

**Visvesvaraya Technological University**  
**Belagavi, Karnataka-590 018**



**A**  
**MINI PROJECT REPORT ON**  
**“Comedy Show Booking System”**

Submitted in partial fulfilment of the requirements for the DBMS Laboratory with mini project  
(18CLS58) course of the 5<sup>th</sup> semester

**BACHELOR OF ENGINEERING**  
**IN**  
**COMPUTER SCIENCE AND ENGINEERING**

**Submitted by,**

**RAIYAN ARSH**  
1JS20CS123

**HARSH MISHRA**  
1JS20CS068

**KARUNA SAGAR**  
1JS20CS077

Under the guidance of

**Dr. Rohitaksha K**  
Associate Professor,  
Dept of CSE  
JSSATE, Bengaluru



**Mrs. Snehalatha N**  
Assistant Professor,  
Dept of CSE  
JSSATE, Bengaluru

**JSS ACADEMY OF TECHNICAL EDUCATION, BENGALURU**  
**Department of Computer Science and Engineering**  
**2022-2023**

JSS MAHAVIDHYAPEETHA, MYSURU

# JSS Academy of Technical Education

JSS Campus, Uttrahalli Kengeri Main road, Bengaluru – 560060

## Department of Computer Science and Engineering



### CERTIFICATE

This is to certify that the mini-project work entitled "**COMEDY SHOW BOOKING SYSTEM**" is a Bonafide work carried out by **Mr.RAIYAN ARSH (1JS20CS123)**, **Mr. HARSH MISHRA (1JS20CS068)** and **Mr.KARUNA SAGAR(1JS20CS077)** in partial fulfillment for the **Database Management Systems Laboratory with Mini Project (18CSL58)** of 5th semester **Bachelor of Engineering in Computer Science and Engineering of the Visvesvaraya Technological University, Belgaum** during the academic year 2022-2023. It is certified that all corrections and suggestions indicated for Internal Assessment have been incorporated in the report deposited in the department library. The project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the said degree.

**Dr. Rohitaksha K**

Associate Professor,  
Dept of CSE  
JSSATE, Bengaluru

**Mrs. Snehalatha N**

Assistant Professor,  
Dept of CSE  
JSSATE, Bengaluru

**Dr. P B Mallikarjuna**

Associate Professor & Head  
Dept of CSE  
JSSATE, Bengaluru

NAME OF THE EXAMINERS

SIGNATURE WITH DATE

1).....

.....

2).....

.....

## ACKNOWLEDGEMENT

We express our humble pranamas to His Holiness Jagadguru **Sri Sri Sri Shivarathri Deshikendra Mahaswamiji** who has showered their blessings on us for framing our career successfully.

We express our sincere thanks to our beloved principal, **Dr. Bhimasen Soragaon** for having supported us in our academic endeavors.

We are also indebted to **Dr. P B Mallikarjuna**, Head of Department of Computer Science and Engineering for the facilities and support extended towards us.

We are thankful to the resourceful guidance, timely assistance and graceful gesture of our guide **Dr. ROHITAKSHA K**, Associate Professor, Department of Computer Science and Engineering and **Mrs. Snehalatha N**, Assistant Professor, Department of Computer Science and Engineering, who has helped us in every aspect of our project work.

And last but not the least, we would be very pleased to express our heart full thanks to all the **teaching and non-teaching staff of CSE department** and our **friends** who have rendered their help, motivation and support.

The completion of any project involves the efforts of many people. We have been lucky enough to have received a lot of help and support from all quarters during the making of this project, so with gratitude, we take this opportunity to acknowledge all those who have given guidance and encouragement thereby helping us emerge successfully.

RAIYAN ARSH

1JS20CS123

HARSH MISHRA

1JS20CS068

KARUNA SAGAR

1JS20CS077

# ABSTRACT

A comedy show ticket booking system is a software application that allows users to purchase tickets for comedy shows in advance. The system is designed to be user-friendly and efficient, making it easy for customers to find and book the comedy show of their choice. The system is typically accessed through a website or mobile application, and customers can browse through a list of comedy shows, view showtimes and select the seats of their choice.

The system allows customers to choose the comedy show they want to watch, the date and time of the screening, and the number of tickets they want to purchase. The system then provides a list of available seats, allowing customers to select the seats they want. Once the customer has selected the seats, they can proceed to the checkout page where they can enter their personal information and make the payment.

The system also allows customers to cancel or change their booking if needed. Customers can log into their account and view their booking history, and if they need to make any changes, they can do so through the system. This feature is particularly useful for customers who may have made a mistake while booking or who may have changed their mind about the comedy show they want to watch.

The comedy show ticket booking system also provides valuable data and analytics to comedy show operators. They can use this data to track customer behavior and preferences, analyze ticket sales and occupancy rates, and make informed decisions about which comedy shows to show and when. This data can also be used to optimize pricing strategies and to develop targeted marketing campaigns.

# TABLE OF CONTENTS

CHAPTER	TITLE	PAGE NO.
	ACKNOWLEDGEMENT ABSTRACT TABLE OF CONTENTS TABLE OF FIGURES	
1	INTRODUCTION	2
2	DESIGN 1. ER DIAGRAM 2. RELATIONAL SCHEMA	3 6 8
3	SYSTEM REQUIREMENTS 3.1 FRONT END 3.2 BACK END 3.3 WEB SERVER 3.4 OPERATING SYSTEM 3.5 FUNCTIONAL AND NON-FUNCTIONAL REQ 3.6 CLASS DIAGRAM	9 9 10 11 12 12 13
4	IMPLEMENTATION 4.1 CODE SNIPPET 4.2 TABLE DESCRIPTION 4.3 TABLE CREATION 4.4 INSERTION OF TUPLES 4.5 QUERIES 4.6 TRIGGERS	14 14 18 20 23 25 27
5	RESULTS 5.1 FRONT END 1. CUSTOMER SIDE 2. ADMIN SIDE	28 28 29 31
6	CONCLUSION	34
7	REFERENCES	36

## **Chapter 1. INTRODUCTION**

A database management system (DBMS) is a software application that allows users to create, manage and maintain a database. It provides a way to store, organize and retrieve data in an efficient and secure manner. A DBMS allows users to interact with the data using a specific language called SQL (Structured Query Language) which enables them to perform tasks such as inserting, updating, and retrieving data from the database. In summary, DBMS provides a reliable and efficient way to store, manage and retrieve data, and it is widely used in various industries like healthcare, finance, retail and more. It enables organizations to make data-driven decisions, automate processes and improve the overall performance of their business operations.

Comedy show Ticket Booking system is a database project implemented using HTML, CSS, JAVASCRIPT, (For frontend) & PHP and MySQL (For backend). It primarily allows customers to reserve seats and purchase tickets for upcoming films online or at a box office with ease.

A comedy show ticket booking system is a platform that allows users to purchase tickets for comedy shows. The system can be accessed through a website or a mobile application and typically includes features such as browsing showtimes, selecting seats, and making payments. The system can also include additional features such as trailers, reviews, and ratings to help users make informed decisions about which comedy shows to see.

Many comedy show ticket booking systems also offer the option to save payment information for faster checkout and the ability to view and manage upcoming reservations. The system also allows event owners to manage their inventory and showtimes, as well as keep track of customer information for marketing purposes. With the convenience of online booking, it has become increasingly popular among comedy show-goers, as it eliminates the need to physically stand in long queues for tickets.

Additionally, it allows for a more efficient and streamlined process for both the customer and event staff. This system is a great way for comedy show-goers to plan their comedy show-watching experience and for theater owners to increase their revenue.

## Chapter 2: DESIGN

### Entity types:

An entity type is a category of objects or things represented in a database, where each instance of the entity type is represented by a row in the table and defined by a set of attributes

For example: in our database there exists an entity type 'event'

### Entity set:

An entity set is a collection of all entities of a particular entity type in a database. It is represented by a table in the database, where each row corresponds to an instance of the entity type. An entity set can be thought of as a set of all entities of a specific type.

### Attributes:

Attributes are characteristics or properties of an entity in a database. They are used to describe the entity and are typically represented as columns in a table. Each attribute has a name and a data type, such as text, numbers, date, etc. An attribute can be considered as a descriptor of the entity.

### Types of Attributes:

#### single-valued Attributes:

Most attributes have a single value for a particular entity For eg:**Age** is a single-valued attribute of a person

In our database all the attributes used are single-valued attributes

#### Multivalued Attributes:

An entity having multiple values for that attribute

For eg: color of a color color={black,red}

Person's degree degree={BE, MTech, PhD}

In our database there are no multivalued attributes.

### **Stored and Derived attributes:**

Two (or more) attribute values are related—for eg: the Age and Birth\_date attributes of a person

The value of Age can be determined from the current (today's) date and the value of that person's Birth\_date

The Age attribute is hence called a **derived** attribute

Birth\_date attribute is called a **stored** attribute

### **NULL Value Attributes:**

In some cases, a particular entity may not have an applicable value for an attribute.

For eg: the Apartment\_number attribute of an address applies only to addresses that are in apartment buildings and not to other types of residences, such as single-family homes College\_degrees attribute applies only to people with college degrees

### **Complex Attributes:**

composite and multivalued attributes can be nested arbitrarily  
arbitrary nesting by grouping components of a composite attribute between parentheses ( ) and separating the components with commas, and by displaying multivalued attributes between braces { }. Such attributes are called complex attributes

For example, if a person can have more than one residence and each residence can have a single address and multiple phones, an attribute Address\_phone for a person

## **RELATIONSHIP TYPES**

**One-to-one:** A relationship in which one record in a table is related to one and only one record in another table.

In Fig 2.1 one bill can have only one discount

**One-to-many:** A relationship in which one record in a table is related to multiple records in another table.

In Fig 2.1 one customer can have more than one bookings for a event

**Many-to-many:** A relationship in which multiple records in a table are related to multiple records in another table.

In Fig 2.1 many customers can book many shows

**Self-referencing:** A relationship in which a table is related to itself.



**Hierarchical:** A relationship in which one record in a table is related to one or more records in the same table, creating a parent-child relationship.

**Associative:** A relationship in which two or more tables are related through an additional table, also known as a join table or junction table.

### **Structural constraints:**

Structural constraints are rules and limitations that are placed on the design and structure of a database in order to maintain its integrity and consistency. These constraints are used to define the relationships between tables and columns, and to ensure that data is entered and stored in a consistent and predictable manner.

In Fig 2.1 we have a structural constraint on the table 'mechanics\_list' with total participation as participation constraint and cardinality ratio as 1 meaning only one mechanic can be assigned for one service request.

### **Min Max Notation:**

In an Entity-Relationship (ER) diagram, the "min" and "max" notation is used to indicate the minimum and maximum cardinality of a relationship between two entities. Cardinality refers to the number of instances of one entity that can be associated with each instance of another entity.

In Fig 2.1 there exists a min max notation like (n,1) between 2 entity types 'service\_requests' and 'mechanics\_list' joined by the relation 'request\_meta' meaning many service\_requests can be handled by one mechanic.

## 2.1 ER-Diagram:

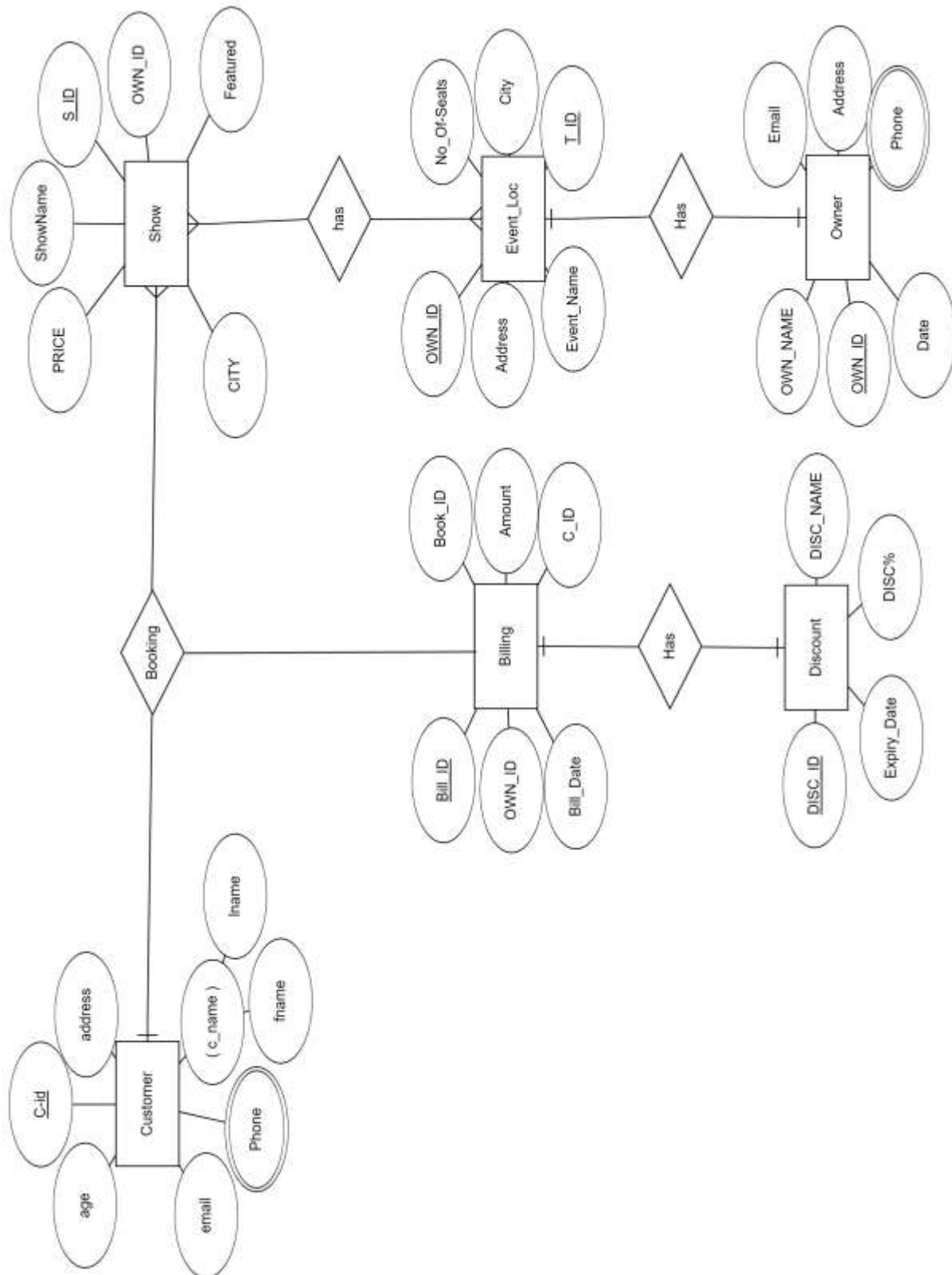


Fig 2.1: ER- Diagram

## 2.2 ER-to-Relational mapping:

**Step 1:** For each **regular (strong) entity type** E in the ER schema, create a relation R that includes all the simple attributes of E.

**Step 2:** For each **weak entity type** W in the ER schema with owner entity type E, create a relation R, and include all simple attributes (or simple components of composite attributes) of W as attributes. In addition, include as foreign key attributes of R the primary key attribute(s) of the relation(s) that correspond to the owner entity type(s).

**Step 3:** For each **binary 1:1 relationship type** R in the ER schema, identify the relations S and T that correspond to the entity types participating in R. Choose one of the relations, say S, and include the primary key of T as a foreign key in S. Include all the simple attributes of R as attributes of S.

**Step 4:** For each regular **binary 1:N relationship type** R identify the relation (N) relation S. the primary key of T as a foreign key of S. Simple attributes of R map to attributes of S.

**Step 5:** For each **binary M:N relationship type** R, create a relation S. Include the primary keys of participant relations as foreign keys in S. Their combination will be the primary key for S. Simple attributes of R become attributes of S.

**Step 6:** For each **multi-valued attribute A**, create a new relation R. This relation will include an attribute corresponding to A, plus the primary key K of the parent relation (entity type or relationship type) as a foreign key in R. The primary key of R is the combination of A and K.

**Step 7:** For each **n-ary relationship type R**, where  $n > 2$ , create a new relation S to represent R. Include the primary keys of the relations participating in R as foreign keys in S. Simple attributes of R map to attributes of S. The primary key of S is a combination of all the foreign keys that reference the participants that have cardinality constraint  $> 1$ . For a recursive relationship, we will need a new relation.

## 2.3 Relational Schema

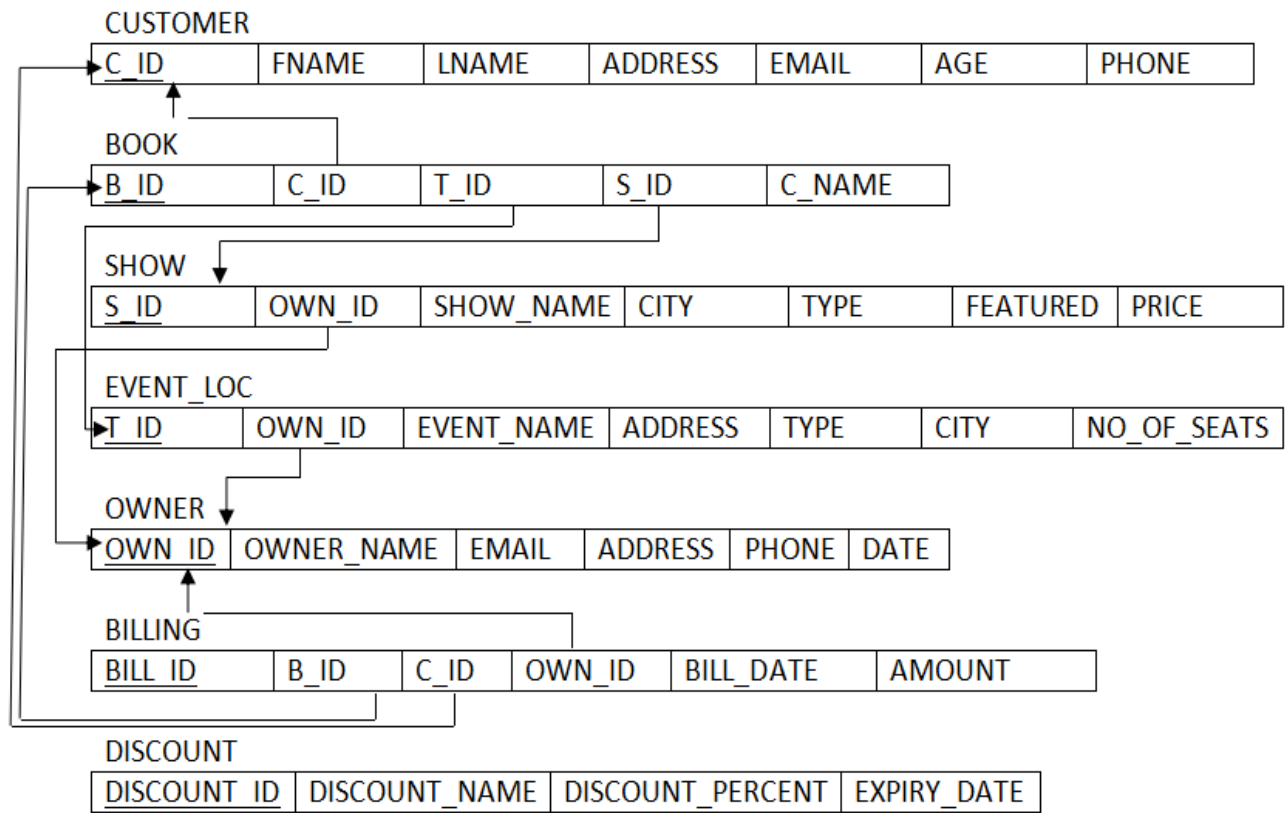


Fig 2.2: Relational Schema

## CHAPTER 3: SYSTEM REQUIREMENTS

### 3.1 Front End

**HTML:** HTML (Hypertext Markup Language) is the standard language used to create web pages. It uses a system of tags and attributes to structure and format the content of a webpage, including text, images, and links. HTML documents are viewed in web browsers and are rendered into a visual representation of the page, which can include text, images, videos, and interactive elements. HTML is the foundation of all websites and is used in conjunction with other languages such as CSS and JavaScript to create dynamic and interactive web pages.

**CSS:** CSS (Cascading Style Sheets) is a language used to control the presentation and layout of HTML documents. It allows developers to separate the presentation of a webpage from its structure and content, defined in HTML. With CSS, you can control the colors, fonts, spacing, and overall layout of a webpage, as well as add visual effects such as hover states, animations, and transitions. CSS can be written in separate files or included in the same document as the HTML, and it can be applied to individual elements or groups of elements on a webpage. It's a powerful tool that allows developers to create visually appealing and consistent designs across multiple web pages and devices.

**JAVASCRIPT:** JavaScript is a programming language that is primarily used to create interactive front-end web applications and dynamic website content. It is a scripting language that runs in web browsers and enables developers to create interactive elements such as forms, animations, and other dynamic features. JavaScript is also commonly used on the back-end, through technologies such as Node.js, to create server-side applications. It is a widely-used and versatile language that is supported by all modern web browsers and is easy to learn for both beginners and experienced programmers.

**jQuery:** jQuery is a fast, small, and feature-rich JavaScript library. It makes HTML document traversal and manipulation, event handling, and animation much simpler with an easy-to-use API that works across a multitude of browsers. With a

combination of versatility and extensibility, jQuery has changed the way that millions of people write JavaScript

**Ajax:** Ajax (short for Asynchronous JavaScript and XML) is a set of web development techniques used for creating interactive and responsive web applications. It allows for the creation of dynamic web pages without the need for a page refresh. This is achieved by using JavaScript to make asynchronous requests to a server, and updating only the relevant parts of the page with the new data. This allows for a smoother, faster user experience and enables web developers to create more complex and interactive applications. jQuery provides a convenient API for making Ajax requests, which makes it easy to use in a web development project.

**Bootstrap:** Bootstrap is a free and open-source front-end development framework that helps developers create responsive, mobile-first web pages and web applications. It is a collection of HTML, CSS, and JavaScript components and tools that are designed to be used in conjunction with the Bootstrap CSS framework. Bootstrap provides a consistent, easy-to-use set of classes and styles that can be used to create common web design elements such as forms, buttons, navigation bars, and more. It also includes JavaScript plugins for common web development tasks such as creating modals, carousels, and other interactive elements. One of the benefits of Bootstrap is that it makes it easy to create responsive designs that look and function well on a wide range of devices, from desktop computers to smartphones

## 3.2 Back End

**PHP v8.0.7:** dynamic and interactive websites. PHP is executed on the server, and the results are sent to the browser as plain HTML. This allows PHP to handle tasks such as reading and writing to files, sending and receiving cookies, and creating and manipulating databases.

PHP can be integrated with a variety of web development technologies, including HTML, CSS, and JavaScript. It is often used in conjunction with other web development technologies such as MySQL (a popular open-source database

management system) to create dynamic and interactive web applications. PHP is also often used for creating RESTful web services, that can be consumed by other systems.

PHP is widely supported by web hosting providers, making it easily accessible and easy to run on a variety of platforms, including Windows, Linux, and MacOS. It's one of the most popular server-side scripting language and has a large community support and many frameworks built on top of it, like Laravel, CodeIgniter, CakePHP and many more.

**MySQL Database:** MySQL is a popular open-source relational database management system (RDBMS) that is used to store, organize, and retrieve data in a structured manner. It is widely used in web applications to store and manage data such as user information, website content, and other data.

MySQL uses a structured query language (SQL) to manage the data in the databases, which allows for tasks such as creating tables, inserting and updating data, and retrieving data based on specific criteria. MySQL supports various data types including integers, floating-point numbers, strings and date and time, and it also supports advanced features such as stored procedures, triggers, views, and indexes which are used to optimize the performance of data retrieval.

MySQL can be used on a variety of platforms, including Windows, Linux, and MacOS. It's widely supported by a large community and has a wide range of tools and libraries for management, administration and monitoring. It's also known for its high performance and scalability. It's often used in conjunction with other web development technologies such as PHP to create dynamic and interactive web applications.

### 3.3 Web Server

**XAMPP v3.3.0:** XAMPP is a free, open-source, and cross-platform web server solution that allows developers to create and test web applications on their local machines. XAMPP stands for Cross-Platform (X), Apache (A), MariaDB (M), PHP (P) and Perl (P). It is a package of software components that includes Apache web server, MariaDB (MySQL) database, PHP, and Perl programming languages. This

bundle allows developers to set up a local web server environment with the necessary components to run PHP and MySQL-based web applications without the need for an internet connection.

XAMPP is easy to install and use, and it is particularly popular among developers working on WordPress, Joomla, and other PHP-based web applications. With XAMPP, developers can test their code and see how it behaves on a live web server, without having to upload it to a remote web server. This can save time and money, and it also allows developers to work offline. my local webserver that has a PHP Version 8.0.7:

## **4. Operating System**

- **Minimum Windows 7**

## **5. Functional and non-functional requirements**

Functional requirements of a Comedy show ticket booking system include:

1. User registration and login
2. Search for Comedy shows by title, genre, release date, etc.
3. View Comedy show details, including synopses and showtimes
4. Select seats and purchase tickets
5. View and print tickets
6. Cancel or modify existing bookings

Non-functional requirements of a Comedy show ticket booking system include:

1. Performance: The system should be able to handle a large number of concurrent users and transactions without significant delays.
2. Security: The system should protect sensitive user information, such as credit card details, from unauthorized access.
3. Scalability: The system should be able to handle an increase in users and transactions as the business grows.
4. Reliability: The system should be available to users at all times, with minimal downtime for maintenance or updates.
5. Usability: The system should be easy to use and understand, with a clear and intuitive user interface



### 3.6 CLASS DIAGRAM

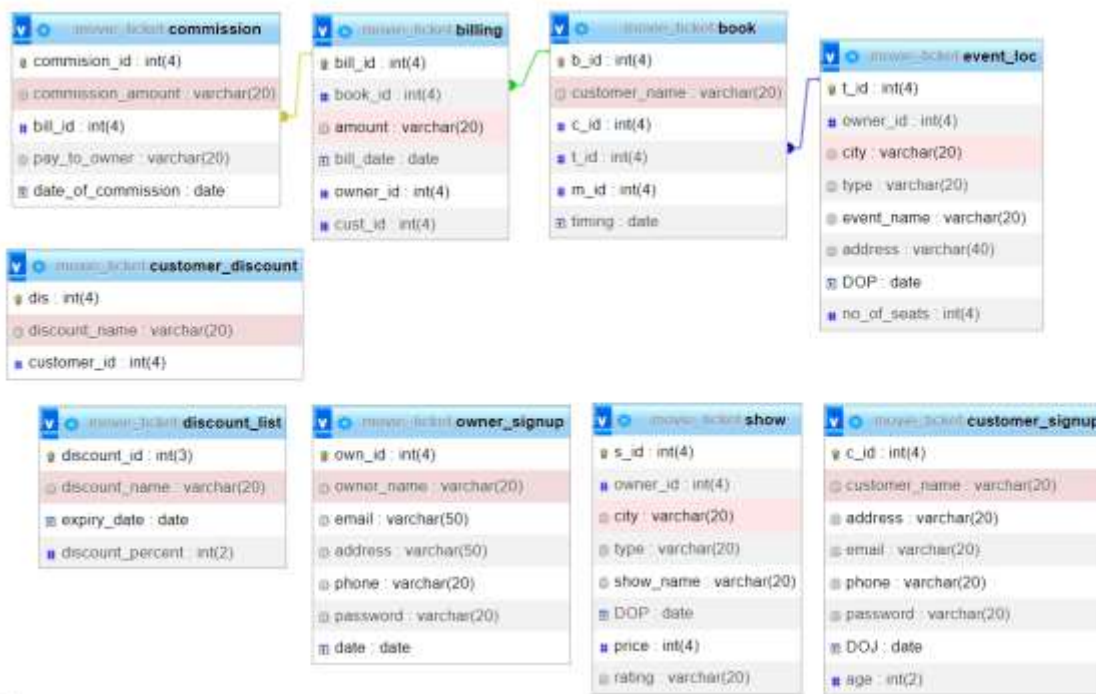


Fig 3.1 Class diagram

A class diagram is a type of diagram in software engineering that describes the structure of a system by showing the classes, attributes, and relationships between them. The Fig 3.1 describes a class diagram for our database system.

The class diagram has seven classes: Owner\_Signup, Customer\_signup, billing, Show, Event, Commission and Discount. Each class has a set of attributes, such as id, name, and status, and a primary key.

## CHAPTER 4: IMPLEMENTATION

### 4.1 CODE SNIPPET

#### HOME PAGE CODE SNIPPET

```

<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Main page</title>

  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/tailwindcss@2.2.19/dist/tailwind.min.css" />
  <link rel="stylesheet" href="https://fonts.googleapis.com/css2?family=Inter:wght@300;400;500;600;700&display=swap" />
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/tw-elements/dist/css/index.min.css" />
  <script src="https://cdn.jsdelivr.net/npm/tw-elements/dist/js/index.min.js"></script>
  <script src="https://cdn.tailwindcss.com"></script>

</head>

<!-- login card body -->
<div class="flex"
  style="display: flex; flex-direction: row; margin: 50px 50px 50px 50px; justify-content: space-
  evenly">
  <div class="flex justify-center">
    <div class="rounded-lg shadow-lg bg-white max-w-sm" style = "background:rgb(31, 37, 51)">
      <a href="\covid\owner_login.php">
        
      </a>
      <div class="p-6" style = "display: flex; justify-content: center;" >
        <!-- <h5 class="text-gray-900 text-xl font-medium mb-2">Card title</h5> -->
        <!-- <p class="text-gray-700 text-base mb-4">
          Some quick example text to build on the card title and make up the bulk of the card's
          content.
        </p> -->
        <a href="\show\owner_login.php">
          <button type="button"

```

```

        class=" inline-block px-6 py-2.5 bg-blue-600 text-white font-medium text-xs leading-tight
uppercase rounded shadow-md hover:bg-blue-700 hover:shadow-lg focus:bg-blue-700 focus:shadow-lg
focus:outline-none focus:ring-0 active:bg-blue-800 active:shadow-lg transition duration-150 ease-in-
out" style = "border-radius: 50px;background: rgb(134 135 137)">Owner Login</button>
    </a>
</div>
</div>
</div>

<div class="flex justify-center">
    <div class="rounded-lg shadow-lg bg-white max-w-sm" style = "background:rgb(31, 37, 51)">
        <a href="\covid\login_page.php">
            
        </a>
        <div class="p-6" style = "display: flex;justify-content: center;" >
            <!-- <h5 class="text-gray-900 text-xl font-medium mb-2">Card title</h5> -->
            <!-- <p class="text-gray-700 text-base mb-4">
                Some quick example text to build on the card title and make up the bulk of the card's
                content.
            </p> -->
            <a href="\show\login_page.php">
                <button type="button"
                    class=" inline-block px-6 py-2.5 bg-blue-600 text-white font-medium text-xs leading-
tight uppercase rounded shadow-md hover:bg-blue-700 hover:shadow-lg focus:bg-blue-700
focus:shadow-lg focus:outline-none focus:ring-0 active:bg-blue-800 active:shadow-lg transition
duration-150 ease-in-out" style = "border-radius: 50px;background:rgb(134 135 137)">Customer
Login</button>
            </a>
        </div>
    </div>
</div>
</div>

```

## LOGIN PAGE CODE SNIPPET

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="owner_login.css">
    <title>Owner login</title>
</head>
<body >
    <div class="center">
        <h1>Owner Login</h1>
        <form action="http://localhost/covid/owner_login.php" method="POST">

            <div class="text_field">
                <input type="text" required name="Username">
                <span></span>
                <label>User</label>
            </div>

```

```
<div class="text_field">
    <input type="password" required name="Password">
    <span></span>
    <label>password</label>
</div>
<div class="pass">forget password?</div>
<input type="submit" name="save" value="login">
<div class="signup">
    <!-- Not a member?<input type="submit" name="signup1" value="sign up"> -->
    Not a member?
    <a href="owner_signup.html" name="signup1">signup</a>
</div>
</form>
</div>
<div class="img">
    
</div>
</body>
</html>
```

## BOOKING PAGE CODE SNIPPET

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" rel="stylesheet">
    <link href="https://getbootstrap.com/docs/5.2/assets/css/docs.css" rel="stylesheet">
    <title>Bootstrap Example</title>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js"></script>
    <link rel="stylesheet" href="toBook1.css">
    <title>Customer main page</title>
</head>
<body>
    style="background-image : url('t1.jpeg'); background-size : cover; background-repeat : no-
repeat; backdrop-filter: blur(1px);">
    <div class="center" >
        <h1>Book Show</h1>
        <div class="failed">
            <span>
                <?php
                    if($flag==1)
                    {
                        echo"Booked successfully!!!";
                    }
                ?>
            </span>
        </div>
    </div>
```

## BOOKING DETAILS CODE SNIPPET

```

</head>

<body style="background-image : url('bg1.jpg'); background-size : cover; background-repeat : no-
repeat; backdrop-filter: blur(0px);">
  <div class="center" style="  background:lightblue">
    <h1 style="  background: rgb(31,37,51);
    color: white;">Booking Details</h1>

    <table class="table" style="border:black">
      <thead>
        <tr>
          <!-- <th scope="col">sno</th>
          <th scope="col">Name</th>
          <th scope="col">City</th>
          <th scope="col">Show Name</th>
          <th scope="col">Rating</th>
          <th scope="col">Event Name</th>
          <th scope="col">Event Address</th>
          <th scope="col">Timing</th>
          <th scope="col">Price</th>
          <th scope="col">Cancel</th>
        </tr>
      </thead>
      <tbody>
        <tr>

          <?php
            while($row = mysqli_fetch_assoc($result))
            {
              ?>
              <td><?php echo $row['city']; ?></td>
              <td><?php echo $row['type']; ?></td>
              <td><?php echo $row['Show_name']; ?></td>
              <td><?php echo $row['rating']; ?></td>
              <td><?php echo $row['event_name']; ?></td>
              <td><?php echo $row['address']; ?></td>
              <td><?php echo (new DateTime($row['timing']))->format('Y-m-d H:i:s'); ?></td>
              <td><?php echo $row['price']; ?></td>
              <td><a href="cancel.php?bid=<?php echo $row['b_id']; ?>" > <input type="submit" name="cancel"
value="cancel" style="background:rgb(31,37,51)"></td>
            </tr>
          <?php
            }
          ?>
        </tbody>
      </table>

    </div>

  </body>

</html>

```

## PDF GENERATION CODE SNIPPET

```
require("fpdf/fpdf.php");

$pdf = new FPDF();
$pdf->AddPage();

$pdf -> SetFont("Arial","",12);
$pdf -> Cell(0,10,"INVOICE-BILL",1,1,'C');

$pdf -> Cell(20,10,"Time : ".date("h:i:s A"),0,0);
$pdf -> Cell(158,10,"#TICKET-ID : ".$book_id,0,1,'R');
$pdf -> Cell(20,5,"Date : ".date("d/m/y"),0,1);

$pdf -> Cell(30,10,"Show Name",1,0);
$pdf -> Cell(30,10,"Rating",1,0);
$pdf -> Cell(30,10,"No of Seats",1,0);
$pdf -> Cell(30,10,"Price/seats",1,0);
-> Cell(60,10,"Sub Total",1,1);

$pdf -> Cell(30,10,"$Show",1,0);
$pdf -> Cell(30,10,"$rating",1,0);
// $pdf -> Cell(25,10,"$startdate",1,0);
// $pdf -> Cell(25,10,"$enddate",1,0);
$pdf -> Cell(30,10,"$seats",1,0);
$pdf -> Cell(30,10,"$price",1,0);
// $pdf -> Cell(30,10,"$subtotal",1,0);
$pdf -> Cell(60,10,"$subtotal",1,1);

$pdf -> Cell(182,15,"Sub Total : Rs.".$subtotal,0,1,'R');
$pdf -> Cell(179.5,1,"Tax : Rs.".$tax,0,1,'R');
$pdf -> Cell(181.8,15,"Gross Total : Rs.".$total,0,1,'R');
$pdf -> Cell(179.5,1,"Discount : Rs.".(($total-$total_after),0,1,'R');
$pdf -> Cell(181.5,15,"Net Total : Rs.".(($total_after),0,1,'R');

$pdf->output();



?>
```

## 4.2 Table descriptions


Table name:

categories

Billing:

#	Name	Type	Collation	Attributes	Null	Default
1	<b>bill_id</b> 	int(4)			No	None
2	<b>book_id</b> 	int(4)			No	None
3	<b>amount</b>	varchar(20)	utf8mb4_general_ci		No	None
4	<b>bill_date</b>	date			No	current_timestamp()
5	<b>owner_id</b>	int(4)			No	None
6	<b>cust_id</b>	int(4)			No	None

Event\_Loc:

#	Name	Type	Collation	Attributes	Null	Default
1	<b>t_id</b> 	int(4)			No	None
2	<b>owner_id</b>	int(4)			No	None
3	<b>city</b>	varchar(20)	utf8mb4_general_ci		No	None
4	<b>type</b>	varchar(20)	utf8mb4_general_ci		No	None
5	<b>event_name</b>	varchar(20)	utf8mb4_general_ci		No	None
6	<b>address</b>	varchar(40)	utf8mb4_general_ci		No	None
7	<b>DOP</b>	date			No	current_timestamp()
8	<b>no_of_seats</b>	int(4)			No	None

## Discount:

#	Name	Type	Collation	Attributes	Null
1	<b>discount_id</b> 🔑	int(3)			No
2	<b>discount_name</b>	varchar(20)	utf8mb4_general_ci		No
3	<b>expiry_date</b>	date			No
4	<b>discount_percent</b>	int(2)			No

## Customer:


#	Name	Type	Collation	Attributes	Null	Default
1	<b>c_id</b> 🔑	int(4)			No	None
2	<b>customer_name</b>	varchar(20)	utf8mb4_general_ci		No	None
3	<b>address</b>	varchar(20)	utf8mb4_general_ci		No	None
4	<b>email</b>	varchar(20)	utf8mb4_general_ci		No	None
5	<b>phone</b>	varchar(20)	utf8mb4_general_ci		No	None
6	<b>password</b>	varchar(20)	utf8mb4_general_ci		No	None
7	<b>DOJ</b>	date			No	current_timestamp()
8	<b>age</b>	int(2)			Yes	NULL

## Show:

#	Name	Type	Collation	Attributes	Null	Default
1	<b>s_id</b>	int(4)			No	None
2	<b>owner_id</b>	int(4)			No	None
3	<b>city</b>	varchar(20)	utf8mb4_general_ci		No	None
4	<b>type</b>	varchar(20)	utf8mb4_general_ci		No	None
5	<b>show_name</b>	varchar(20)	utf8mb4_general_ci		No	None
6	<b>DOP</b>	date			No	None
7	<b>price</b>	int(4)			No	None
8	<b>rating</b>	varchar(20)	utf8mb4_general_ci		No	None



**Owner:**

#	Name	Type	Collation	Attributes	Null	Default
1	<b>own_id</b> 	int(4)			No	None
2	<b>owner_name</b>	varchar(20)	utf8mb4_general_ci		No	None
3	<b>email</b>	varchar(50)	utf8mb4_general_ci		No	None
4	<b>address</b>	varchar(50)	utf8mb4_general_ci		No	None
5	<b>phone</b>	varchar(20)	utf8mb4_general_ci		No	None
6	<b>password</b>	varchar(20)	utf8mb4_general_ci		No	None
7	<b>date</b>	date			No	current_timestamp()

**4.3 Table Creation****Billing:**

```
CREATE TABLE `billing` (
  `bill_id` int(4) NOT NULL,
  `book_id` int(4) NOT NULL,
  `amount` varchar(20) NOT NULL,
  `bill_date` date NOT NULL DEFAULT current_timestamp(),
  `owner_id` int(4) NOT NULL,
  `cust_id` int(4) NOT NULL
)
```

**BOOK:**

```
CREATE TABLE `book` (
  `b_id` int(4) NOT NULL,
  `customer_name` varchar(20) NOT NULL,
  `c_id` int(4) NOT NULL,
  `t_id` int(4) NOT NULL,
  `m_id` int(4) NOT NULL,
  `timing` date NOT NULL
)
```

**COMMISSION:**

```
CREATE TABLE `commission` (
  `commision_id` int(4) NOT NULL,
  `commission_amount` varchar(20) NOT NULL,
  `bill_id` int(4) NOT NULL,
  `pay_to_owner` varchar(20) NOT NULL,
  `date_of_commission` date NOT NULL DEFAULT current_timestamp()
)
```

**DISCOUNT:**

```
CREATE TABLE `customer_discount` (
  `dis` int(4) NOT NULL,
  `discount_name` varchar(20) NOT NULL,
  `customer_id` int(4) NOT NULL
)
```

**CUSTOMER SIGNUP:**

```
CREATE TABLE `customer_signup` (
  `c_id` int(4) NOT NULL,
  `customer_name` varchar(20) NOT NULL,
  `address` varchar(20) NOT NULL,
  `email` varchar(20) NOT NULL,
  `phone` varchar(20) NOT NULL,
  `password` varchar(20) NOT NULL,
  `DOJ` date NOT NULL DEFAULT current_timestamp(),
  `age` int(2) DEFAULT NULL
)
```

**DISCOUNT LIST:**

```
CREATE TABLE `discount_list` (
  `discount_id` int(3) NOT NULL,
  `discount_name` varchar(20) NOT NULL,
  `expiry_date` date NOT NULL DEFAULT current_timestamp(),
  `discount_percent` int(2) NOT NULL
)
```

**SHOW:**

```
CREATE TABLE `SHOW` (
  `s_id` int(4) NOT NULL,
  `owner_id` int(4) NOT NULL,
  `city` varchar(20) NOT NULL,
  `show_name` varchar(20) NOT NULL,
  `DOP` date NOT NULL,
  `price` int(4) NOT NULL,
  `featured` varchar(20) NOT NULL
)
```

**OWNER SIGNUP:**

```
CREATE TABLE `owner_signup` (  
  `own_id` int(4) NOT NULL,  
  `owner_name` varchar(20) NOT NULL,  
  `email` varchar(50) NOT NULL,  
  `address` varchar(50) NOT NULL,  
  `phone` varchar(20) NOT NULL,  
  `password` varchar(20) NOT NULL,  
  `date` date NOT NULL DEFAULT current_timestamp()  
)
```

**EVENT\_LOC:**

```
CREATE TABLE `event_loc` (  
  `t_id` int(4) NOT NULL,  
  `owner_id` int(4) NOT NULL,  
  `city` varchar(20) NOT NULL,  
  `type` varchar(20) NOT NULL,  
  `event_name` varchar(20) NOT NULL,  
  `address` varchar(40) NOT NULL,  
  `DOP` date NOT NULL DEFAULT current_timestamp(),  
  `no_of_seats` int(4) NOT NULL  
)
```

#### 4.4 Insertion of tuples

```
INSERT INTO `billing` (`bill_id`, `book_id`, `amount`, `bill_date`, `owner_id`, `cust_id`)
VALUES
```

```
(70, 113, '3680', '2023-01-14', 4, 13),
(71, 114, '3050', '2023-01-14', 4, 13),
(77, 120, '236.25', '2023-01-15', 4, 19),
(90, 133, '535.5', '2023-01-16', 4, 13),
(91, 133, '535.5', '2023-01-16', 4, 13),
(92, 133, '535.5', '2023-01-16', 4, 13),
(98, 139, '18800', '2023-01-19', 14, 19);
```

```
INSERT INTO `book` (`b_id`, `customer_name`, `c_id`, `t_id`, `s_id`, `timing`) VALUES
```

```
(113, 'raiyan', 13, 15, 5, '2023-01-14'),
(114, 'harsh', 13, 15, 5, '2023-01-14'),
(120, 'karuna', 19, 15, 4, '2023-01-11'),
(133, 'suthar', 13, 15, 4, '2023-01-19'),
(139, 'shivram', 19, 29, 16, '2023-01-20');
```

```
INSERT INTO `customer_discount` (`dis`, `discount_name`, `customer_id`) VALUES
```

```
(27, 'WELCOME15', 13),
(28, 'WELCOME15', 19),
(29, 'WELCOME15', 17),
(30, 'WELCOME15', 19),
(31, 'WELCOME15', 20),
(32, 'WELCOME15', 21),
(33, 'WELCOME15', 19);
```

```
INSERT INTO `customer_signup` (`c_id`, `customer_name`, `address`, `email`, `phone`,
`password`, `DOJ`, `age`) VALUES
```

```
(13, 'raiyan', 'Hurhuru, Hazaribagh,', 'raiyan0@gmail.c', '7481 016483', '123', '2022-12-19', 19),
(14, 'harsh', 'bangalore', 'harsh@gmail.com', '1234567890', '111', '2022-12-23', 18),
(17, 'karuna', 'Hurhuru, Hazaribagh,', 'karuna@gmail.com', '07481 016483', '123', '2023-01-02', 21),
(18, 'shubham', 'Hurhuru, Hazaribagh,', 'shubham@gmail.com', '07481 016483', '123', '2023-01-05', 22),
(19, 'tarun', 'stay grand', 'tarun@gmail.com', '7687576465', '123', '2023-01-14', 21),
(20, 'shiv', 'stay grand', 'shiv12@gmail.com', '9090909090', '1233', '2023-01-16', 20),
(21, 'vimki', 'stay grand', 'vimki67@gmail.com', '8989898989', '123', '2023-01-16', 15);
```

```
INSERT INTO `discount_list` (`discount_id`, `discount_name`, `expiry_date`,  
`discount_percent`) VALUES  
(4, 'WELCOME15', '2023-01-16', 15);
```

```
INSERT INTO `show` (`s_id`, `owner_id`, `city`, `type`, `show_name`, `DOP`, `price`,  
`rating`) VALUES  
(19, 15, 'Bangalore', 'the laugh club', 'bas kar bassi', '2023-01-28', 499, 'anubhav'),  
(22, 15, 'Bangalore', 'the comedy club', 'jo bolta h wahi hota', '2023-01-28', 299, 'harsh  
gujral'),  
(23, 15, 'Mumbai', 'the comedy club', 'hostel', '2023-01-28', 299, 'upmanyu'),  
(24, 15, 'Bangalore', 'the laughter', 'election and voting', '2023-01-29', 599, 'varun grover'),  
(25, 15, 'Bangalore', 'raiyan', 'abcd', '2023-01-29', 400, 'raiyan');
```

```
INSERT INTO `owner_signup` (`own_id`, `owner_name`, `email`, `address`, `phone`,  
`password`, `date`) VALUES  
(4, 'raiyan', 'raiyan@gmail.com', 'bangalore', '9876543210', '123', '2022-12-19'),  
(6, 'harsh', 'harsh0@gmail.c', 'Hurhuru, Hazaribagh,', '07481 016483', '123', '2023-01-02'),  
(7, 'karuna', 'karuna@gmail.c', 'Hurhuru, Hazaribagh,', '07481 016483', '12345', '2023-01-  
05'),  
(8, 'tarun', 'tarun@gmail.com', 'rajasthan', '9982031303', 'pt', '2023-01-05'),  
(9, 'tarun', '1jssjpdj@gmail.com', 'stay grand', '3368768763', '123', '2023-01-15'),  
(10, 'smruti', 'smruti@gmail.com', 'kjjshfjkehnf', '8687585478', '123', '2023-01-15'),  
(11, 'raiyan arsh', 'tarunsmalglmhh8@gmail.com', 'B-113 sidhi vinayak colony',  
'09982031303', '123', '2023-01-15'),  
(12, 'shiv', 'shiv45@gmail.com', 'stay grand', '7878787878', '1233', '2023-01-16'),  
(13, 'vimki', 'vimki23@gmail.com', 'stay grand', '1234567890', '123', '2023-01-16'),  
(14, 'suresh', 'suresh@gmail.com', 'stay grand', '1234567896', '123', '2023-01-19');
```

```
INSERT INTO `event_loc` (`t_id`, `owner_id`, `city`, `type`, `event_name`, `address`,  
`DOP`, `no_of_seats`) VALUES  
(32, 15, 'Bangalore', '', 'the laugh club', 'stay grand', '2023-01-28', 18),  
(33, 15, 'Bangalore', '', 'the comedy club', 'uttarahali', '2023-01-28', 300),  
(34, 15, 'Bangalore', '', 'the laughter', 'whitefield', '2023-01-28', 500),  
(35, 15, 'Bangalore', '', 'raiyan', 'hazaribagh', '2023-01-28', 600)  
(32, 15, 'Bangalore', '', 'the laugh club', 'stay grand', '2023-01-28', 18),  
(33, 15, 'Bangalore', '', 'the comedy club', 'uttarahali', '2023-01-28', 300),  
(34, 15, 'Bangalore', '', 'the laughter', 'whitefield', '2023-01-28', 500),  
(35, 15, 'Bangalore', '', 'raiyan', 'hazaribagh', '2023-01-28', 600);
```

## 5. Queries

The below mentioned queries are all embedded in the PHP scripting language in our database:

```
$sql_query = "SELECT * FROM `show` WHERE `owner_id` = '$id'";  
$result = mysqli_query($conn,$sql_query);  
mysqli_close($conn);
```

```
$sql_query = "SELECT * FROM `event` WHERE `owner_id` = '$id'";  
$result = mysqli_query($conn,$sql_query);  
mysqli_close($conn);
```

```
$sql_query = "SELECT `show`.`city` ,`show`.`type`  
,`show`.`m_id`,`show`.`show_name` ,`show`.`DOP` ,`show`.`price` ,`show`.`rating`  
,`event`.`event_name`,`event`.`t_id`,`event`.`no_of_seats` FROM `show`,`event`  
WHERE `show`.`city` = '$city' and `show`.`type` = '$type' and `event`.`city` = '$city'  
and `event`.`type` = '$type' and `event`.`no_of_seats` >= '$seats'";  
$result = mysqli_query($conn,$sql_query);  
mysqli_close($conn);
```

```
$sql_query5 = "UPDATE `event` SET `no_of_seats` = `no_of_seats`-$seats WHERE  
`event`.`t_id` = $tid;";  
$result5 = mysqli_query($conn,$sql_query5);
```

```
$sql_query = "SELECT `b_id` FROM `book` WHERE `c_id` = '$c_id' AND `m_id` =  
'$mid' ";  
$result = mysqli_query($conn,$sql_query);  
while ($row = $result->fetch_assoc()) {  
    $book_id = $row['b_id'];  
}
```

```

$sql_query1 = "SELECT `owner_id` FROM `show` WHERE `m_id` = '$mid' ";
$result1 = mysqli_query($conn,$sql_query1);
while ($row = $result1->fetch_assoc()) {
    $owner_id = $row['owner_id'];
}

```

*// adding to billing table*

```

$sql_query2 = "INSERT INTO `billing`
(`book_id`,`amount`,`bill_date`,`owner_id`,`cust_id`) VALUES
('$book_id','$total_after',current_timestamp(),'$owner_id','$c_id');"
$result2 = mysqli_query($conn,$sql_query2);

```

*// adding to customer\_discount table*

```

if($flag == 1)
{
    $sql_query4 = "SELECT `discount_name` FROM `discount_list`;";
    $result4 = mysqli_query($conn,$sql_query4);
    while ($row = $result4->fetch_assoc()) {
        if(($row['discount_name'])==$coupon)
        {
            $sql_query3 = "INSERT INTO `customer_discount`
            (`discount_name`,`customer_id`) VALUES ('$coupon','$c_id');"
            $result3 = mysqli_query($conn,$sql_query3);
        }
    }
}

```

```

$sql_query1 = "INSERT INTO `book` (`customer_name`, `c_id`, `m_id`,
`t_id`,`timing`) VALUES ('$cust_name', '$c_id', '$mid', '$tid','$timing');"
$result1 = mysqli_query($conn,$sql_query1);

```

```

$sql_query1 = "DELETE FROM `book` WHERE `book`.`b_id` = '$bid'";
$result1 = mysqli_query($conn,$sql_query1);

```

```

$sql_query = "SELECT
`book`.`timing`,`book`.`b_id`,`show`.`city`,`show`.`type`,`show`.`show_name`,`show`.`
rating`,`show`.`price`,`event`.`event_name`,`event`.`address` FROM
`book`,`show`,`event` WHERE `book`.`c_id` = '$id' and `book`.`t_id`=`event`.`t_id`
and `book`.`m_id`=`show`.`m_id`;";
$result = mysqli_query($conn,$sql_query);
mysqli_close($conn);

```

## 4.6 TRIGGERS

```
CREATE TRIGGER `commission_to_admin` AFTER INSERT ON `billing` FOR  
EACH ROW INSERT INTO  
commission(commission_amount,pay_to_owner,bill_id,date_of_commission)  
VALUES((new.amount*0.25),(new.amount*0.75),new.bill_id,now())  
$$  
DELIMITER
```

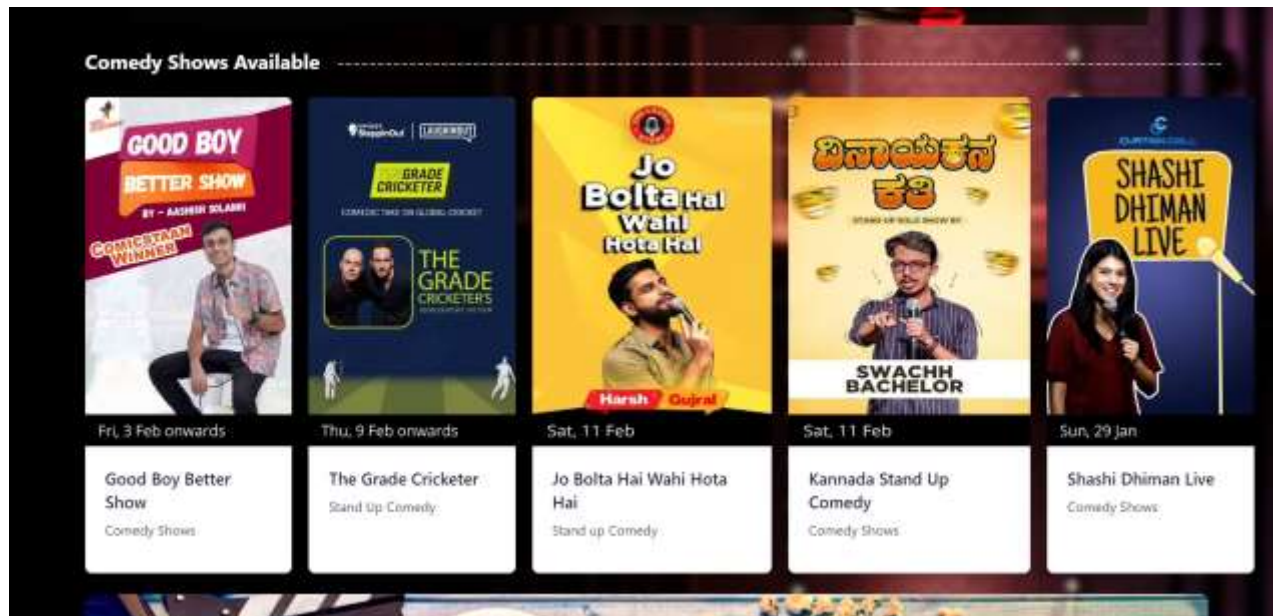
The commission trigger in the show ticket booking system is a mechanism that automatically calculates and deducts a 25% commission from the total revenue generated from ticket sales. This commission is typically taken from the ticket vendor or theater operator and is used to cover the costs of the booking system and any other associated expenses. The commission trigger is activated every time a ticket is purchased through the system and is automatically calculated based on the total revenue generated from the sale. Once the commission has been deducted, the remaining amount is then credited to the account of the vendor or theater operator. This process ensures that the booking system is able to recoup its expenses while also providing a fair and transparent system for vendors and theater operators to sell their tickets



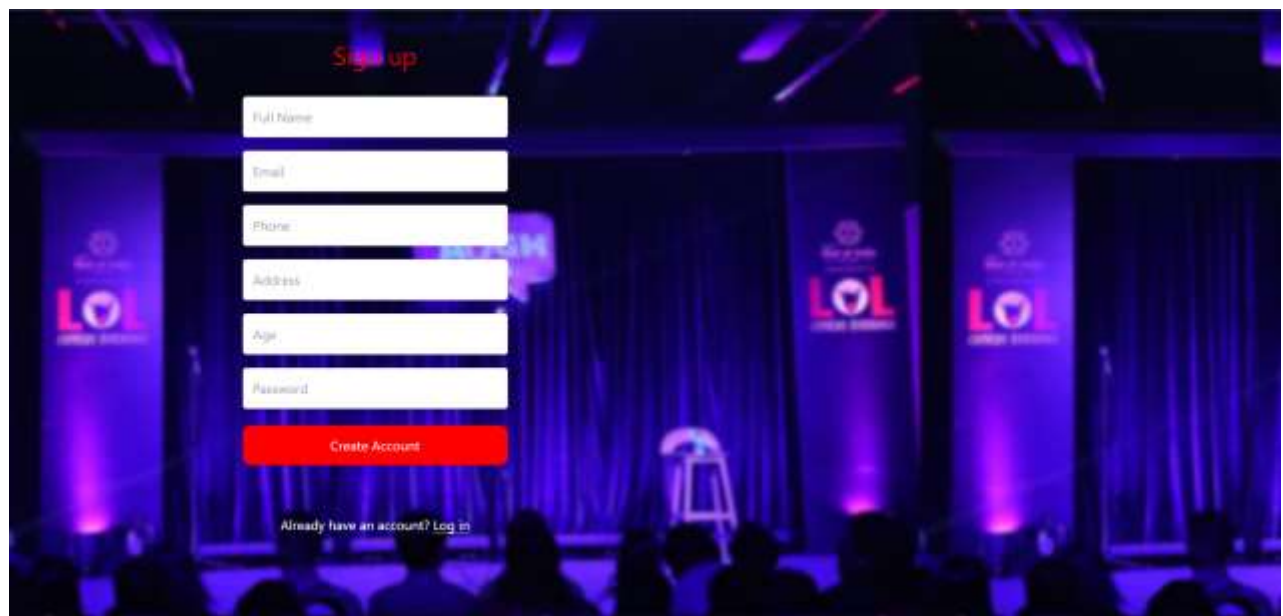
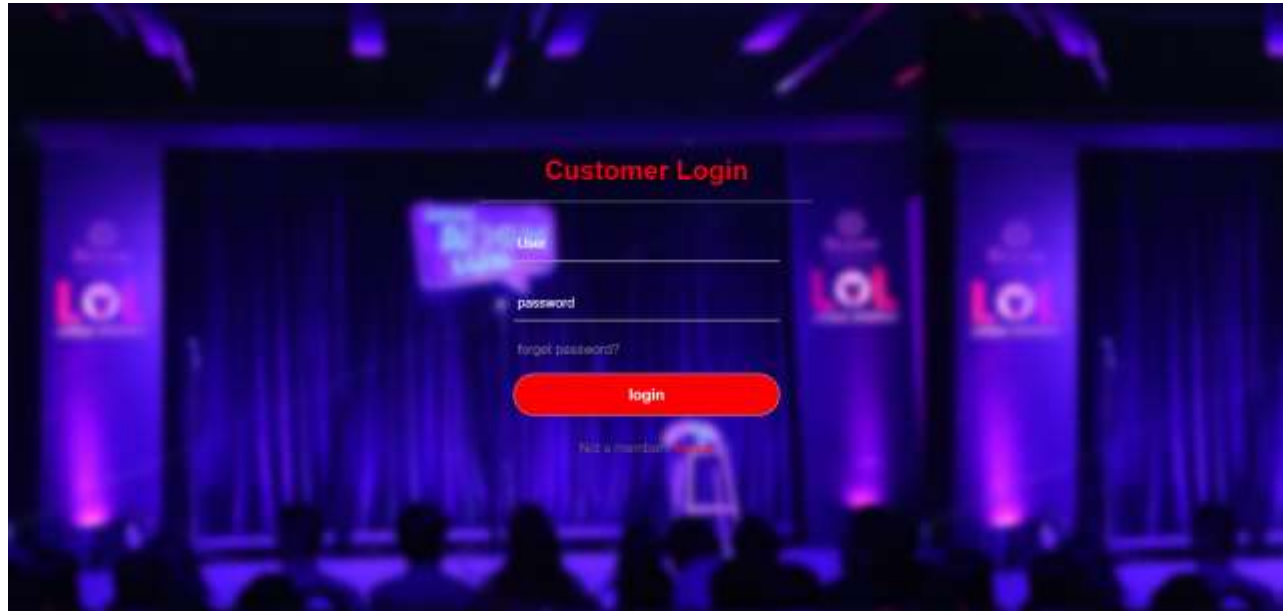
## CHAPTER 5: RESULTS

### 1. Front End

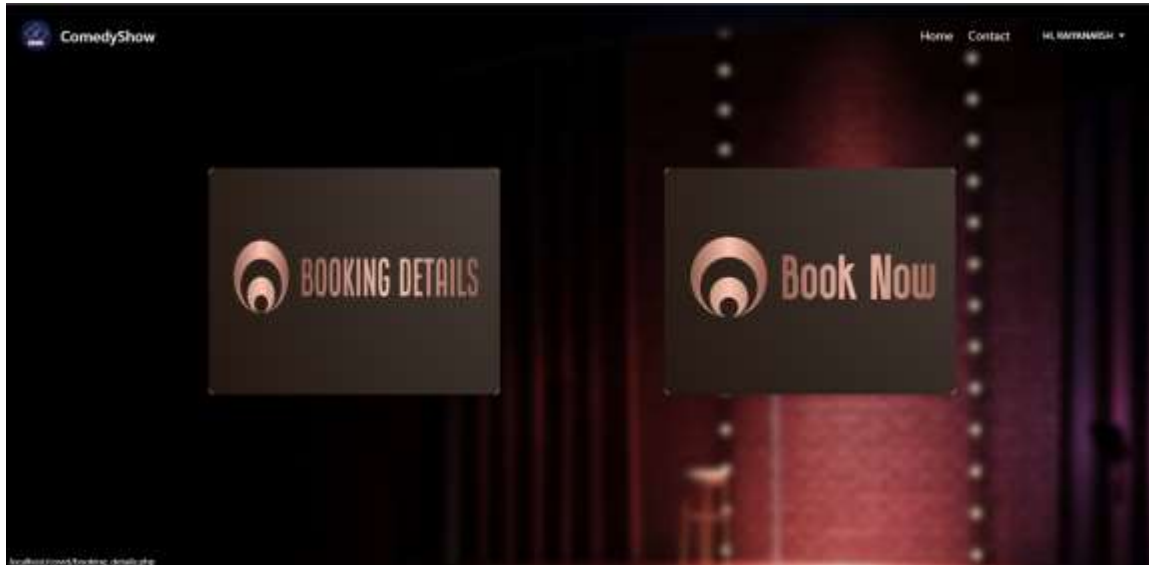
#### 1.HOMEPAGE



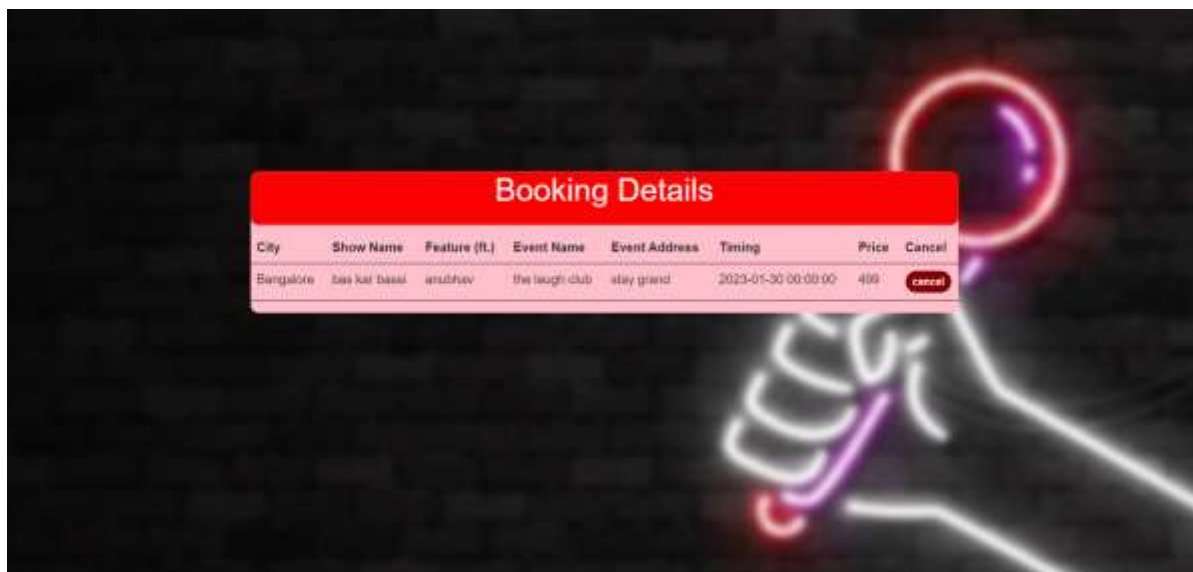
## 2.Customer Login & Signup



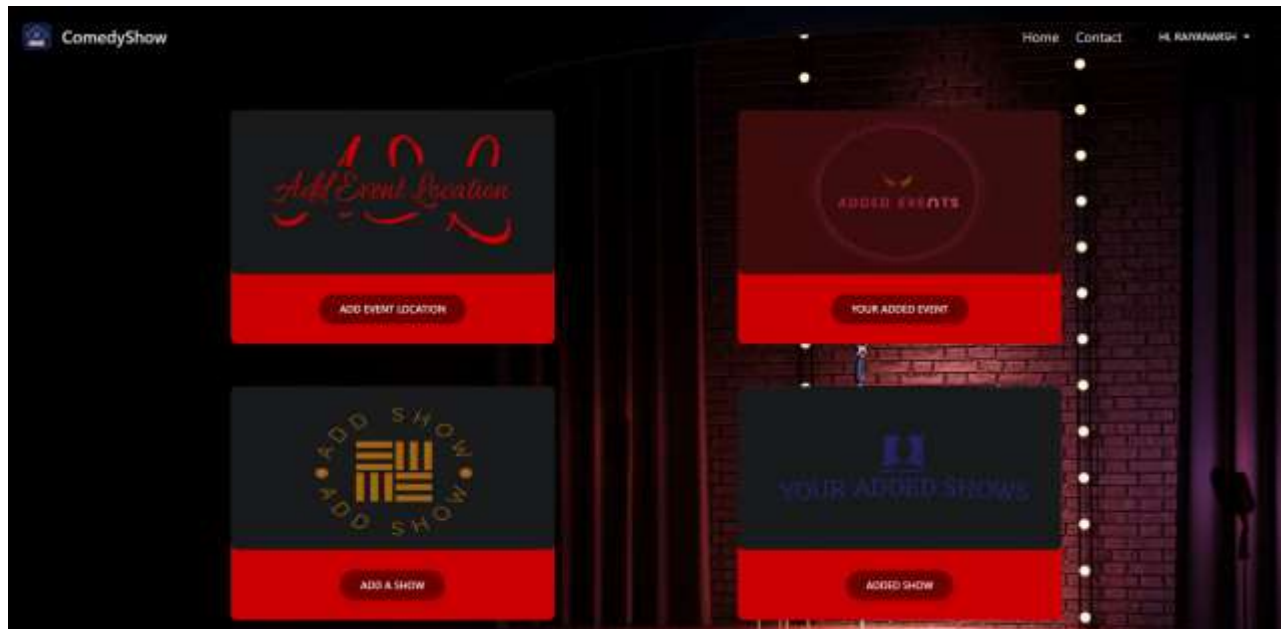
### 3.Customer Dashboard



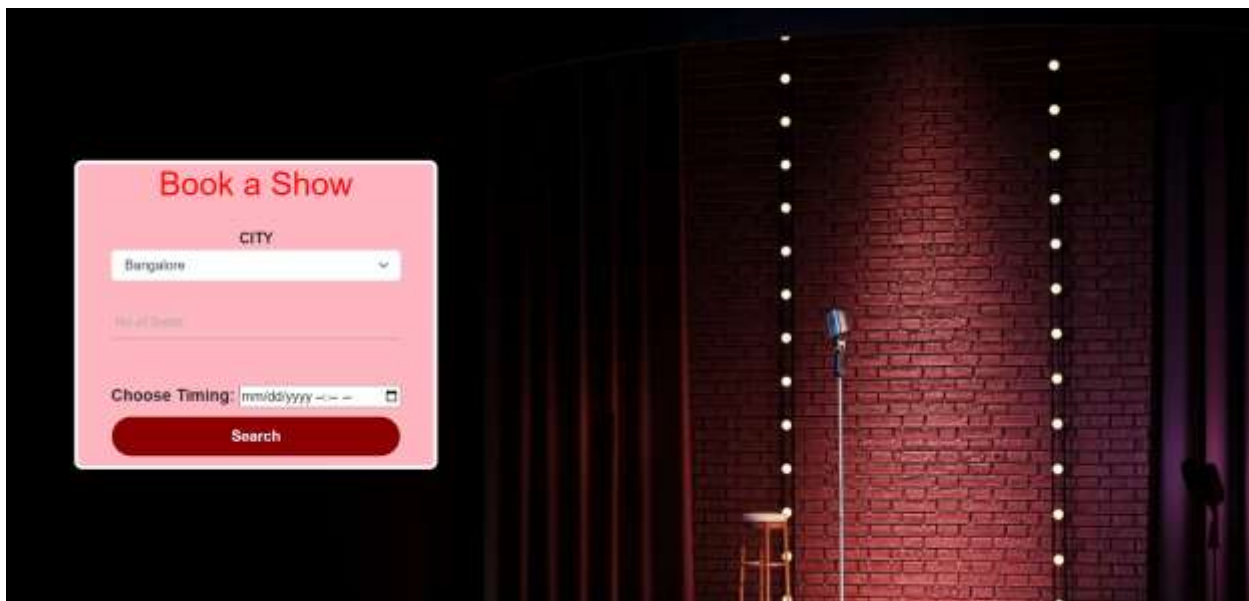
### 4.Booking Details



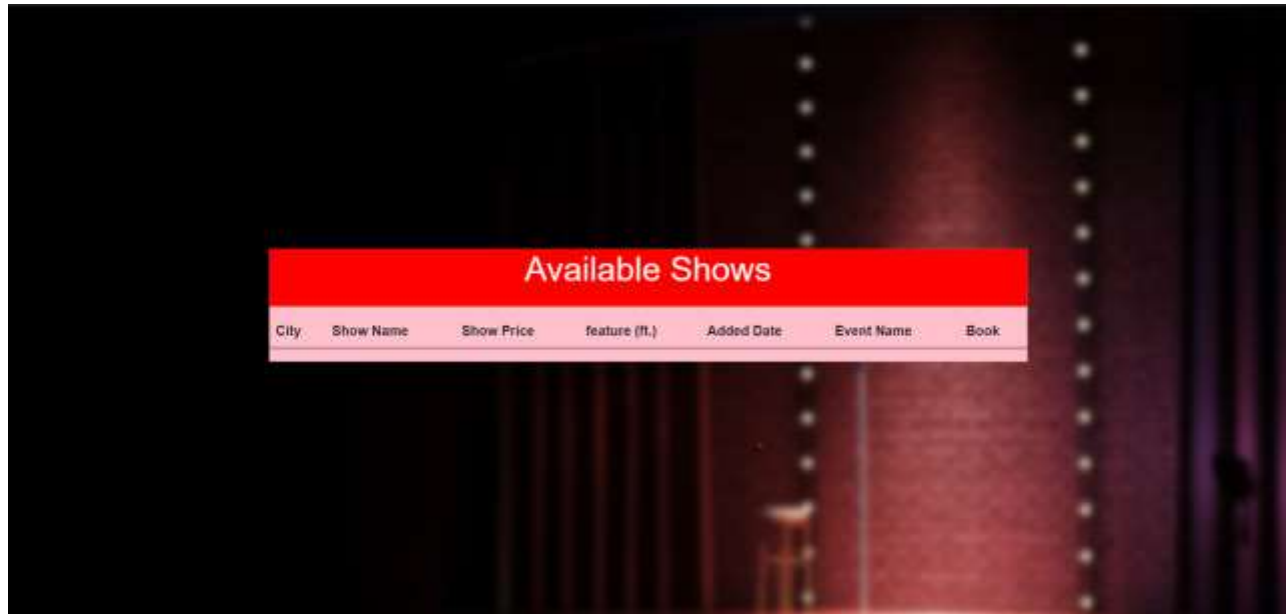
## 5. OWNER DASHBOARD



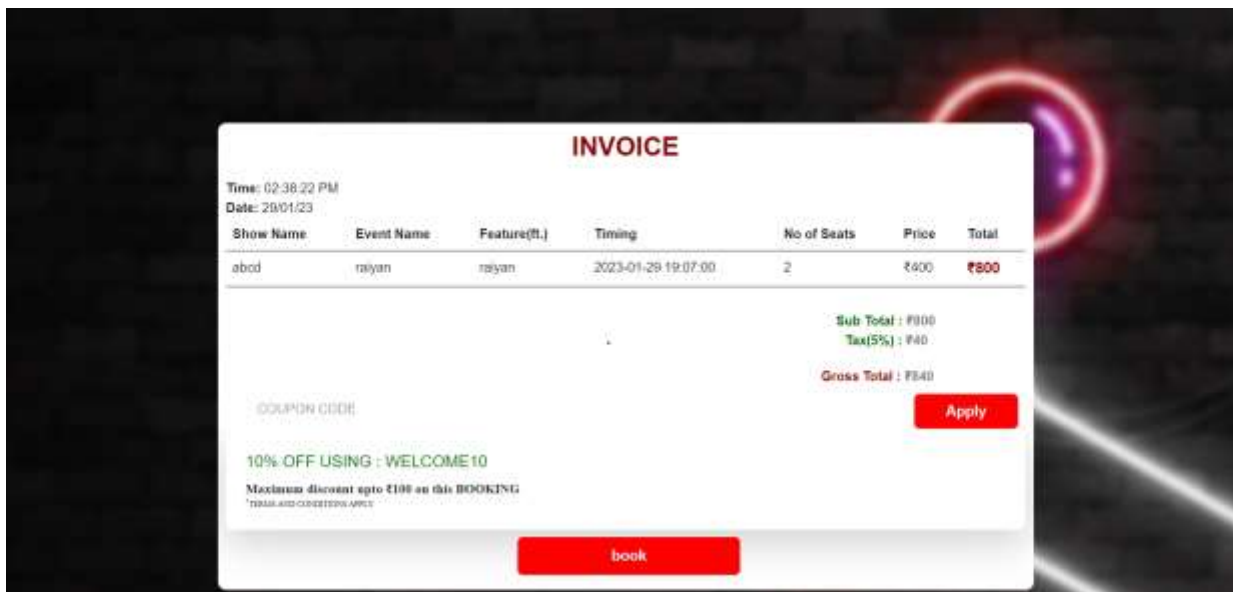
## 6. Show Selection



## 6. Available Shows



## 7. Billing



## 8. Pdf bill

INVOICE-BILL				
Time : 02:40:04 PM			#TICKET-ID : 149	
Date : 29/01/23				
Show Name	Feature(ft.)	No of Seats	Price/seats	Sub Total
abcd	raayan	2	400	800
Sub Total : Rs.800				
Tax : Rs.40				
Gross Total : Rs.840				
Discount : Rs.0				
Net Total : Rs.840				



## **CHAPTER 6: CONCLUSION**

### **CONCLUSION**

A Comedy show ticket booking system is a software application that allows users to purchase tickets for Comedy shows online

The system typically includes a user interface for browsing and selecting Comedy shows, showtimes, and seats.

Users can also add on additional services such as concessions or special seating.

Payment can be made through a variety of methods such as credit card or mobile wallet

Once the transaction is complete, the user will receive a confirmation of their purchase and can either print out or digitally access their tickets

The system also allows for the management of inventory and revenue tracking for the Comedy show event.

### **Limitations**

- Limited performance: Both PHP and MySQL can be less performant than other languages and databases when handling large amounts of data or performing complex calculations, particularly when dealing with high concurrency.
- Limited scalability: While MySQL and PHP can handle large amounts of data, they may not scale as well as other languages and databases in terms of handling a large number of concurrent users or requests.
- Security vulnerabilities: As with any system, if the PHP code and MySQL database are not properly secured, there may be security vulnerabilities that can be exploited by attackers.
- Limited support for modern features: Both PHP and MySQL are older technologies and may not have built-in support for newer technologies and features, such as machine learning or web assembly.

- Limited integration with other systems: If the system is not able to integrate with other systems, such as inventory management or accounting software, it may be difficult to effectively manage all aspects of the vehicle service process.

## **Future Enhancements**

- Mobile App integration: A mobile app that allows users to book tickets and select seats from their smartphones.
- Virtual Reality (VR) integration: Users can experience a virtual tour of the event before booking their tickets.
- Social Media Integration: Users can share their plans with friends on social media platforms and invite them to join.
- Personalized recommendations: A recommendation system that suggests Comedy shows based on a user's viewing history and preferences.
- Voice Recognition: Users can book tickets using voice commands through virtual assistants like Alexa or Google Assistant.
- Virtual Queue: A virtual queue system that allows users to book tickets in advance and avoid long lines at the box office.
- Loyalty program: A loyalty program that rewards frequent Comedy show-goers with discounts and perks.
- Integrate with other services: Integrate the Comedy show ticket booking system with food delivery services, ride-hailing services, and other relevant services to make the overall experience more seamless.



## **CHAPTER 7: REFERENCES**

### **BOOK REFERENCES**

- Fundamentals of Database Systems, Ramez Elmasri and Shamkant B. Navathe, 7th Edition, 2017, Pearson.
- Database management systems, Ramakrishnan, and Gehrke, 3rd Edition, 2014, McGraw Hill
- Silberschatz Korth and Sudharshan, Database System Concepts, 6th Edition, McGrawHill, 2013.
- Coronel, Morris, and Rob, Database Principles Fundamentals of Design, Implementation and Management, Cengage Learning 2012.

### **WEB REFERENCES**

<https://stackoverflow.com/>

<https://www.youtube.com/>

<https://www.w3schools.com/php/default.asp>