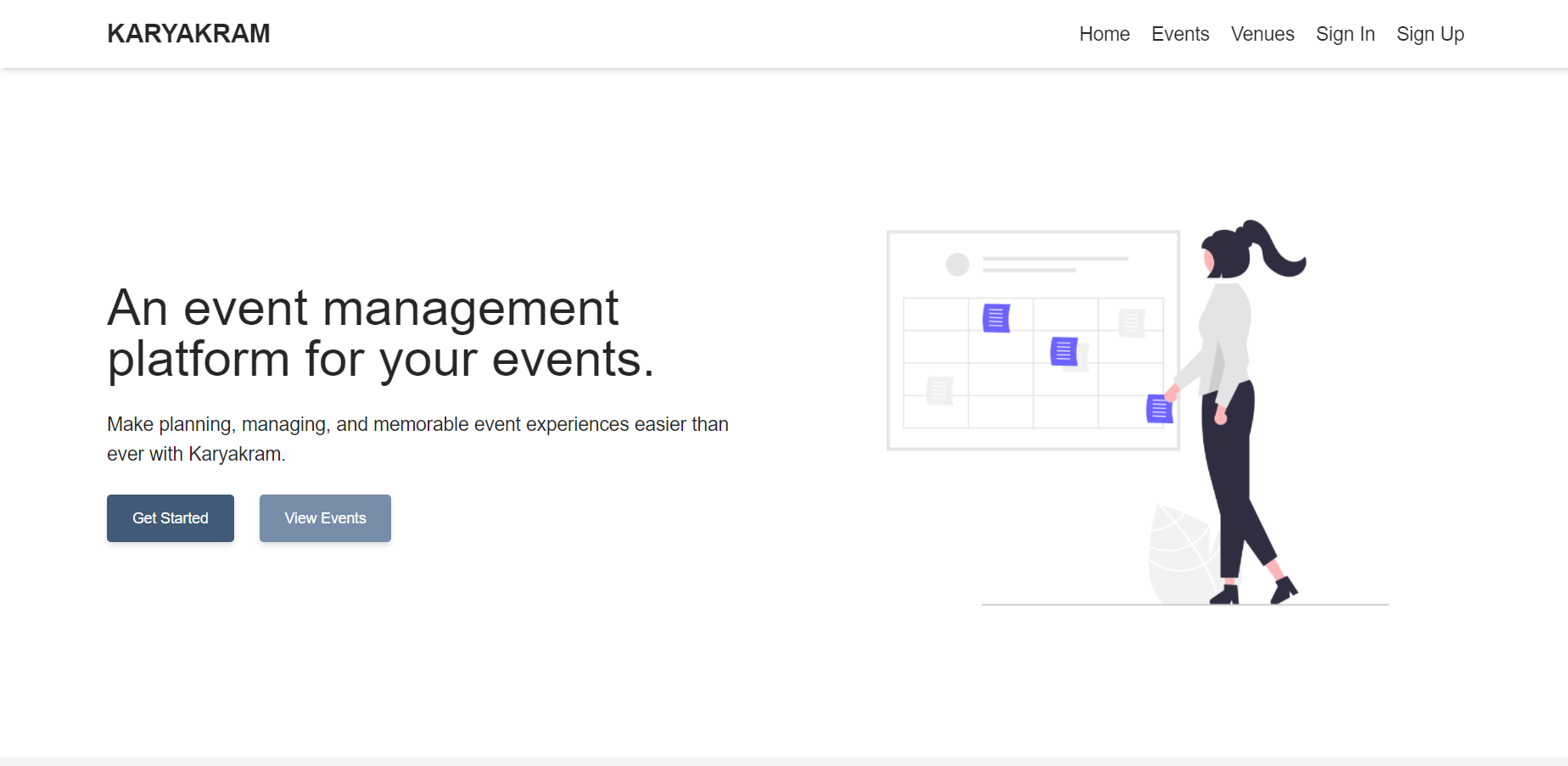


Figure 22 Landing Page