< Online Event Hall Booking Management System >

**Design Document**

Version 1.0



**Group Id: S2002687F7**

**Supervisor Name: Khaqan Khawer**

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date (dd/mm/yyyy)** | **Version** | **Description** | **Author** |
| 15/06/2020 | 1.0 | Introduction of document  Entity Relationship Diagram  Sequence Diagram  Architecture Diagram  Class Diagram  Database Design  Interface Design  Test Cases | **BC170402435** |
|  |  |  |  |

1. [Introduction of Design Document](#One)
2. [Entity Relationship Diagram (ERD)](#ERD)
3. [Sequence Diagrams](#Six)
4. [Architecture Design Diagram](#Seven)
5. [Class Diagram](#class)
6. [Database Design](#databasedesign)
7. [Interface Design](#interfacedesign)
8. [Test Cases](#testcases)

**Design Document**

1. **Introduction of Design Document**

The requirement document does not specify the architectural or implementation details, but specifies information at the higher level of description. The problem statement, the customer's expectations, and the criteria for success are examples of high-level descriptions.

Sometimes, if an exact engineering detail needs to be specified, this detail will also appear in the requirement document. This is the exception and should not be the rule. These exceptions occur for many reasons including maintaining the consistency with other established systems, availability of particular options, customer's demands, and to establish, at the requirement level, a particular architecture vision.

The **analysis phase** defines the requirements of the system, independent of how these requirements will be accomplished. This phase defines the problem that the customer is trying to solve. The deliverable result at the end of this phase is a requirement document. Ideally, this document states in a clear and precise fashion what is to be built. This analysis represents the ``what'' phase. The requirement document tries to capture the requirements from the customer's perspective by defining goals and interactions at a level removed from the implementation details.

In the **design phase**, the very important decomposition of the problem leads to the development of data structures and algorithms. A functional decomposition for a distributed environment leads to a natural split of the data structures and algorithms. Examples include distributed client-server systems, where a database holds the data in a server while the algorithms manipulating the data reside on the client.

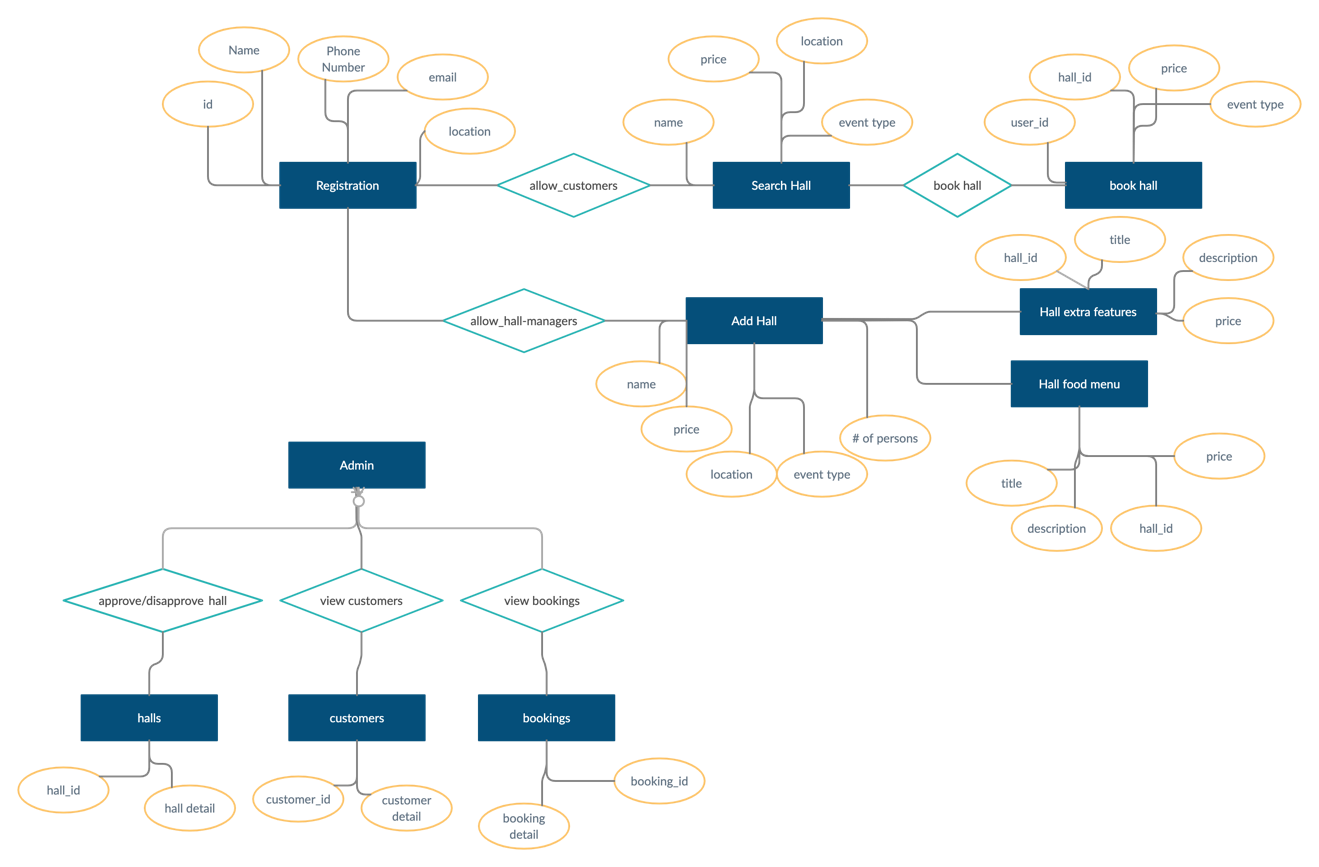
Software development process is a structure imposed on the development of the software. Similar term include software life cycle and software process. It includes many different models for such process each describes variety of task or activities that take place during the process.

The goal of logical design phase is to synchronized the knowledge of application expert and implementation expert to produce a logical model of the system which could be implemented and would work and but perhaps not optimally. This phase insure the systematic development. The process is driven by the product of the analysis phase and end in a model of software program for future development.

This Document include different diagrams like

* Entity Relationship Diagram
* Sequence Diagram
* Architecture Diagram
* Class Diagram
* Database Diagram
* Interface Diagram
* Test Cases

1. **Entity Relationship Diagram (ERD):**



1. Sequence Diagrams:

:Registration

register()

Registered

: User

:Login

login()

Get Login

: User

:Hall Searching

searchHall ()

Get Hall List

: User(customer)

:Add Hall

addHall()

Hall added

: User(hall manager)

: view hall bookings

getHallBookings()

*View Hall booked*

: User(hall manager)

:View Halls

getAllHalls()

View all halls

: User(Admin)

: View Customers

Customers List

: User(Admin)

getCustomers()

: View all bookings

All Bookings List

: User(Admin)

getAllBookings()

:give feedback

placeFeedback()

: User(customer)

Place hall feedback

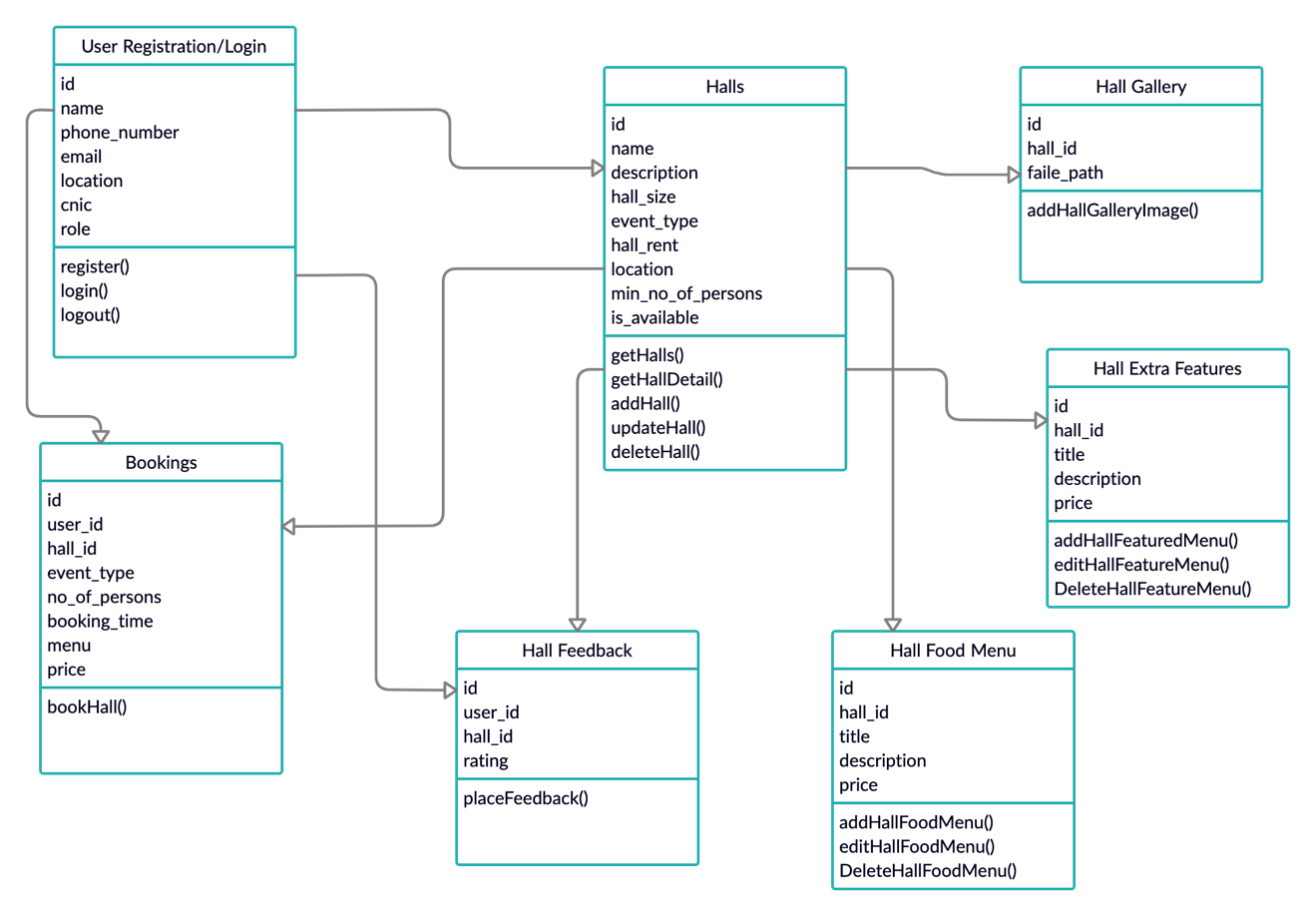
1. **Architecture Design Diagram**

**Application GUI**

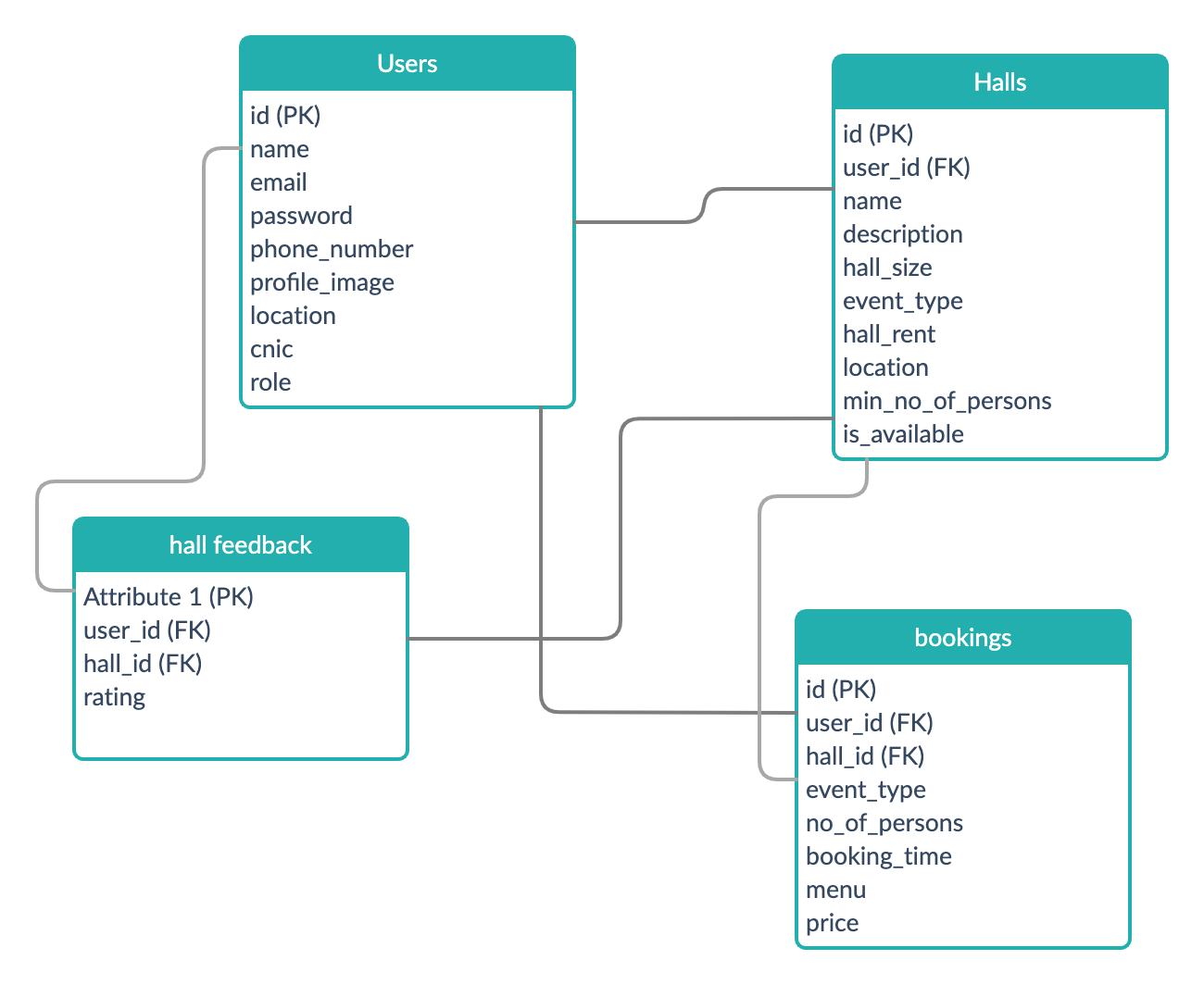
**Hall Management**

**User Management**

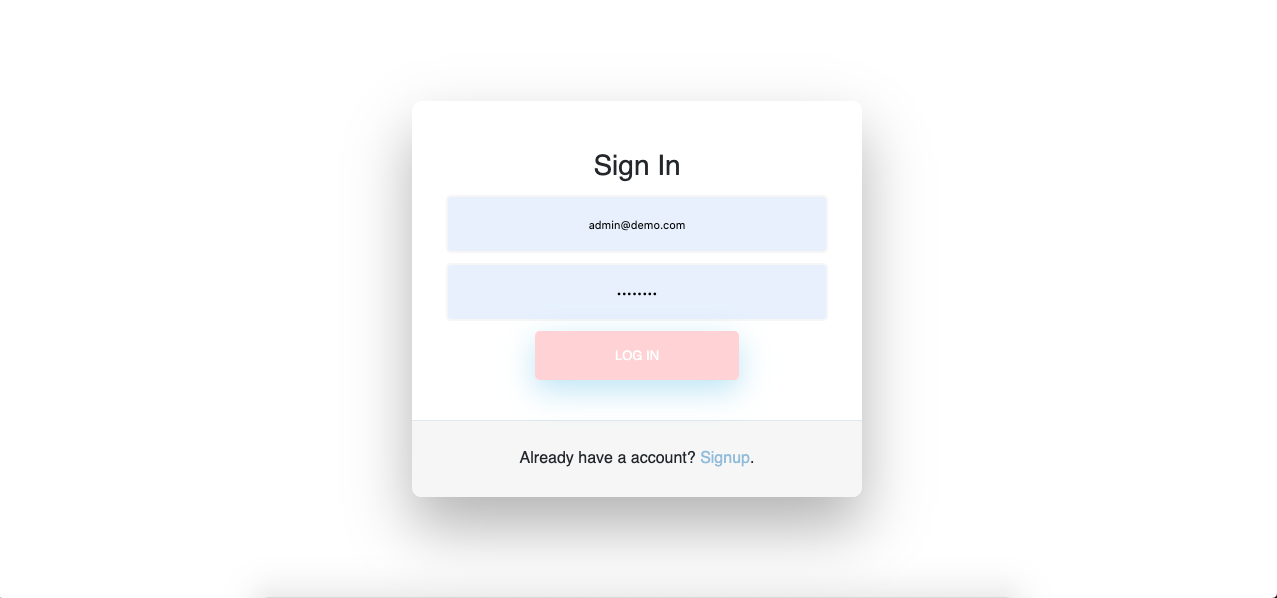
1. **Class Diagram**

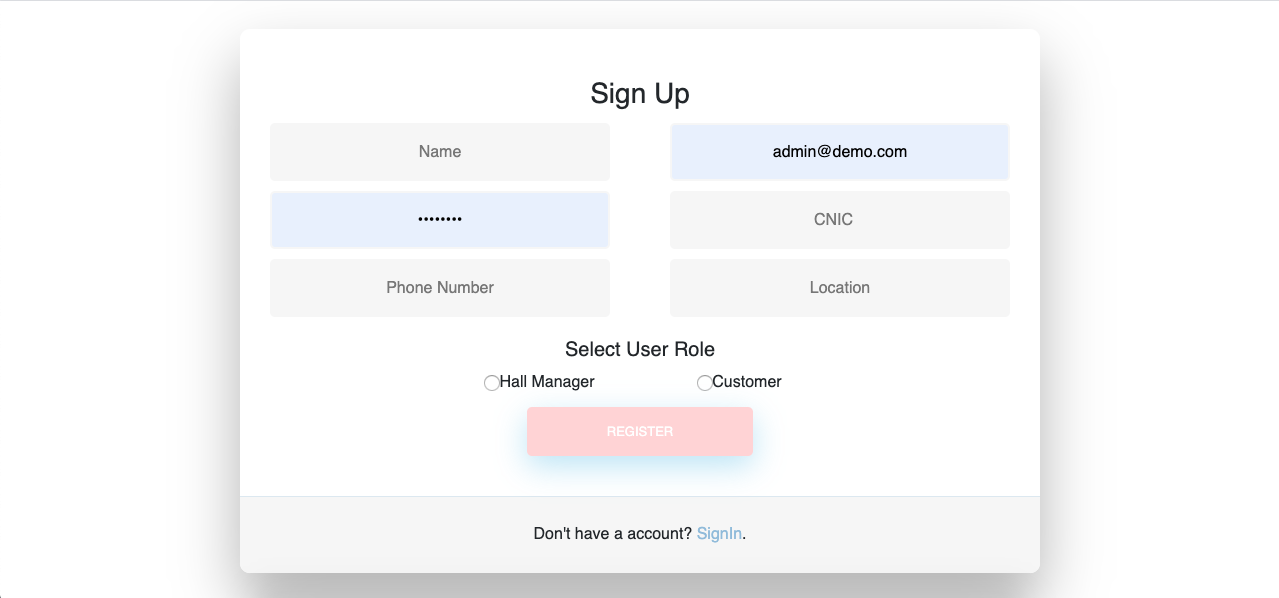


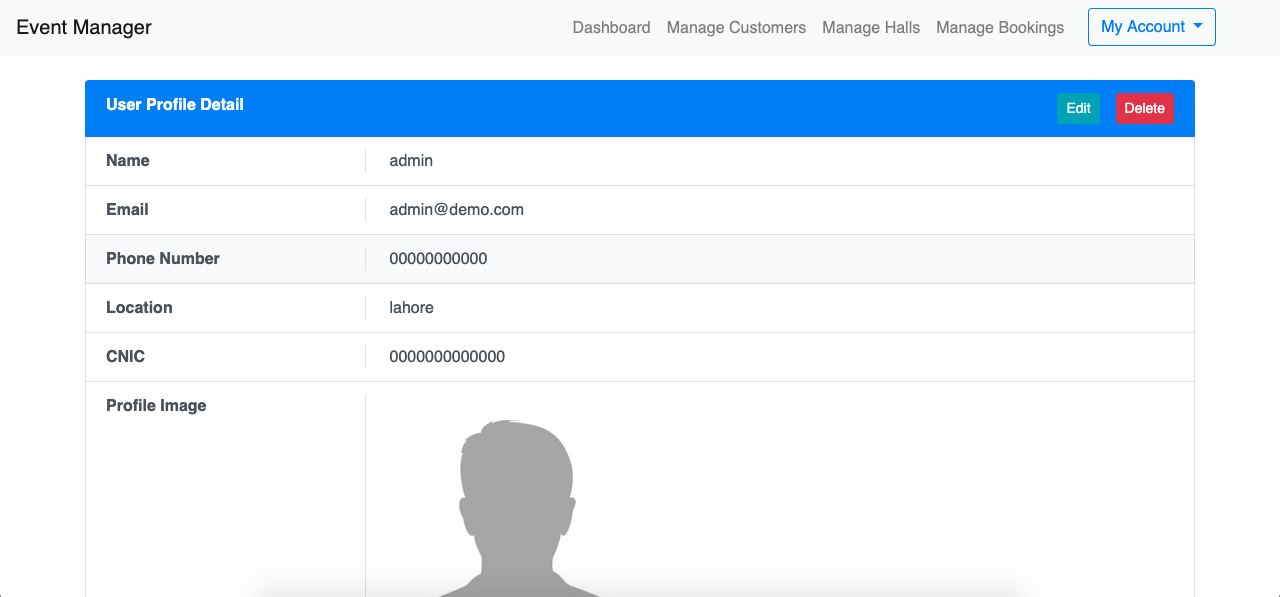
1. **Database Design:**

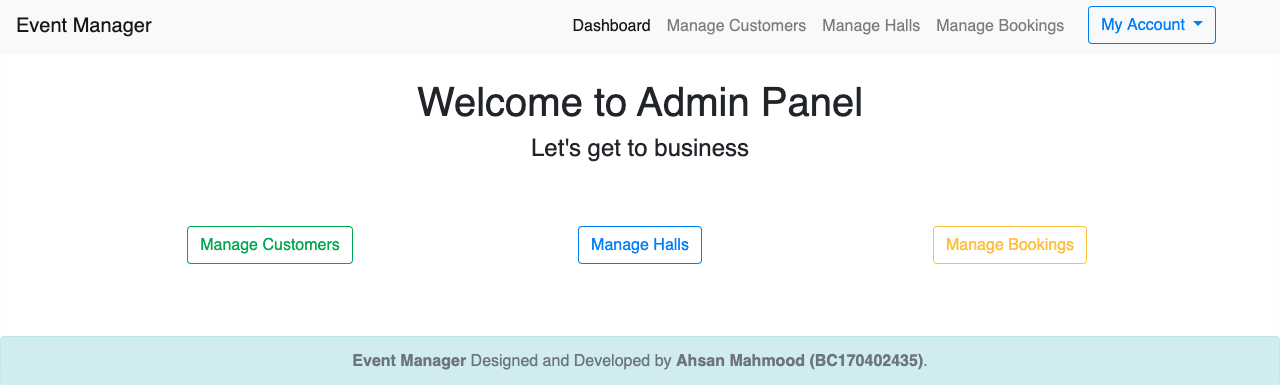


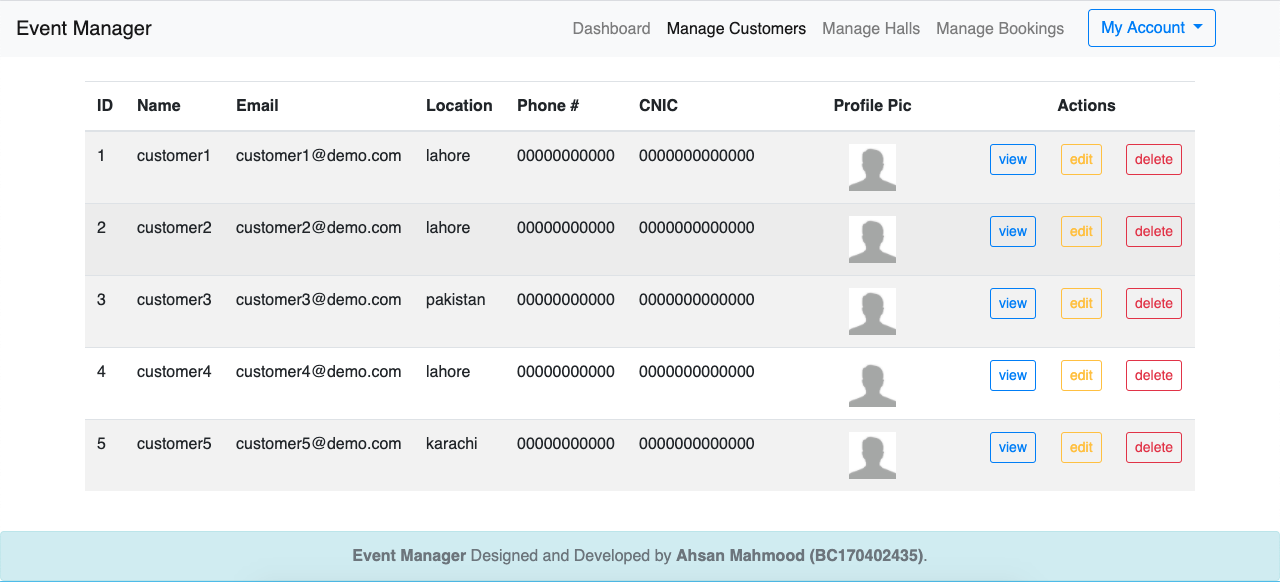
1. Interface Design

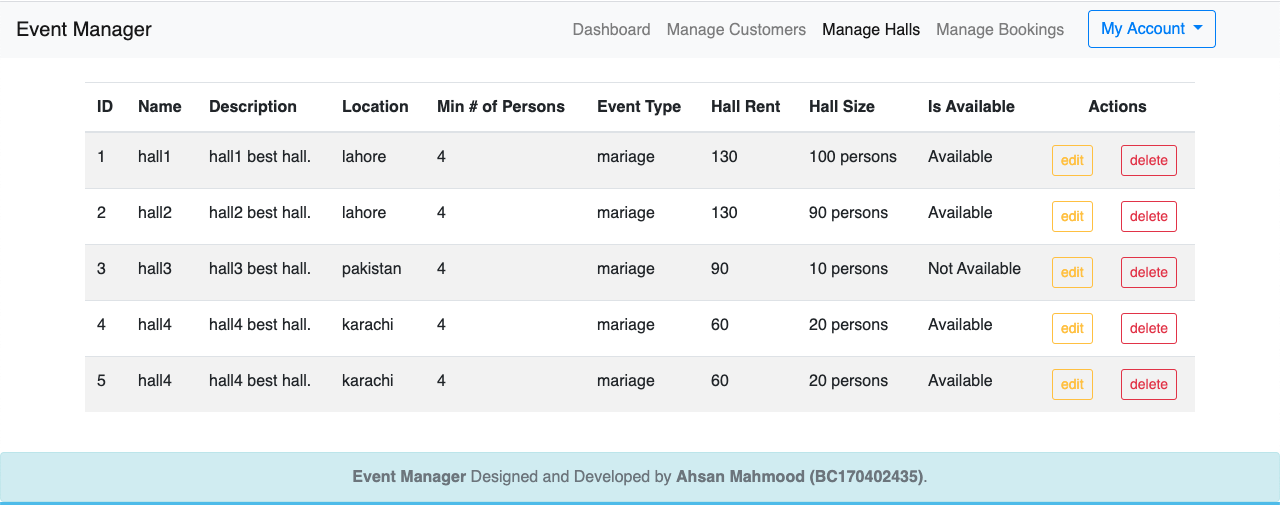


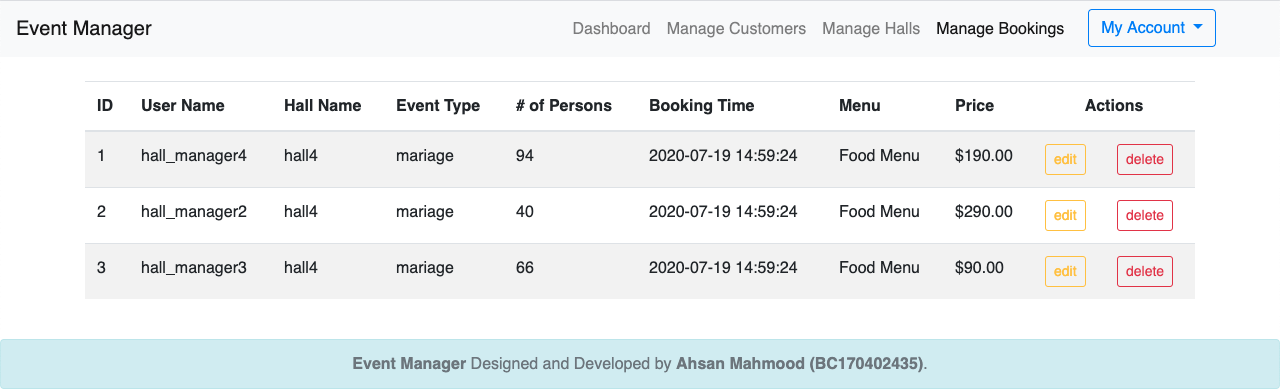












1. **Test Cases:**

|  |  |
| --- | --- |
|  | **Registration** |
| Actions | Open a registration form. |
| Pre-Conditions | User is logged into System. |
| Expected Result: | User is registered. after registration users are allowed to login. |
| Tested by | S2002687F7 |
| Result | Pass |

|  |  |
| --- | --- |
|  | **Login** |
| Actions | * User can login to its account * System will check the email and password |
| Pre-Conditions | The user information was previously saved to the system and the system will check. |
| Expected Result | After matching user info system allow the user to proceed. |
| Tested by | S2002687F7 |
| Result | Pass |

|  |  |
| --- | --- |
|  | **Logout** |
| Action | User will logout from its account. |
| Pre-Conditions | Before logout user can do any process which is allowed to its role. |
| Expected Result | After logout user can login again |
| Tested by | S2002687F7 |
| Result | Pass |

|  |  |
| --- | --- |
|  | **Add Hall** |
| Action | User(hall manager) will add new hall with required details |
| Pre-Conditions | Before logout user can add another hall. |
| Expected Result | New hall. |
| Tested by | S2002687F7 |
| Result | Pass. |

|  |  |
| --- | --- |
|  | **search Hall** |
| Action | User(customer) can search a hall by its name, price, location etc. |
| Pre-Conditions | Before logout user can search another hall. |
| Expected Result | Halls list. |
| Tested by | S2002687F7 |
| Result | Pass. |

|  |  |
| --- | --- |
|  | **Place booking** |
| Action | User(customer) can book a hall with required details |
| Pre-Conditions | Before logout user can book another hall. |
| Expected Result | New booking. |
| Tested by | S2002687F7 |
| Result | Pass. |

|  |  |
| --- | --- |
|  | **Give feedback** |
| Action | User(customer) can give feedback to a hall depending on his/her experience |
| Pre-Conditions | Before logout user can give feedback to another hall. |
| Expected Result | Hall feedback added. |
| Tested by | S2002687F7 |
| Result | Pass. |

|  |  |
| --- | --- |
|  | **Approve Hall** |
| Action | User(Admin) can approve a hall |
| Pre-Conditions | Before logout user can approve another hall. |
| Expected Result | Hall approve. |
| Tested by | S2002687F7 |
| Result | Pass. |

|  |  |
| --- | --- |
|  | **Disapprove Hall** |
| Action | User(Admin) can disapprove a hall |
| Pre-Conditions | Before logout user can disapprove another hall. |
| Expected Result | Hall disapprove. |
| Tested by | S2002687F7 |
| Result | Pass. |

|  |  |
| --- | --- |
|  | **Add Hall** |
| Action | User(hall manager) will add new hall with required details |
| Pre-Conditions | Before logout user can add another hall. |
| Expected Result | New hall. |
| Tested by | S2002687F7 |
| Result | Pass. |

|  |  |
| --- | --- |
|  | **Add Hall Food Menu** |
| Action | User(hall manager) can add hall food menu |
| Pre-Conditions | Before logout user can add another hall food menu. |
| Expected Result | Hall food menu. |
| Tested by | S2002687F7 |
| Result | Pass. |

|  |  |
| --- | --- |
|  | **Add Hall extra feature** |
| Action | User(hall manager) can add hall extra feature |
| Pre-Conditions | Before logout user can add another hall extra feature. |
| Expected Result | Hall extra feature. |
| Tested by | S2002687F7 |
| Result | Pass. |

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*