



National University
of computer and emerging sciences

PROJECT DOCUMENTATION

Course: Software Design and Analysis

Course code: CS-324

BS (CS)-D

Batch: Fall 2018

Submitted by:

Hassan Shahzad 18i-0441

Abeera Fatima 18i-0411

Sana Ali Khan 18i-0439

Azka Khurram 18i-0461

Submitted to:

Ma'am Maheen Arshad

Date of Submission:

26-12-2020

Event Management System

Acknowledgement

In the name of Allah, the Most Beneficent and the Most Merciful Alhamdulillah, all praises to Allah for the strengths and His blessings in completing this project. We would like to thank all the people who made this possible especially Ma'am Maheen Arshad for guiding us throughout the project. She was our source of inspiration and knowledge. Her guidance, advice and patient encouragement enabled us to work in this domain. Her helping us after office hours, solving our issues really made our project to reach the stage where it is right now.

We must also express our gratitude towards our fellow batchmates Adil Fayyaz and Zaynab Batool for contributing towards debugging of our errors at really odd times too.

Table of Contents

<i>Introduction</i>	<i>6</i>
<i>Project Scope</i>	<i>8</i>
<i>Project Stakeholders.....</i>	<i>10</i>
<i>High Level Goals of Stakeholders.....</i>	<i>12</i>
<i>List of Features</i>	<i>14</i>
<i>Functional Requirements.....</i>	<i>16</i>
<i>Non-Functional Requirements.....</i>	<i>21</i>
<i>Use Cases.....</i>	<i>24</i>
<i>Use Case Diagram.....</i>	<i>27</i>
<i>High-Level Use Cases</i>	<i>29</i>
<i>Expanded Use Cases</i>	<i>35</i>
<i>Domain Model.....</i>	<i>41</i>
<i>System Sequence Diagrams.....</i>	<i>43</i>
<i>Sequence Diagrams</i>	<i>47</i>
<i>Class Diagram.....</i>	<i>52</i>
<i>Interface Description (UI)</i>	<i>55</i>
<i>Database Description</i>	<i>67</i>
<i>Implementation Description</i>	<i>70</i>
1. <i>Development Setup:.....</i>	<i>70</i>
2. <i>Implementation:.....</i>	<i>71</i>
3. <i>Error Handling:.....</i>	<i>71</i>
<i>Flow Diagram</i>	<i>74</i>
<i>Testing</i>	<i>76</i>
<i>Results</i>	<i>78</i>
<i>Conclusion.....</i>	<i>80</i>

Chapter 1

Introduction

For our Software Design and Analysis project, we are designing a cross platform (*accessed through the internet and mobile applications*) Event Management System for a fictional Event Management firm named “A.S.H Events”. It will provide the clients of the firm with an easy-to-use platform for communicating with the organizers, whilst also reducing the firm’s labor load (*taking orders will be digitalized*). Our purpose is to override the problems prevailing in the practicing manual system and provide OUR clients (*A.S.H Events*) with a system which will link them to their account, store and categorize their orders (*according to dates/ most time required for planning/level of urgency etc.*) so they can function efficiently and provide their best service.

Our management system has 2 basic categories:

- For Employees of the Firm
- For Customers of the Firm

For the employees, this management system will provide them access to digitalize all their personnel records and manager would be given access to add/edit/remove an employee along with viewing the orders/vendors and more. Similarly, for customers, this management system will be providing them with ease of access. For a customer, this system will be divided into the following three sections:

1. Event Venue bookings and Scheduling
2. Catering Services
3. Multimedia Services

No formal knowledge is needed for the user to use this system as we tried to keep it as minimal as possible, making it a user-friendly software.

The clients will be able to select either pre-defined packages or customize a package according to his/her need. Clients will create an account to use the application, select the option for obtaining event management services and then fill in the respective sections with what they desire. All the information will be stored in a database, from which the firm will access it according to their needs.

Our Management System is not only digitalizing the booking of orders providing ease of access to the customers, but will also digitalize the firm’s functionalities such as adding/removing employees, vendors etc. In short, our management system will digitalize the whole firm along with its jobs.

Chapter 2

Project Scope

The scope of our EMS is to automate the existing manual system by the help of computerized equipments and a software that is fulfilling their requirements so their information can be stored for a longer period along with ease of access to that information. It manages the details of event, organizers, attendees, payments etc. The EMS is built at administrative access, hence only the manager is guaranteed the access to information mentioned above whereas the customer has access to only his/her information (*limited access*). Similarly, customer would also be benefitted as he/she can access this software from anywhere, as to access it the only requirement is having an internet connection.

Our project aims at business process automation. We tried to computerize as many processes of Event Management as we could such as:

- User can get a number of copies of the information he/she entered within seconds
- To utilize our resources in an efficient manner using automation
- Satisfy the user requirements
- Be easy to use for the user and operator
- Easy to operate
- Have a good interface
- Be expandable

Chapter 3

Project Stakeholders

A formal definition of a stakeholder is: “individuals and organizations who are actively involved in the project, or whose interests may be positively or negatively affected as a result of project execution or successful project completion” (*Project Management Institute (PMI®), 1996*).

For our EMS, we have following stakeholders:

- Manager
- Customer
- Employees
- Vendors
- Members of the Public

Chapter 4

High Level Goals of Stakeholders

Stakeholder	Goals in System
➤ Manager	<ul style="list-style-type: none"> ✓ Manager can view/edit and delete the events created by a customer. ✓ Manager can add/edit/delete an employee of his firm. ✓ Manager can view/edit all the vendors that are affiliated with the firm. ✓ Manager can edit his information. ✓ Manager can confirm a booking of an event.
➤ Customer	<ul style="list-style-type: none"> ✓ Create/Edit Account. ✓ Book an Event. ✓ View an Event.
➤ Employee	<ul style="list-style-type: none"> ✓ Login to the system. ✓ Edit Account Details.
➤ Vendors	<ul style="list-style-type: none"> ✓ Vendors are indirectly affected as whatever is booked using our EMS, needs to be informed to the vendors and they need to fulfill the customer's requirements. ✓ Following are the categories of the vendors: <ul style="list-style-type: none"> • Media • Catering
➤ Members of Public	<ul style="list-style-type: none"> ✓ These are the guests that will attend that particular event. ✓ They will also be indirectly affected.

Chapter 5

List of Features

Following is the list of some of the features of our Event Management System:

1. Minimal and Aesthetic UI (*User Interface*)
2. Easy to Use
3. Expandable Implementation
4. Accuracy in Work
5. Smooth Flow
6. Easy and Fast Retrieval of Information
7. Error Handling
8. Additional Comments for more information
9. Robust Database Backend
10. Cross-Platform Support
11. Access of Information Individually
12. Easy to Update Information
13. Well Designed and Minimal Reports (*e.g., Summary of event booked, Payment Confirmation*)
14. Decreases the Load of the Person Involved in Existing Manual System
15. Time Efficiency
16. Space Efficiency

Chapter 6

Functional Requirements

Following are the functional requirements of our EMS:

1. Add/Delete Client Orders:

- The system provides this facility to the manager and customer both. Customer can add his/her order whereas manager can add, view and delete customers' orders.

2. Create/Edit/Delete Client Account:

- Customer can create and edit his/her account details, whereas the manager can delete customers' account, along with editing and viewing it.

3. View Services:

- Customer has an option to view the list of services offered by the firm.

4. Cancel Bookings:

- Manager can cancel any customer's booking.

5. Data Entry to Database:

- All of the data inserted into the system is stored into a database in the backend.
- The database that we used for this project is "MySQL".

6. Customized Catering Menu:

- Customer is able to create a custom menu depending on his/her preferences.

7. Customized Media Services:

- Customer is able to customise their media requirements.

8. Generate Event ID:

- A unique id is allocated to each event in order to access it easily.

9. Generate Customer ID:

- Each customer will be assigned a unique id associated with his/her email address.

10. Generate Employee ID:

- Each employee will be assigned a unique id associated with his/her email address.

11. Generate Catering ID:

- A unique id is associated with each caterer (*vendor*) to access their information from database.

12. Generate Media ID:

- A unique id is associated with each media requirement in the database for ease of access.

13. Generate Menu ID:

- A unique id is associated with each menu in the database for ease of access.

14. Generate Studio ID:

- A unique id is associated with each studio in the database for ease of access.

15. Generate Venue ID:

- A unique id is associated with each venue in the database for ease of access.

16. Generate Total Bill:

- In the end a total bill will be computed from all the choices selected by the customer inclusive of taxes and this bill will be displayed on the screen which user can print.

17. Enter Payment Details:

- Customer will be asked to enter payment details after selecting payment option.

18. Order Confirmation:

- This screen will be displayed showing that the order has been confirmed and awaiting approval.

19. Tax Calculation:

- Tax will be calculated as per government regulations and added to the final bill.

20. Book Event:

- Customer can book event as per his/her requirements.

21. Add/Remove Employee:

- Manager will have the facility to add or remove an employee from the database.

22. Approve/Deny Order:

- All the confirmed orders will need to be approved from the manager.

23. Add/Remove Vendors:

- Manager has the facility to add or remove a vendor (*caterer or studio*).

24. View Orders:

- Customer is able to view his order. Moreover, manager can view a particular order and he/she can view all orders too.

25. Client Sign In:

- When a customer signs in, his/her credentials are verified from the database and then allowed access into the system.

26. Manager Sign In:

- When a manager signs in, his/her credentials are verified from the database and then allowed access into the system.

27. Employee Sign in:

- When an employee signs in, his/her credentials are verified from the database and then allowed access into the system.

28. Pop-up Message:

- Whenever there is an error, a popup message with the error details will be displayed.

29. Error Alarms:

- Whenever there is an error, a notification beep will be played to gain the user's attention followed by an error popup.

30. Sign Out:

- When a user clicks "Sign Out" button, they are logged out of the system and the flow returns back to the main menu screen.

Chapter 7

Non-Functional Requirements

1. Minimal Interface:

- Interface should be simple and easy for users to adapt to.

2. Speed:

- System should be fast and responsive, with as minimal lag as possible.

3. Portability:

- There should be abstraction between the application logic and the system interface – system should be modular

4. Reliability:

- System should be stable, not prone to crashes. User should expect to have a consistent experience every time they use it.

5. Scalability:

- System should be extendable if required, and adapt to greater number of users. It should be open for more functionalities to be added if required.

6. Security:

- User data and/or other private details must be kept confidential and only visible to those authorized to view them.

7. Data Integrity:

- Data stored should not be compromised, and should be accurate and reliable.

Chapter 8

Use Cases

A use case is a methodology used in system analysis to identify, clarify and organize system requirements. The use case is made up of a set of possible sequences of interactions between systems and users in a particular environment and related to a particular goal.

For our EMS, we have the following use cases:

- Creating Client Account
- Editing Client Account
- Deleting Client Account
- Creating Booking
- Deleting Booking
- Viewing Booking
- Editing Booking
- Viewing Services
- Customizing Media Requirements
- Customizing Menu
- Entering Bank Details
- Adding Employee
- Editing Employee

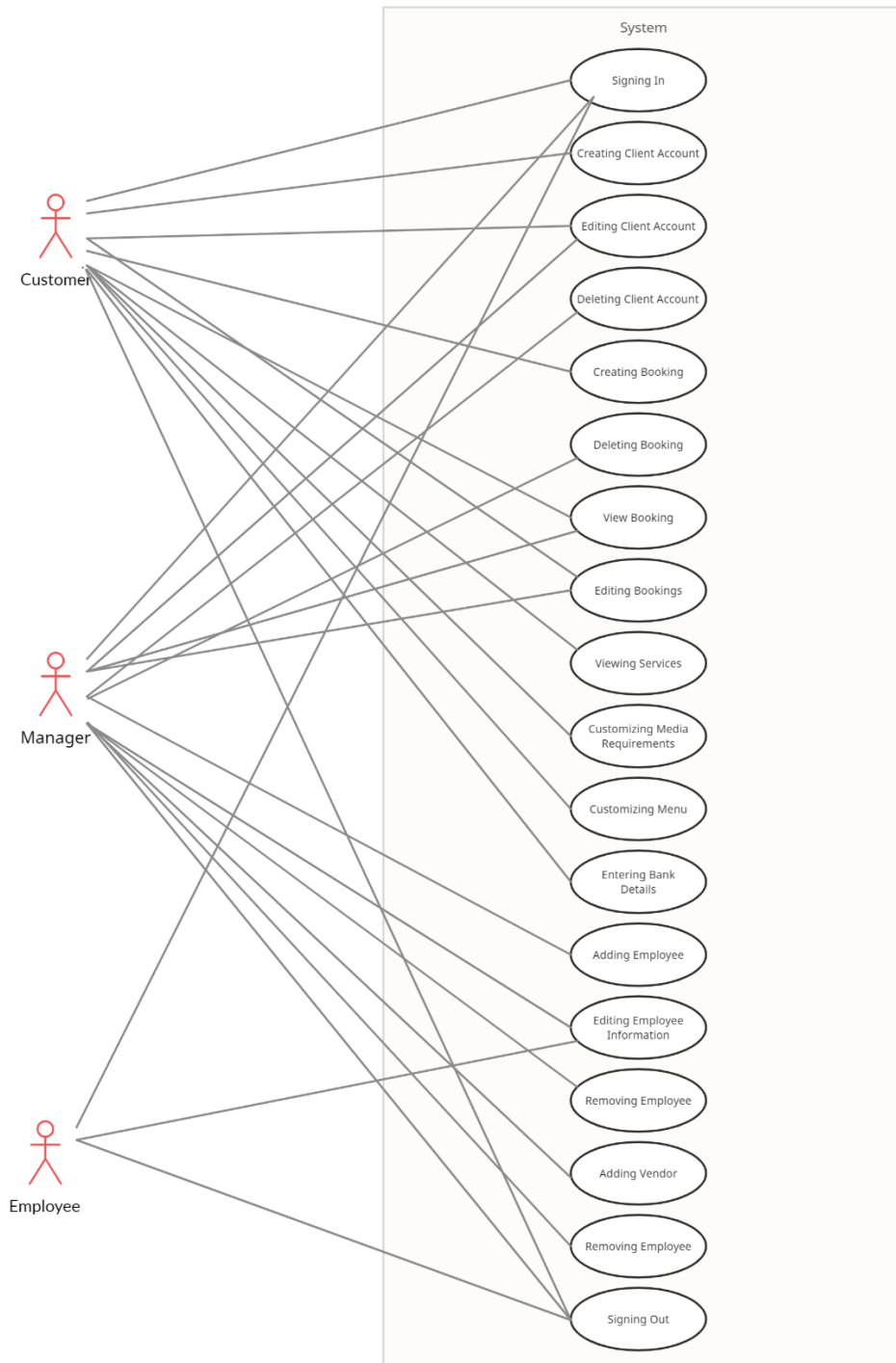
- Removing Employee
- Adding Vendor
- Removing Vendor
- Signing In
- Signing Out
- Viewing All Bookings

Chapter 9

Use Case Diagram

Use case diagram is a behavioral UML diagram type and frequently used to analyze various systems. They enable you to visualize the different types of roles in a system and how those roles interact with the system.

The use case diagram of our EMS is as follows:



Chapter 10

High-Level Use Cases

The high-level use case is simply a summary description of the task. Its purpose is to provide just enough detail to give you a feel for its complexity.

The high-level use cases for our EMS are as follows:

1:	Use Case:	Creating Client Account
	Actors:	Client
	Type:	Primary
	Description:	Client opens the software. Then he/she clicks on "Register". After entering information and creating account, the information is stored in the database.

2:	Use Case:	Editing Client Account
	Actors:	Client
	Type:	Primary
	Description:	Client decides to update his/her contact number in account information. They log in to their account and go to profile, make the desired changes and save the changes.

3:	Use Case:	Deleting Client Account
	Actors:	Manager
	Type:	Primary
	Description:	Clients want to delete his/her account. They will contact the manager and the manager will delete the account from the database.

4:	Use Case:	Creating Booking
	Actors:	Client
	Type:	Primary
	Description:	Client opens the software. After signing in, he/she clicks on “Book an Event”. After entering all required details, his event is confirmed and sent to manager for approval.

5:	Use Case:	Deleting Booking
	Actors:	Manager
	Type:	Primary
	Description:	Client changes his/her mind about booking an event. He/She will request the manager who will upon reviewing the progress of event, cancel the event. Note that advance will be refunded only if booking is deleted within a certain period of time.

6:	Use Case:	Viewing Booking
	Actors:	Manager, Customer
	Type:	Primary
	Description:	Manager and Customer, both can view a specific booking by simply entering the event ID.

7:	Use Case:	Editing Booking
	Actors:	Client, Manager
	Type:	Primary
	Description:	Client and manager can both edit the booking information for a specific event.

8:	Use Case:	Viewing Services
	Actors:	Customer
	Type:	Primary
	Description:	Customer can also view the list of services provided by the firm.

9:	Use Case:	Customizing Media Requirements
	Actors:	Client
	Type:	Primary
	Description:	Client can customize the media services he/she wants according to personal requirements.

10:	Use Case:	Customizing Menu
	Actors:	Client
	Type:	Primary
	Description:	Client can customize the food menu he/she wants according to personal requirements.

11:	Use Case:	Entering Bank Details
	Actors:	Client
	Type:	Primary
	Description:	After confirming a booking, the customer will be asked to select his/her payment option. For this the customer will have to enter their payment details.

12:	Use Case:	Adding Employee
	Actors:	Manager
	Type:	Primary
	Description:	A new employee joins the firm. Manager has the authorization to create their account by entering their details.

13:	Use Case:	Editing Employee
	Actors:	Manager, Employee
	Type:	Primary
	Description:	Employee can edit his/her information to some extent whereas the manager can edit every information about the employee.

14:	Use Case:	Removing Employee
	Actors:	Manager
	Type:	Primary
	Description:	An employee retires/leaves the firm, then the manager will have the facility to delete their account from the database.

15:	Use Case:	Adding Vendor
	Actors:	Manager
	Type:	Primary
	Description:	A contract is signed with a new vendor. Manager has the facility to add any new vendor by simply entering their details in the database.

16:	Use Case:	Removing Vendor
	Actors:	Manager
	Type:	Primary
	Description:	The contract has been expired with a specific vendor. Manager has the facility to remove any vendor.

17:	Use Case:	Signing In
	Actors:	Manager, Customer, Employee
	Type:	Primary
	Description:	In order to access the software, each user needs to sign in to get authenticated.

18:	Use Case:	Signing Out
	Actors:	Manager, Employee, Customer
	Type:	Primary
	Description:	After the user has finished using the software, they must sign out to ensure privacy and security.

19:	Use Case:	Viewing All Bookings
	Actors:	Manager
	Type:	Primary
	Description:	The manager has been facilitated such that he can view all of the booking currently under progress from the database.

Chapter 11

Expanded Use Cases

1:	Use case name:	Book an Event
	Scope:	Event Management System
	Level:	User-goal
	Primary Actor:	Client, Manager
	Stakeholders and Interests:	<ul style="list-style-type: none">✓ Client wants to book an event and will provide all the details related to that event✓ Manager will approve/deny the event
	Pre-conditions:	Client signs in to the software in order to book the event.
	Post-Conditions:	Event is successfully booked and stored in the database and the total bill is displayed.

Main Success Scenario:

Actor Action	System Response
1: User selects the option "Book an Event".	
	2: System opens a window with fields to be filled with the details of that event.
3: User enters all the details in the respective fields.	
	4: System books an event and stores it into the DB.
5: User logs out.	

Extensions:

- Customer enters invalid number of guests (*i.e., a non-numeric value*)
 - a. An error is detected and error message is displayed as a pop-up with a beep
 - b. User is asked to re-enter the number of guests
- Customer forgets to enter the date of event
 - a. An error is detected and error message is displayed as a pop-up with a beep
 - b. User is asked to select the date of the event
- Customer selects a date that has past
 - a. An error is detected and error message is displayed as a pop-up with a beep
 - b. User is asked to enter the correct date

2:	Use case name:	Add Employee
	Scope:	Event Management System
	Level:	User-goal
	Primary Actor:	Manager
	Stakeholders and Interests:	<ul style="list-style-type: none"> ✓ Manager is responsible for adding a new employee to the database. ✓ Oversees every employee.
	Pre-conditions:	Manager is signed into the system. Manager selects the option of “Adding an Employee”.
	Post-Conditions:	New employee is successfully stored in the DB and manager logs out.

Main Success Scenario:

Actor Action	System Response
1: Manager selects “Add Employee” option	
	2: Asks for employee information
3: Enters the employee’s information	
	4: Saves the information into the DB
5: Manager logs out of the system.	

Extensions:

- Managers misses to add cnic of the employee
 - a. An error is detected
 - b. The error message is displayed as a pop-up on the interface
 - c. Manager is asked to re-enter the cnic of the employee
- Managers adds a non-numeric value in cnic
 - a. An error is detected
 - b. The error message is displayed as a pop-up on the interface
 - c. Manager is asked to re-enter the cnic of the employee

1:	Use case name:	Edit Employee Information
	Scope:	Event Management System
	Level:	User-goal
	Primary Actor:	Employee, Manager
	Stakeholders and Interests:	<ul style="list-style-type: none"> ✓ Employee can edit his/her account information with limited access. ✓ Manager can however completely access and edit employee's information
	Pre-conditions:	Employee/Manger signs into the system and clicks on edit information.
	Post-Conditions:	Updated information is successfully stored in Db and manager/employee sign out.

Main Success Scenario:

Actor Action	System Response
1: Employee/Manger clicks on "Edit Employee"	
	2: A screen with information is displayed.
3: Employee/Manger makes the required changes	
	4: The updated information is saved into the DB.
5: Employee/Manger logs out of the system	

Extensions:

- Manager enters negative account number
 - a. An error is detected
 - b. The error message is displayed as a pop-up on the interface
 - c. Manager is asked to re-enter the account number of the employee

1:	Use case name:	Creating Client Account
	Scope:	Event Management System
	Level:	User-goal
	Primary Actor:	Customer
	Stakeholders and Interests:	✓ Customer needs to create his account in order to access the software.
	Pre-conditions:	Client wants to book an event. He/She needs to create an account. He/She clicks on "Register"
	Post-Conditions:	Account was successfully created and he/she can now log in.

Main Success Scenario:

Actor Action	System Response
1: Customer selects option "Register"	
	2: System opens a screen having information fields
3: Customer fills information in those fields	
	4: The information is stored into the system
	5: Customer is taken back to sign in screen
6: Customer signs in	

Extensions:

- Customer enters age less than 18 years
 - a. An error is detected
 - b. The error message is displayed as a pop-up on the interface displaying ***"The minimum age requirement to create an account is 18 years old"***
 - c. Customer is asked to enter the age again
- Customer enters an invalid email
 - a. An error is detected
 - b. The error message is displayed as a pop-up on the interface
 - c. Customer is asked to re-enter his email address

1:	Use case name:	Edit Customer Information
	Scope:	Event Management System
	Level:	User-goal
	Primary Actor:	Customer, Manager
	Stakeholders and Interests:	<ul style="list-style-type: none"> ✓ Customer can edit his/her account information with limited access. ✓ Manager can however completely access and edit customer's information
	Pre-conditions:	Customer/Manger signs into the system and clicks on edit information.
	Post-Conditions:	Updated information is successfully stored in Db and manager/customer sign out.

Main Success Scenario:

Actor Action	System Response
1: Customer/Manger clicks on "Edit Customer"	
	2: A screen with information is displayed.
3: Customer/Manger makes the required changes	
	4: The updated information is saved into the DB.
5: Customer/Manger logs out of the system	

Extensions:

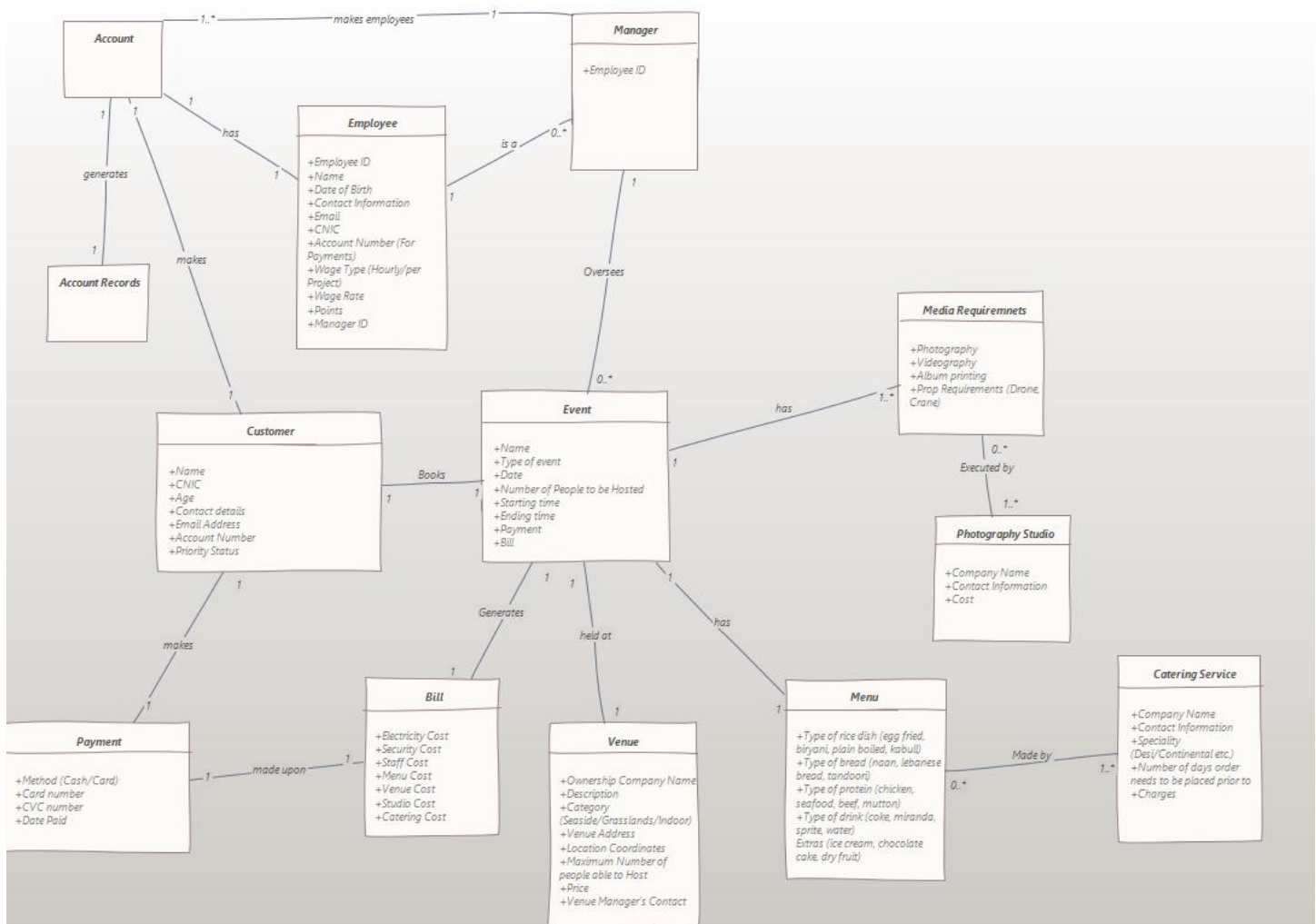
- Manager enters a negative age
 - a. An error is detected
 - b. The error message is displayed as a pop-up on the interface
 - c. User is asked to re-enter the age
- Managers tries to edit cnic of the customer
 - a. An error is detected
 - b. The error message is displayed as a pop-up on the interface
 - c. Manager is told that he/she cannot edit the cnic of the customer

Chapter 12

Domain Model

In software engineering, a domain model is a conceptual model of the domain that incorporates both behavior and data. The model can then be used to solve problems related to that domain. The domain model is a representation of meaningful real-world concepts pertinent to the domain that need to be modeled in software.

The domain model diagram of our EMS is as follows:



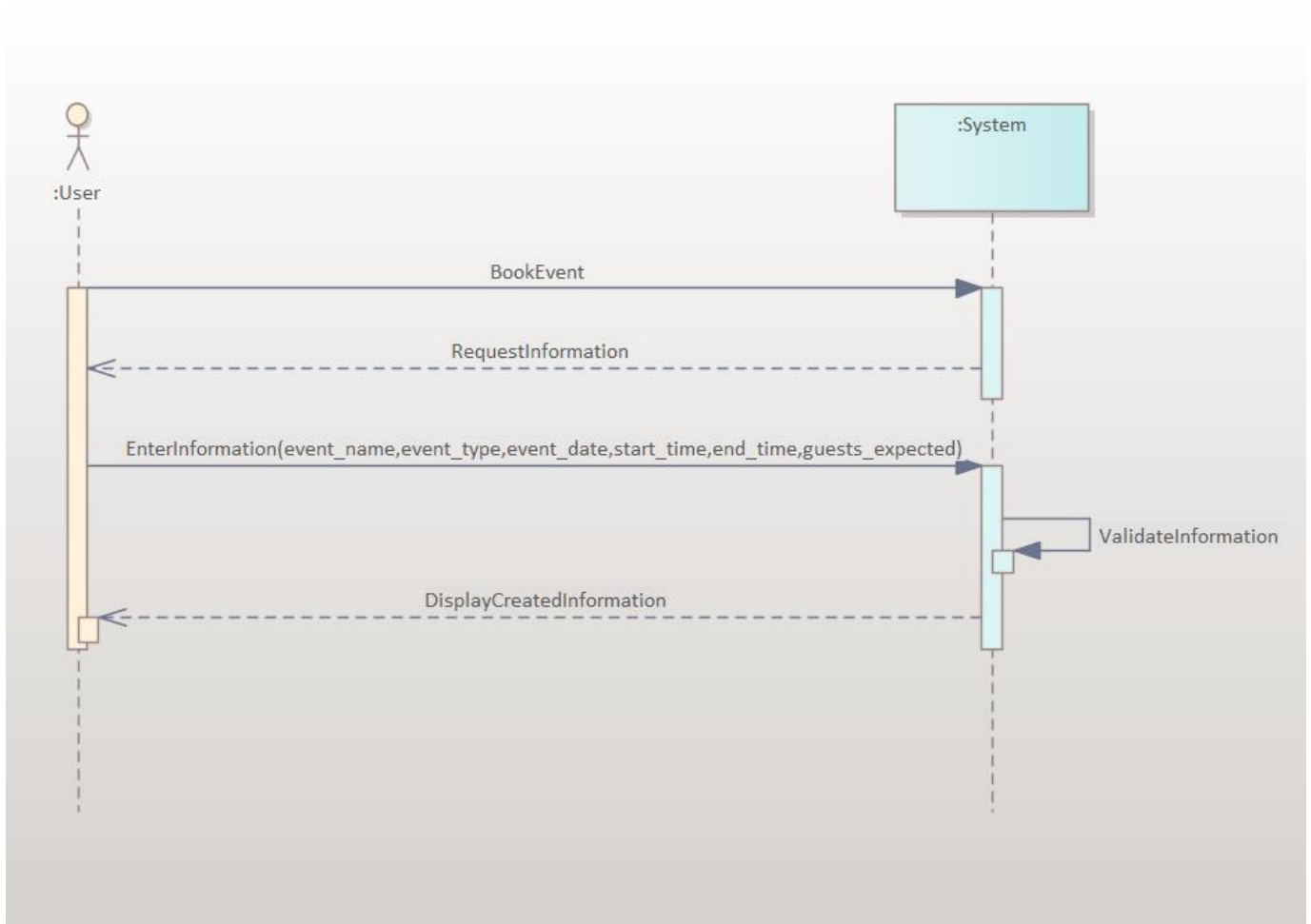
Chapter 13

System Sequence Diagrams

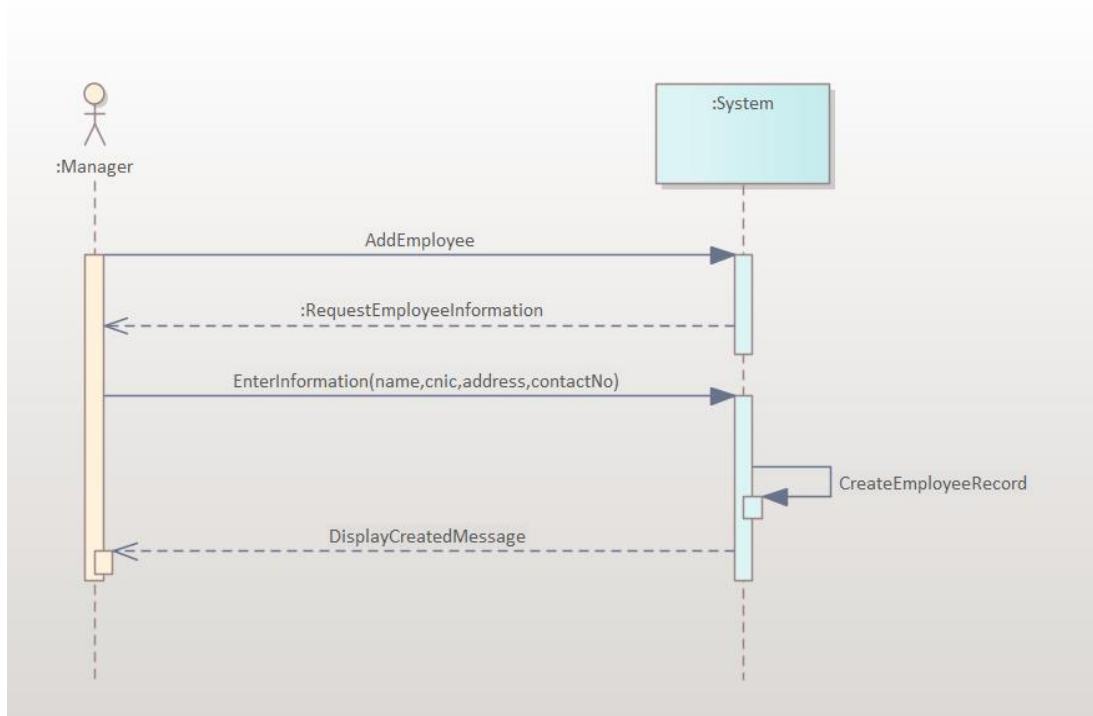
In software engineering, a system sequence diagram is a sequence diagram that shows, for a particular scenario of a use case, the events that external actors generate, their order, and possible inter-system events. System sequence diagrams are visual summaries of the individual use cases.

The system sequence diagrams of the 5 selected use cases from our EMS are as follows:

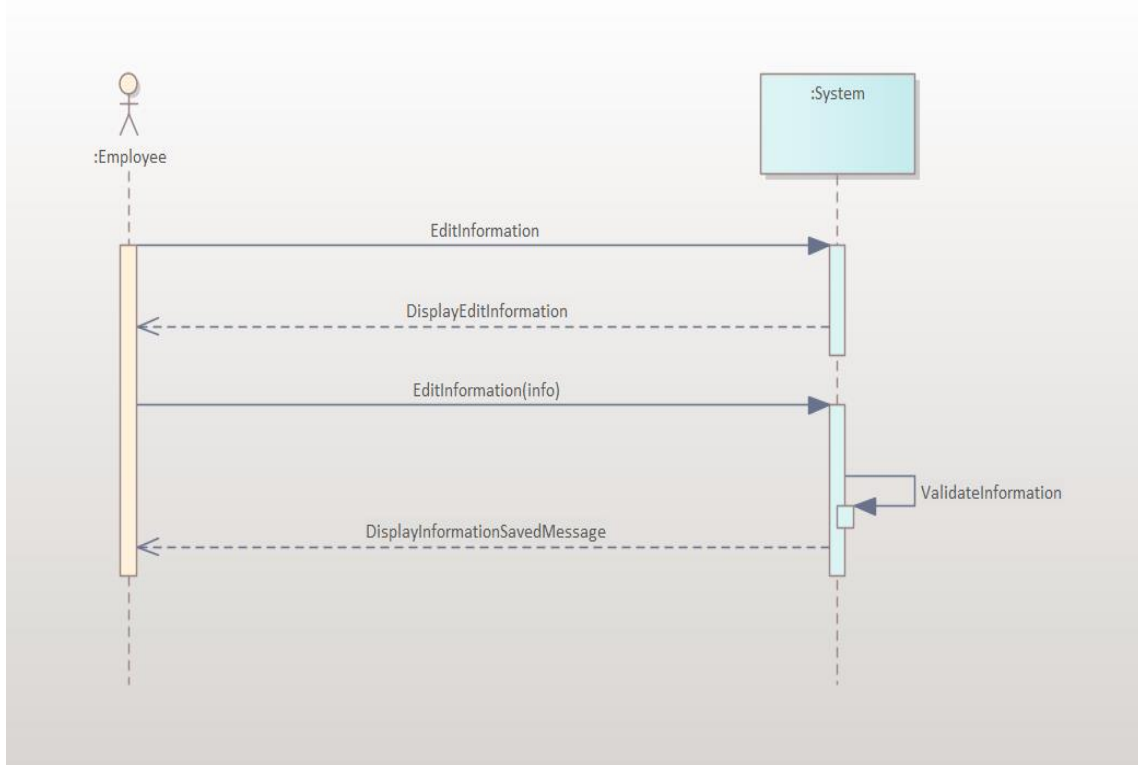
1. Book Event:



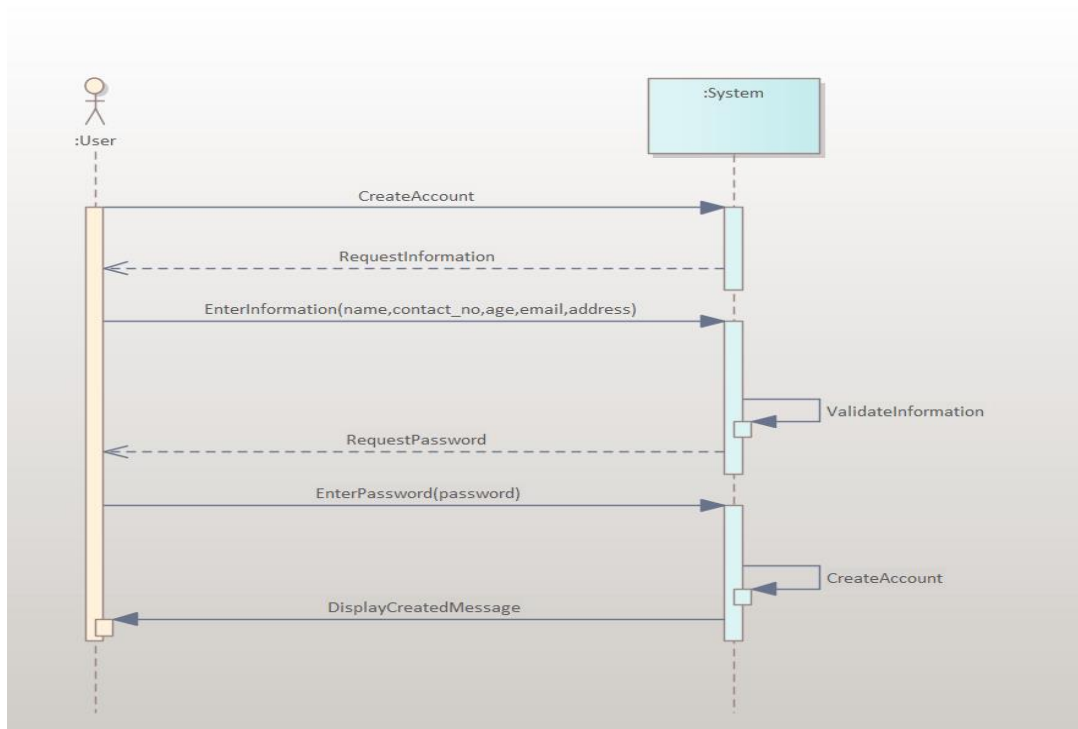
2. Add Employee:



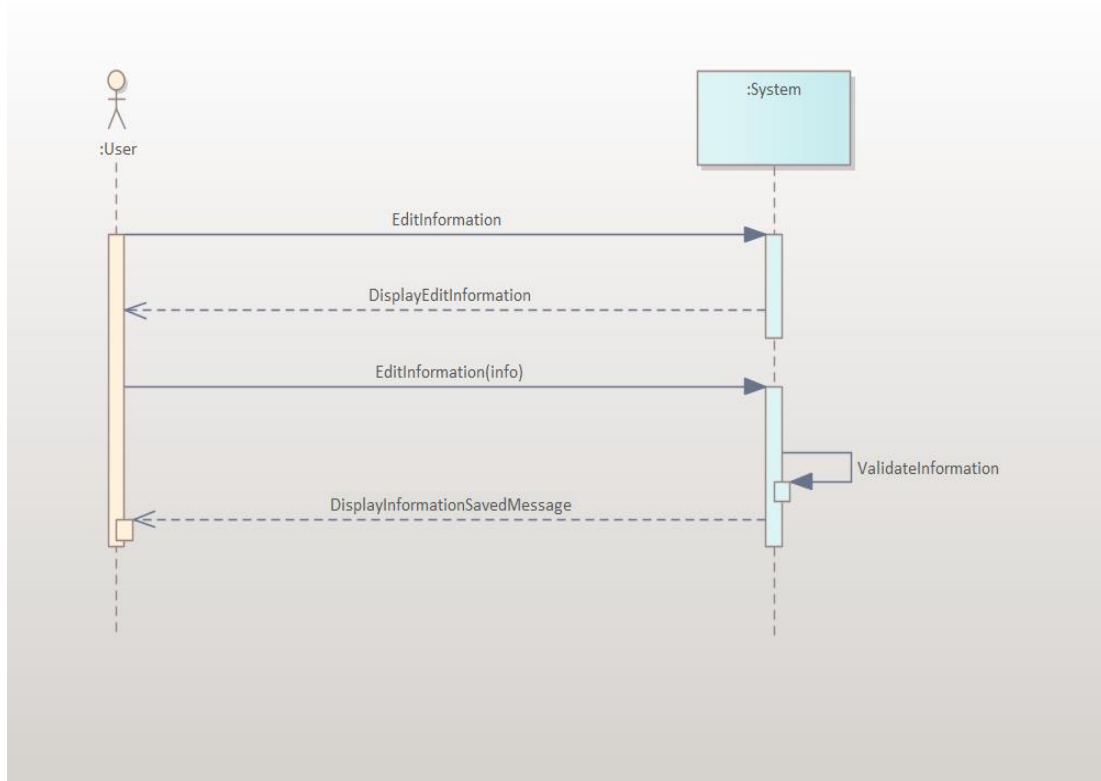
3. Edit Employee Information:



4. Add Customer Account:



5. Edit Customer Information:



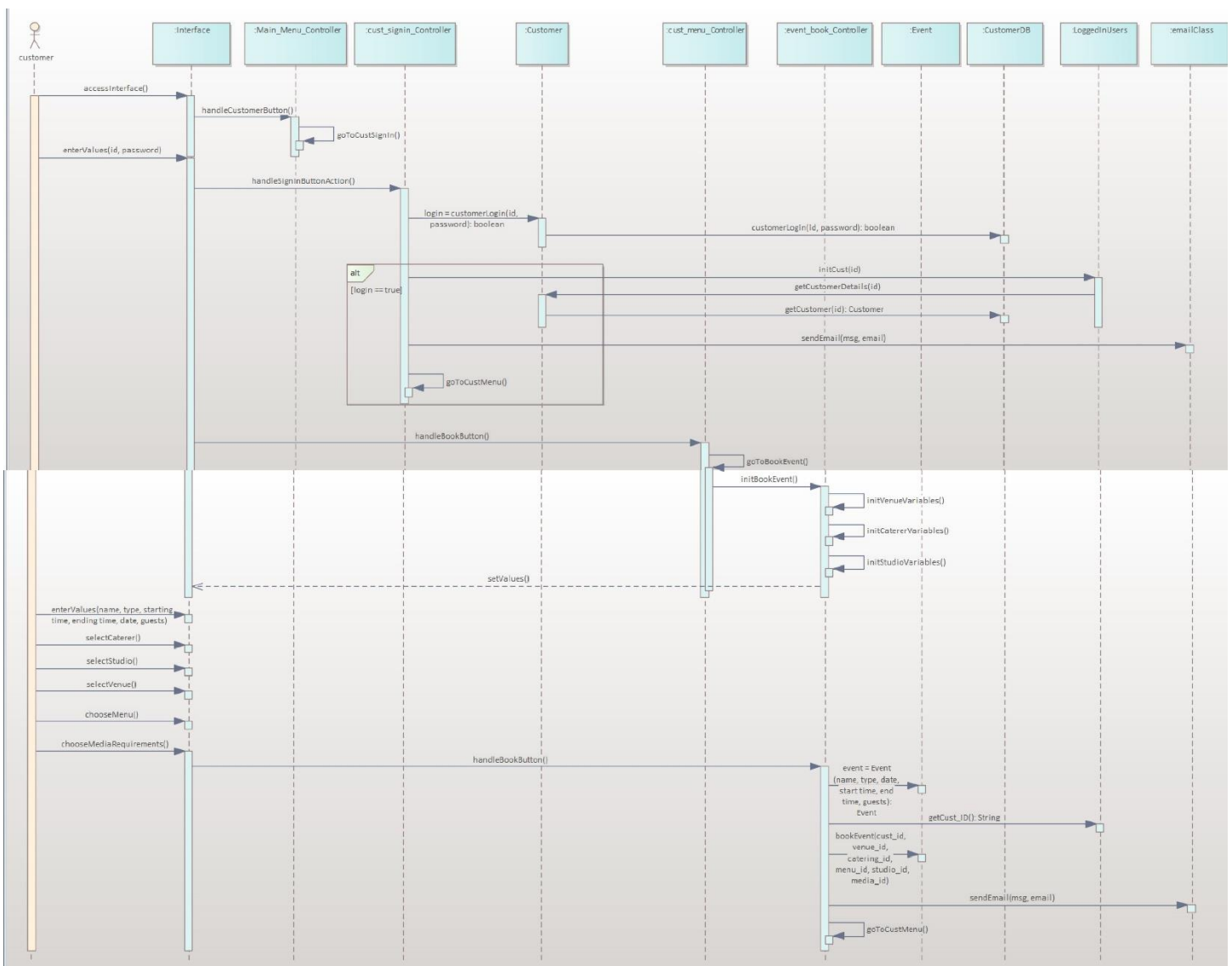
Chapter 14

Sequence Diagrams

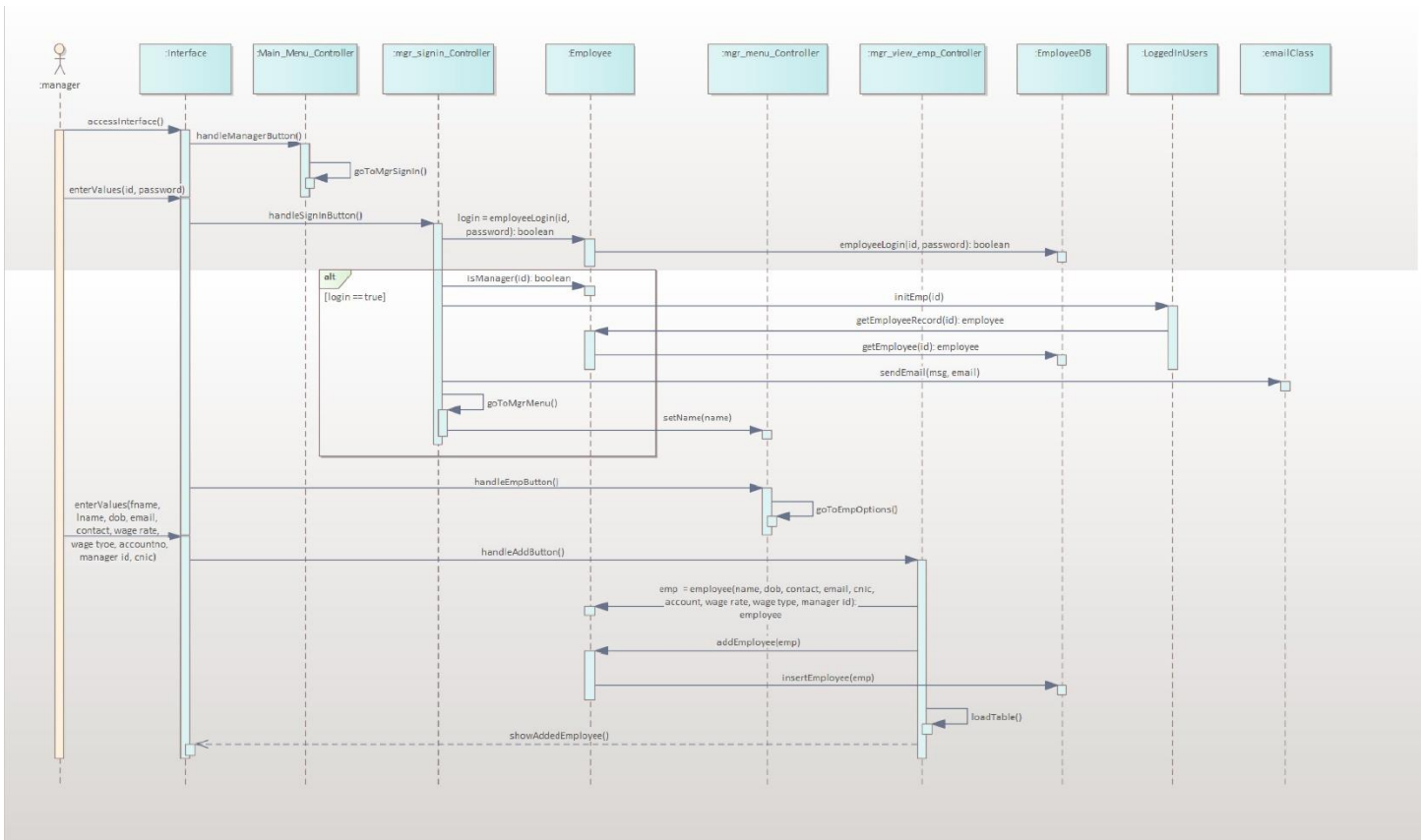
A sequence diagram shows object interactions arranged in time sequence. It depicts the objects involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.

The sequence diagrams of the 5 selected use cases from our EMS are as follows:

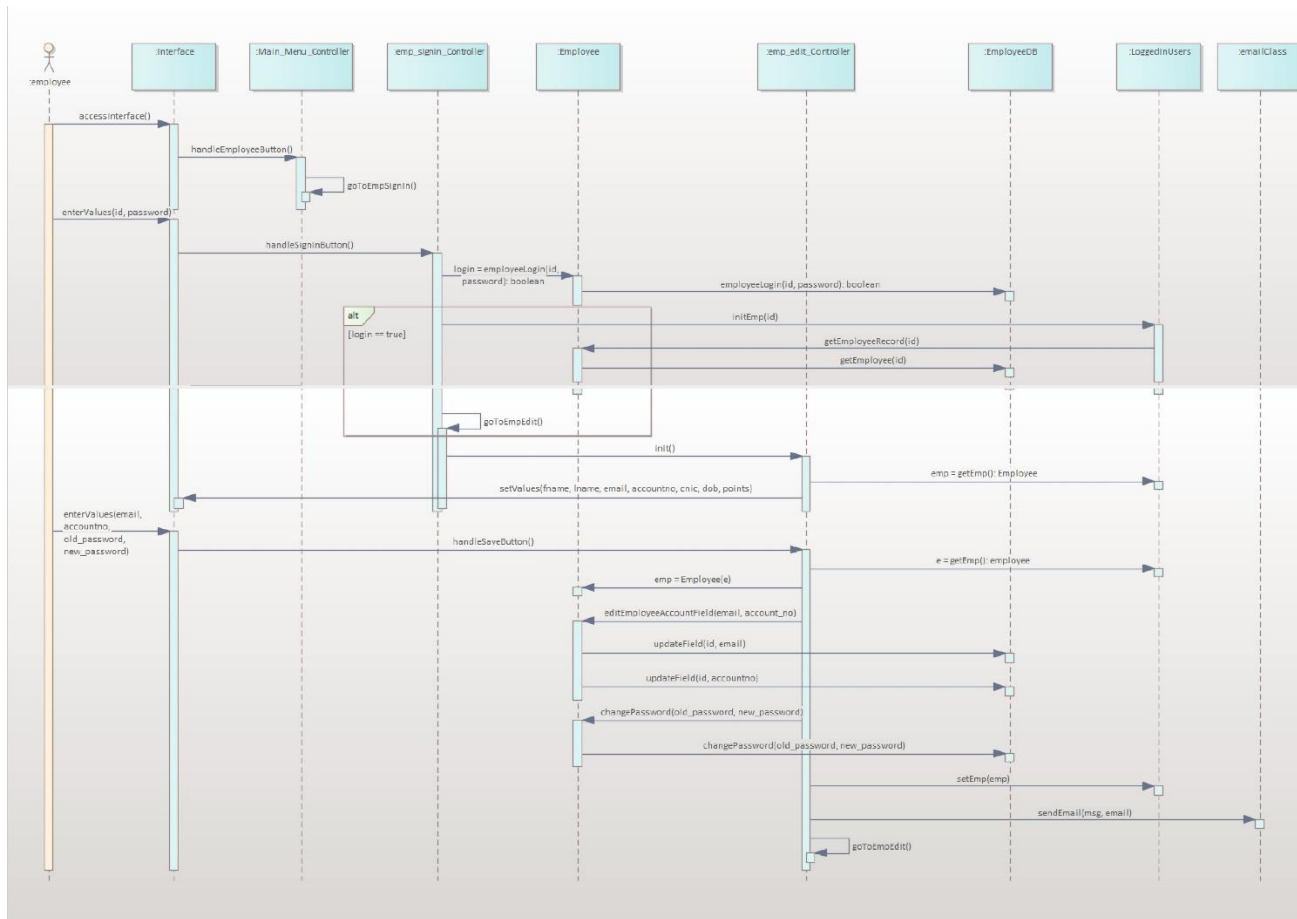
1. Book Event:



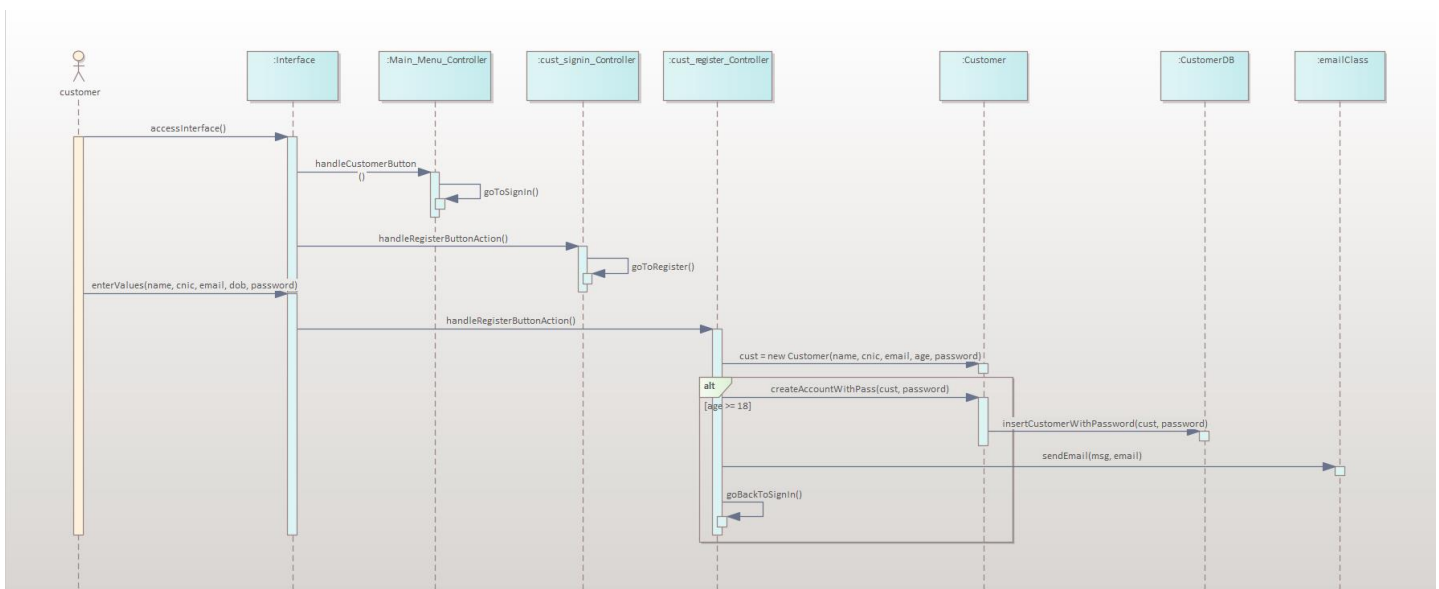
2. Add Employee:



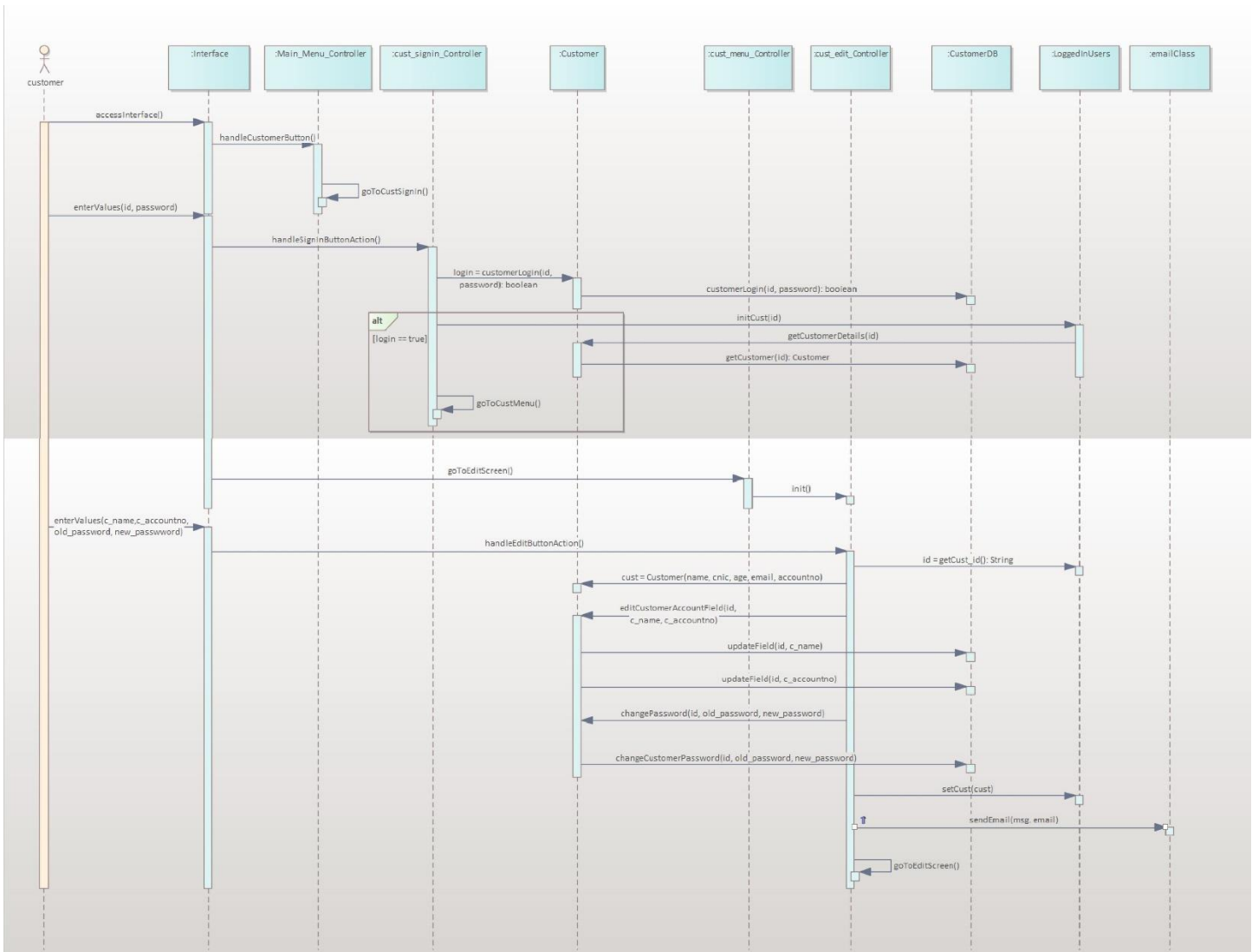
3. Edit Employee Information:



4. Add Customer Account:



5. Edit Customer Account:

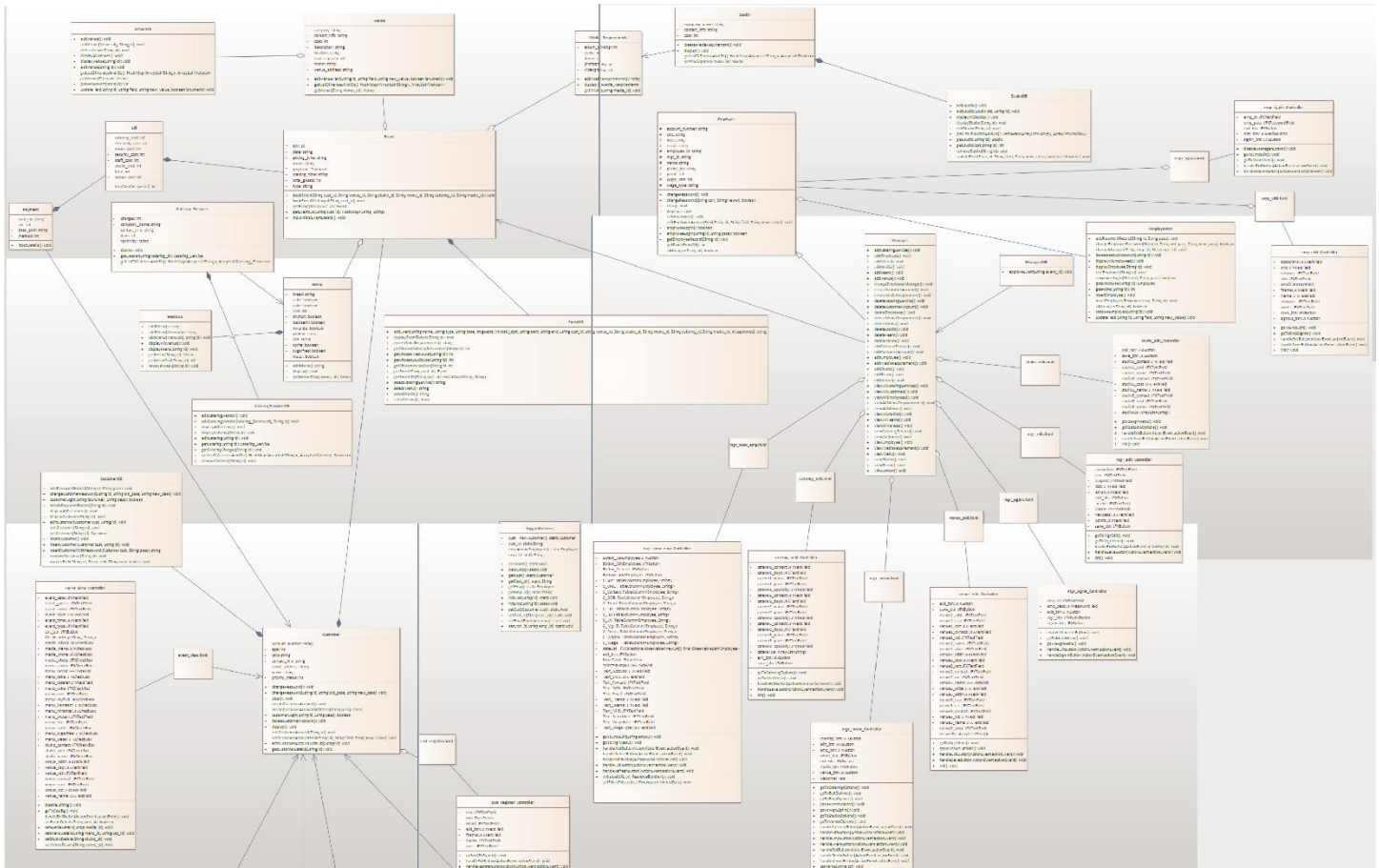


Chapter 15

Class Diagram

In software engineering, a class diagram in the Unified Modeling Language is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations, and the relationships among objects.

The class diagram of our EMS is as follows:



Chapter 16

Interface Description (UI)

At the most basic level, the user interface (UI) is the series of screens, pages, and visual elements—like buttons and icons—that enable a person to interact with a product or service. Interface has the most pivoted role in the whole implementation as in order to please the customers, the software needs to have an appealing UI Design.

So, we had to brainstorm a lot regarding the interface designs and concepts and we shortlisted many too. After designing and analysis of the designs, we discarded many and picked our final design.

For our interface, we wanted it to be aesthetic along with being minimalist at the same time. We wanted it to give out a happy concept to the users keeping it simple and self-explanatory. Our interface stood out from the rest of our competitors because of the uniqueness of our designs. We also tried to synchronize the images with the context as much as we can so they may be self-explanatory.

For our front-end, we decided to use Java FX instead of Java Swing as it had numerous options and functionalities and better designs. Though Java FX is considered a bit complicated than Java Swing, which we also had to agree with during our implementation, but the expected output turned out to be better than we imagined due to which we decided to stick to it.

We used the following libraries in our interface:

- ✓ **Java FX SDK 15.0.1**
- ✓ **Jfoenix 9.0.10** *(used for better designing)*
- ✓ **MySQL connector java 8.0.22** *(used to connect our implementation with DB)*

Some of the examples from our UI design are as follows:

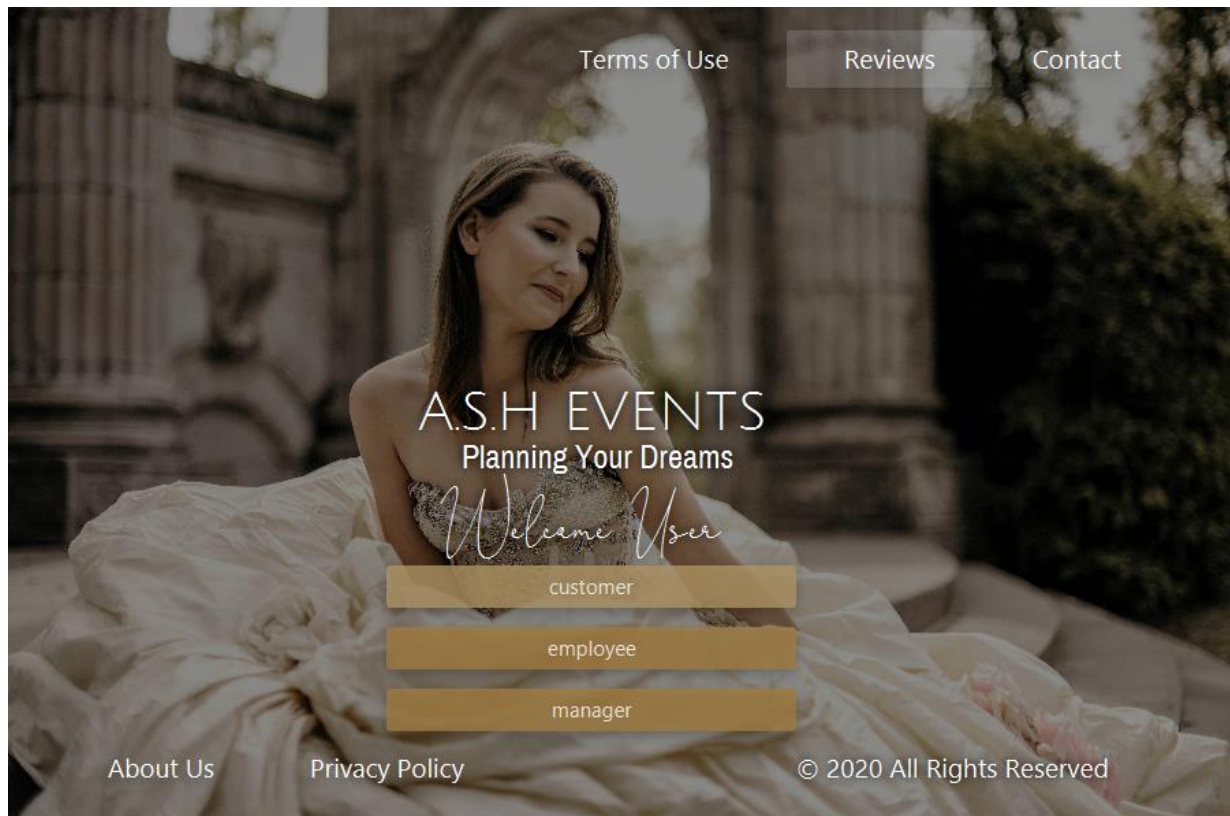


Fig 1. Start Screen of the Software

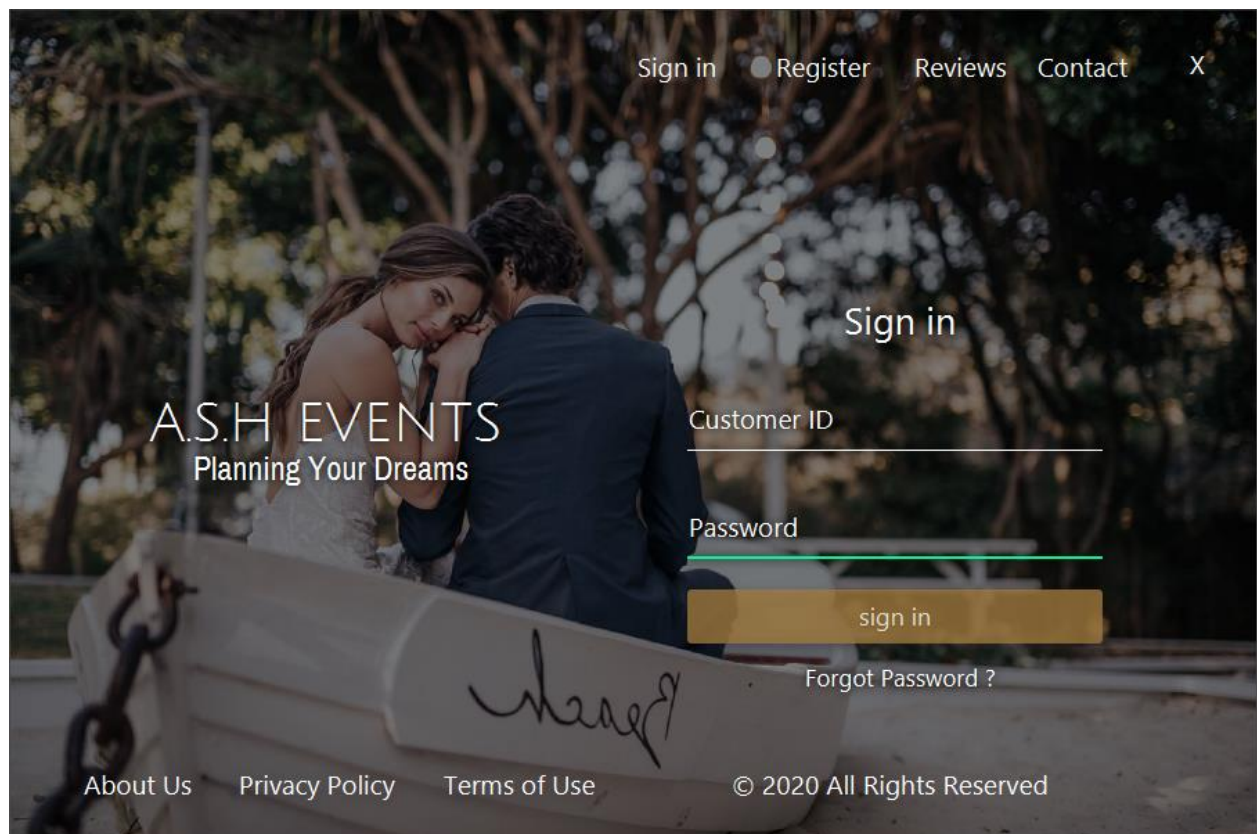


Fig 2. Sign in Screen of Customer

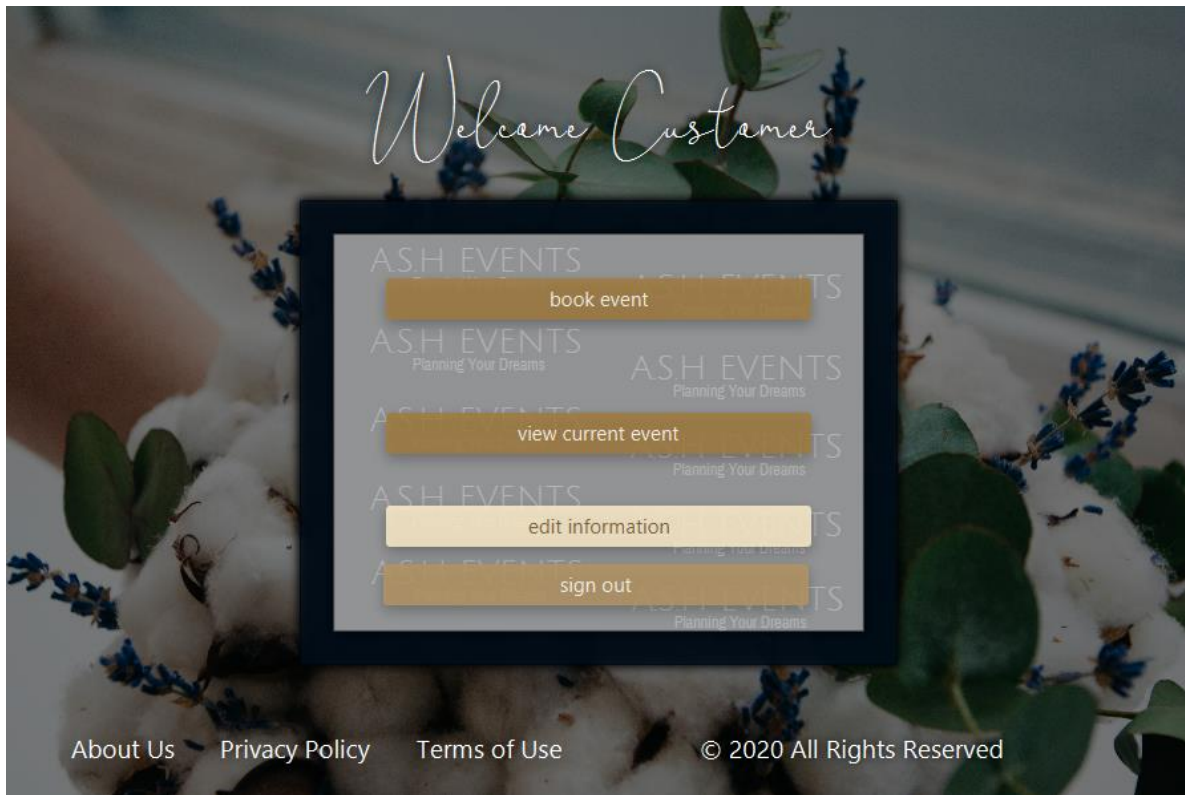


Fig 3. *Welcome Screen of Customer*

If the client chooses to book an event, the flow of interfaces will be as follows:

 The image displays the "Book an Event with Us" screen. The background features a close-up of a bouquet of white flowers and greenery. The title "Book an Event with Us" is written in a large, white cursive font at the top. In the top right corner, there is a small grey square button with a white "X" icon. Below the title, there is a form with two columns of input fields. Each field has a grey label box on the left and a white input area on the right. The fields are: "Event Name", "Event Type", "Event Date", "Starting Time", "Ending Time", "Price Generated", "Event Name", "Type", "Date", "Time", "Guests Expected", and "Price Generated".

Fig 4. *Event Details Screen*

Select a Venue

Three identical venue selection forms are displayed side-by-side. Each form includes a selection checkbox and the following fields:

- Name
- Location
- Address
- Capacity
- Category
- Contact
- Cost

Fig 5. Venue Choice Screen

Select a Caterer

Three identical caterer selection forms are displayed side-by-side. Each form includes a selection checkbox and the following fields:

- Name
- Contact
- Speciality
- Price
- Pre-Booking Days

Fig 6. Caterer Choice Screen



Fig 7. Menu Choice Screen

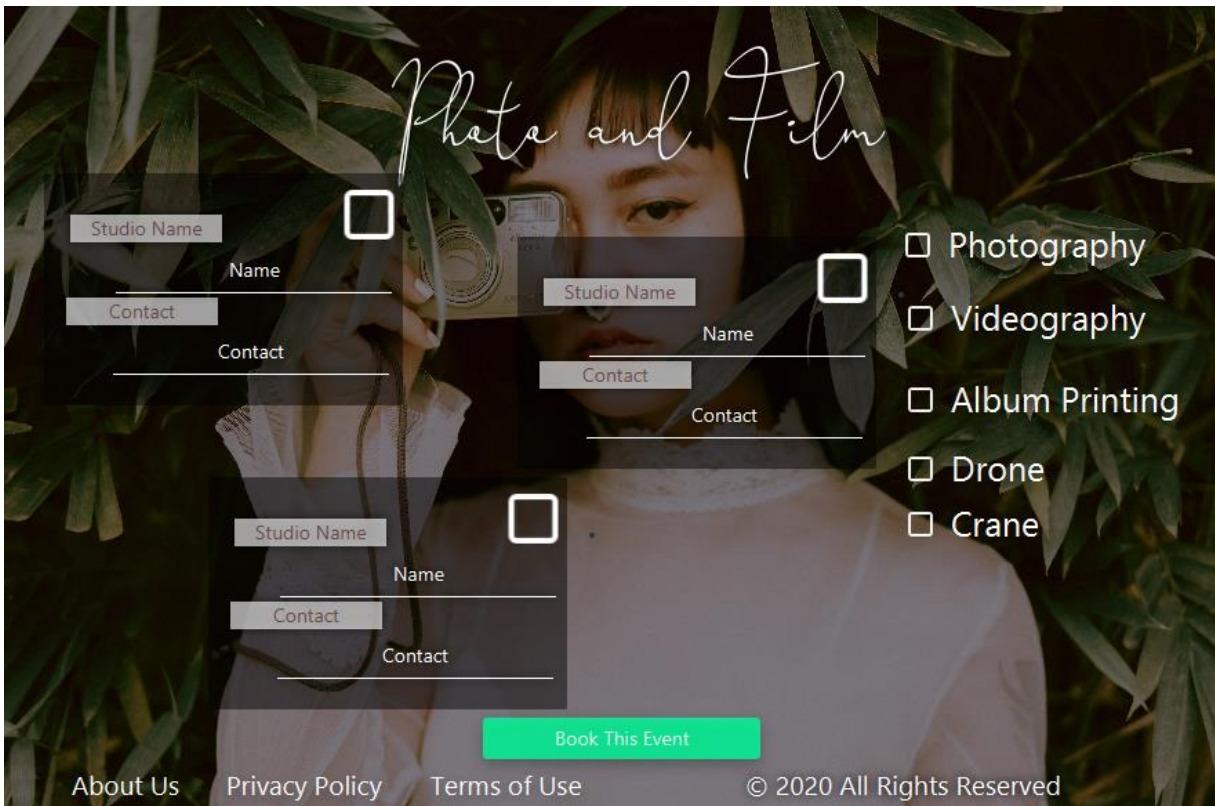


Fig 8. Studio Choice Screen

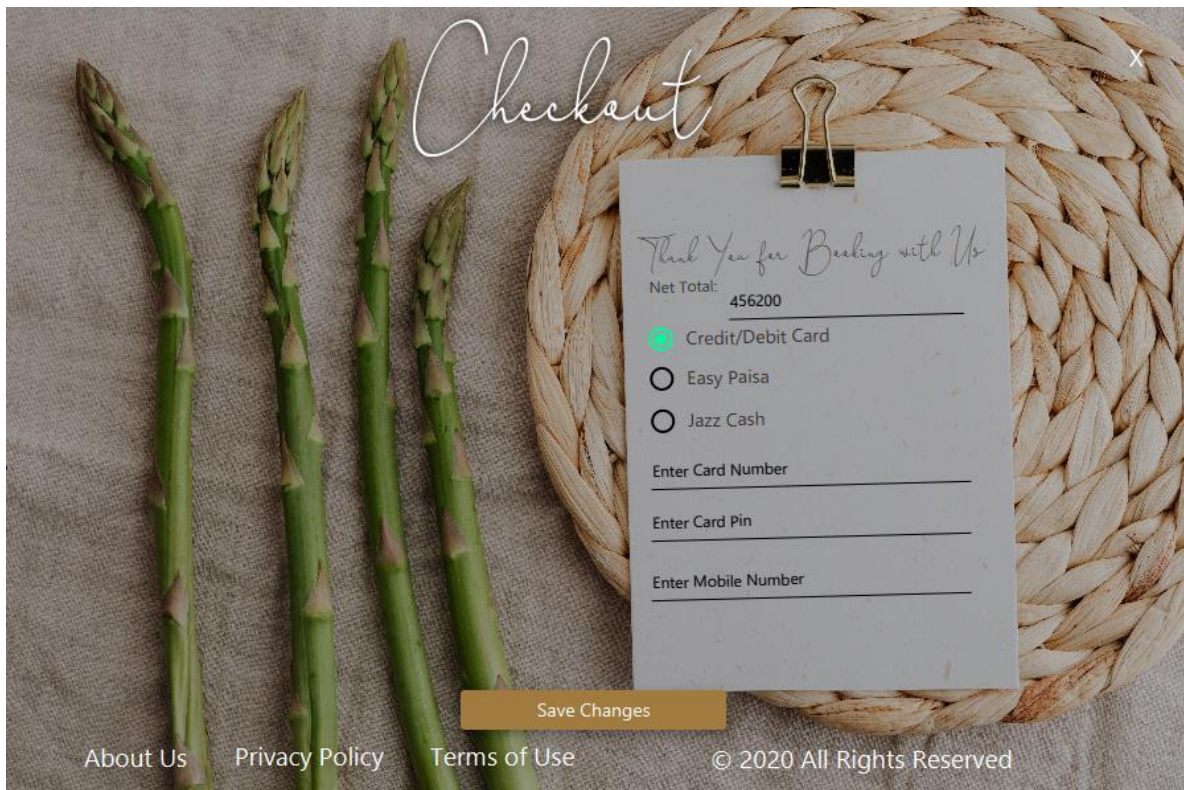


Fig 9. Checkout Screen

The client can also edit his/her information as follows:

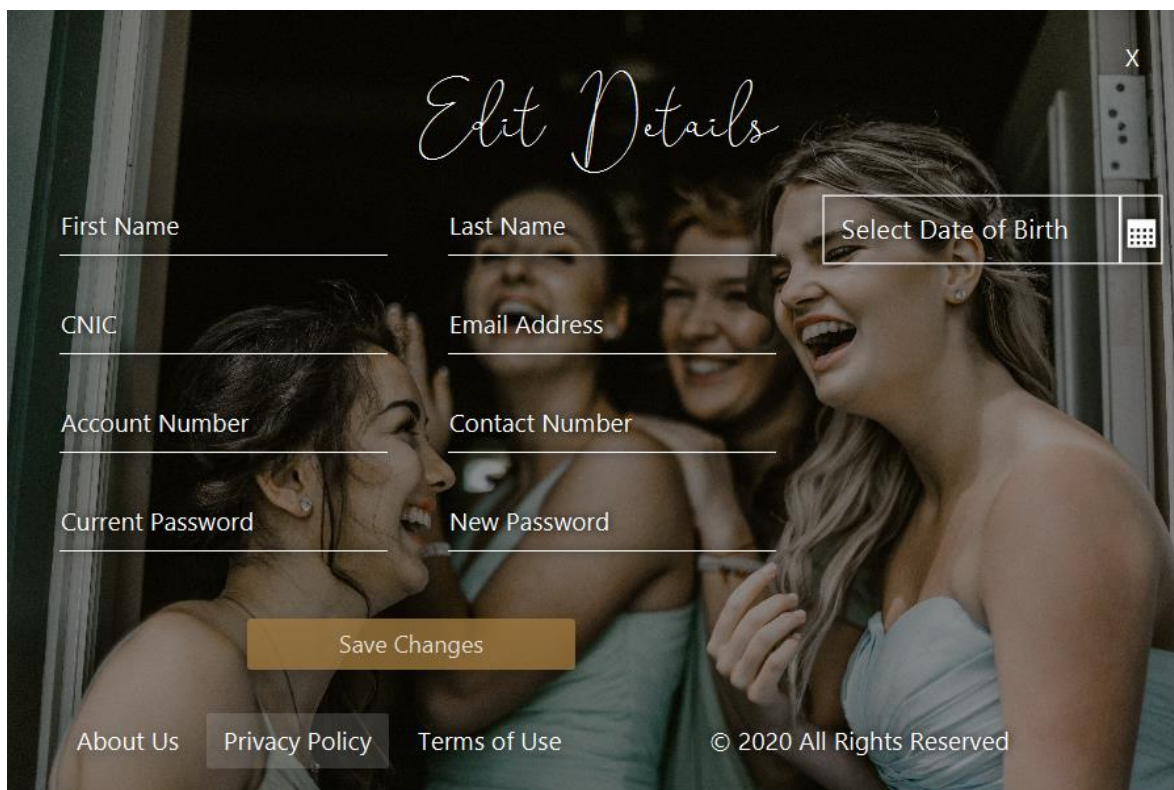


Fig 10. Customer Edit Info Screen

Customer can also view his booked event as follows:

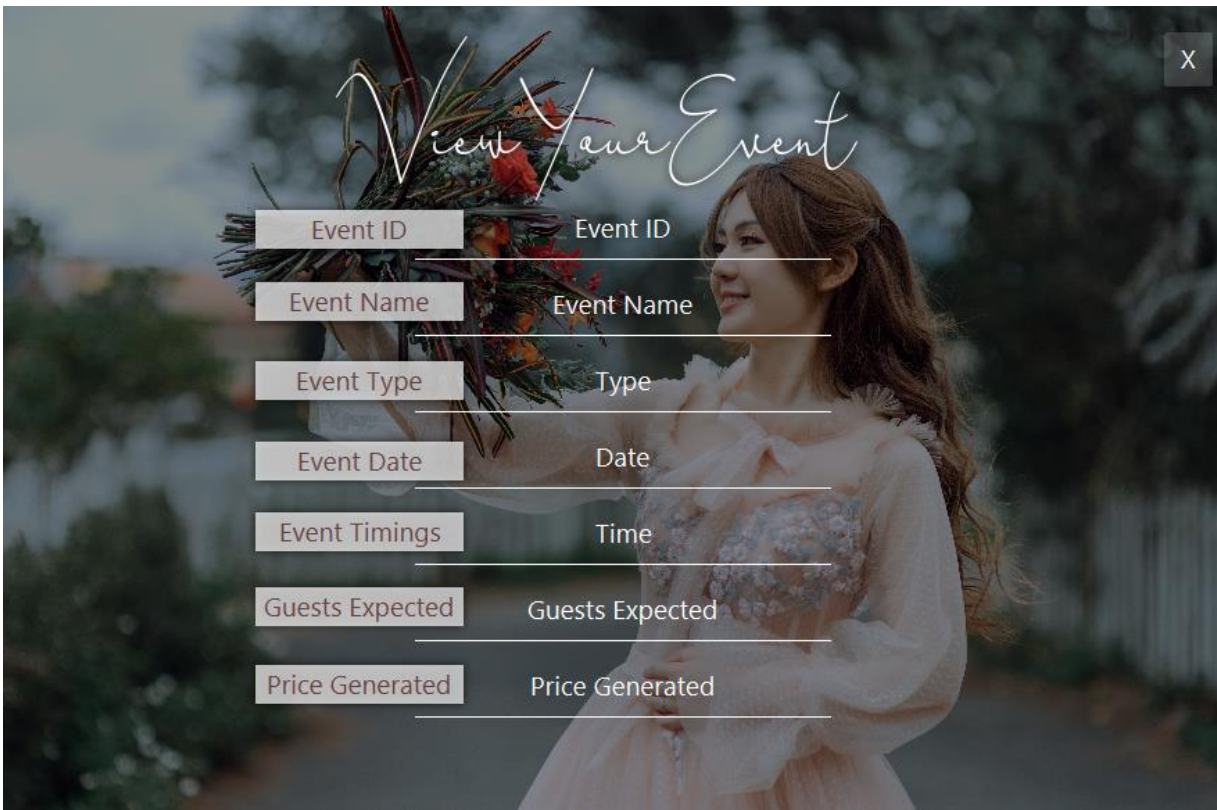


Fig 11. View Current Event Screen

Manager can however perform multiple tasks. The flow of these tasks are as follows:

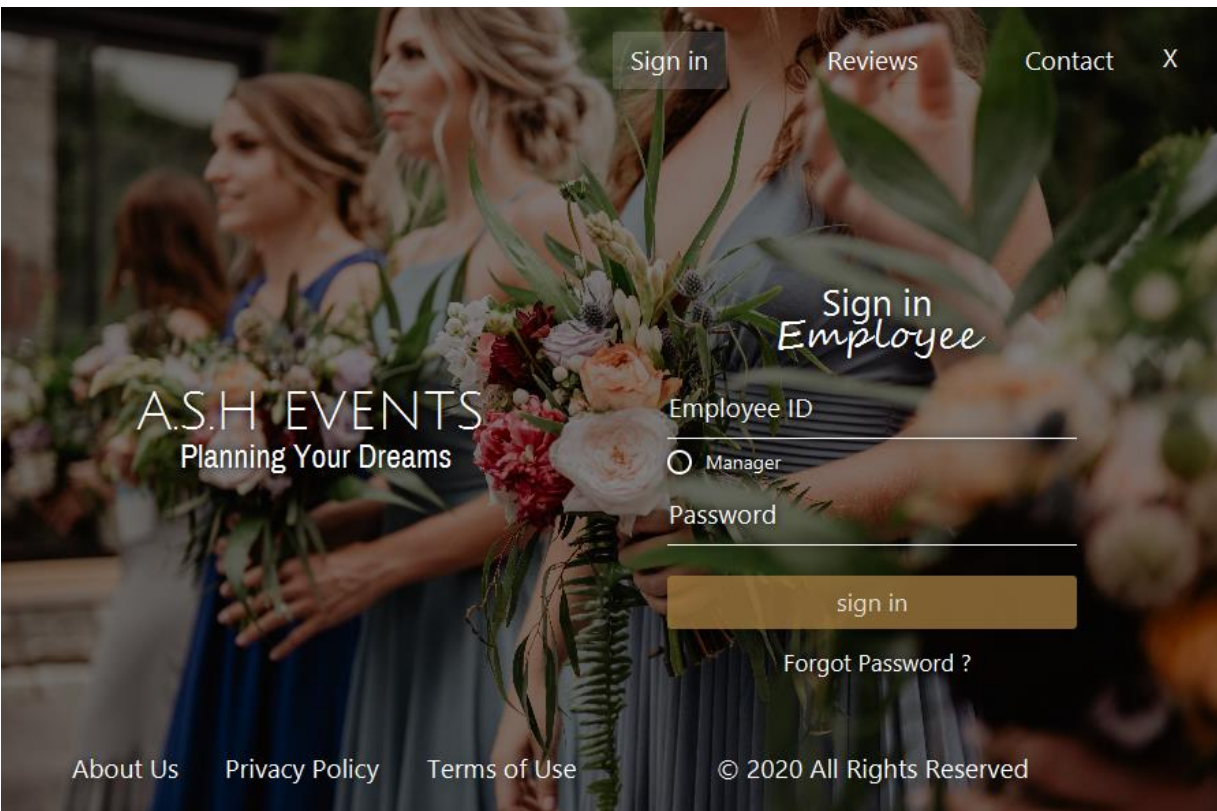


Fig 12. Manager/Employee Sign In Screen

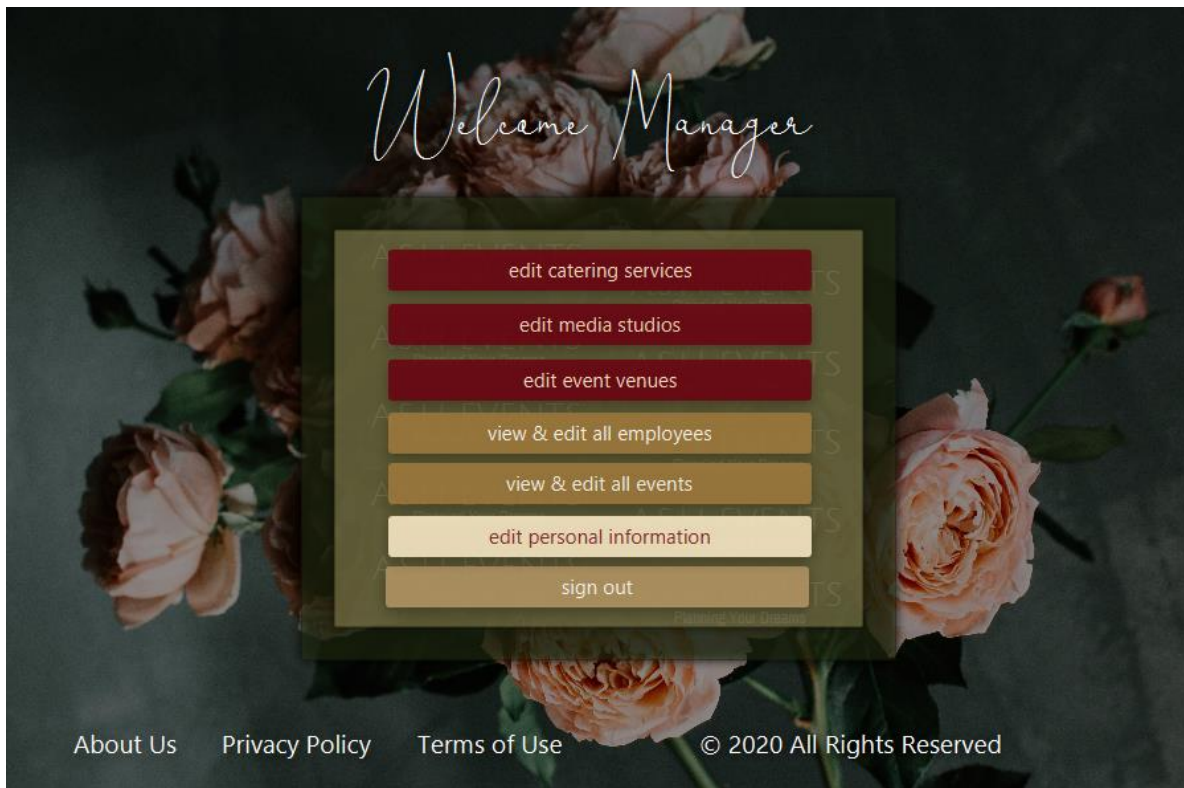


Fig 13. Manager Welcome Screen

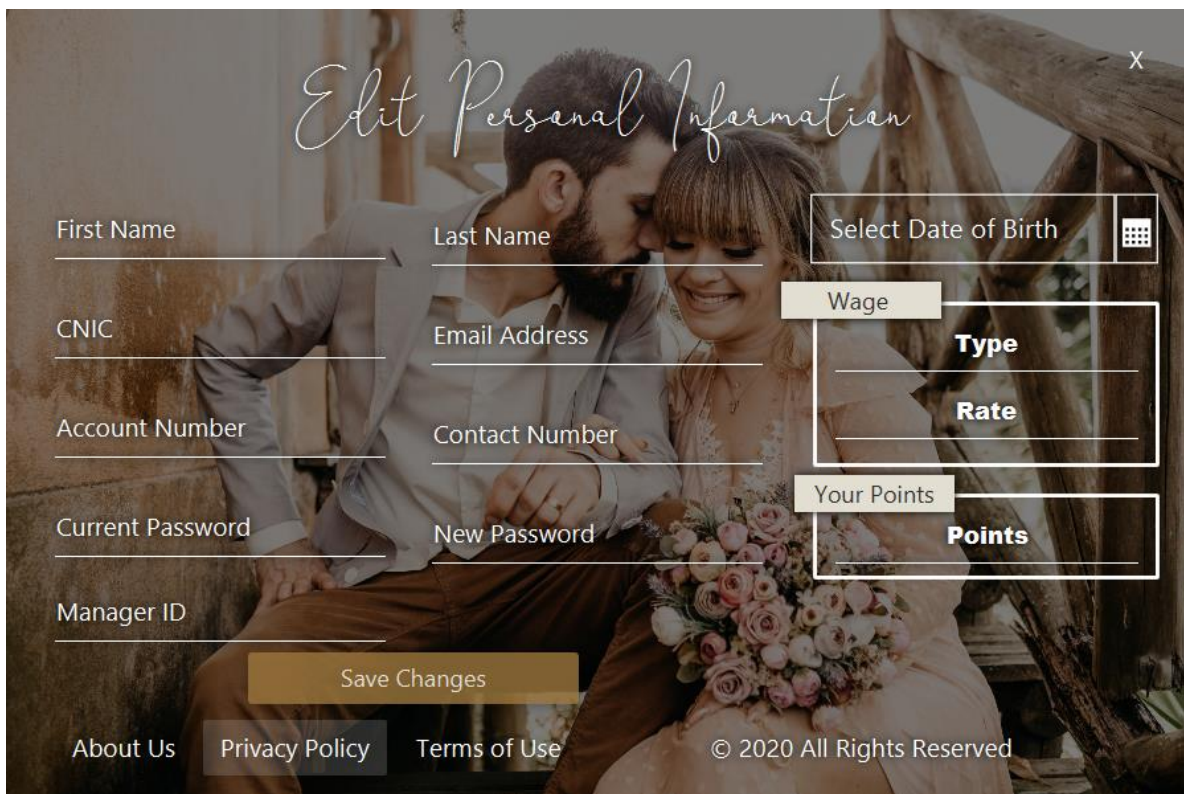
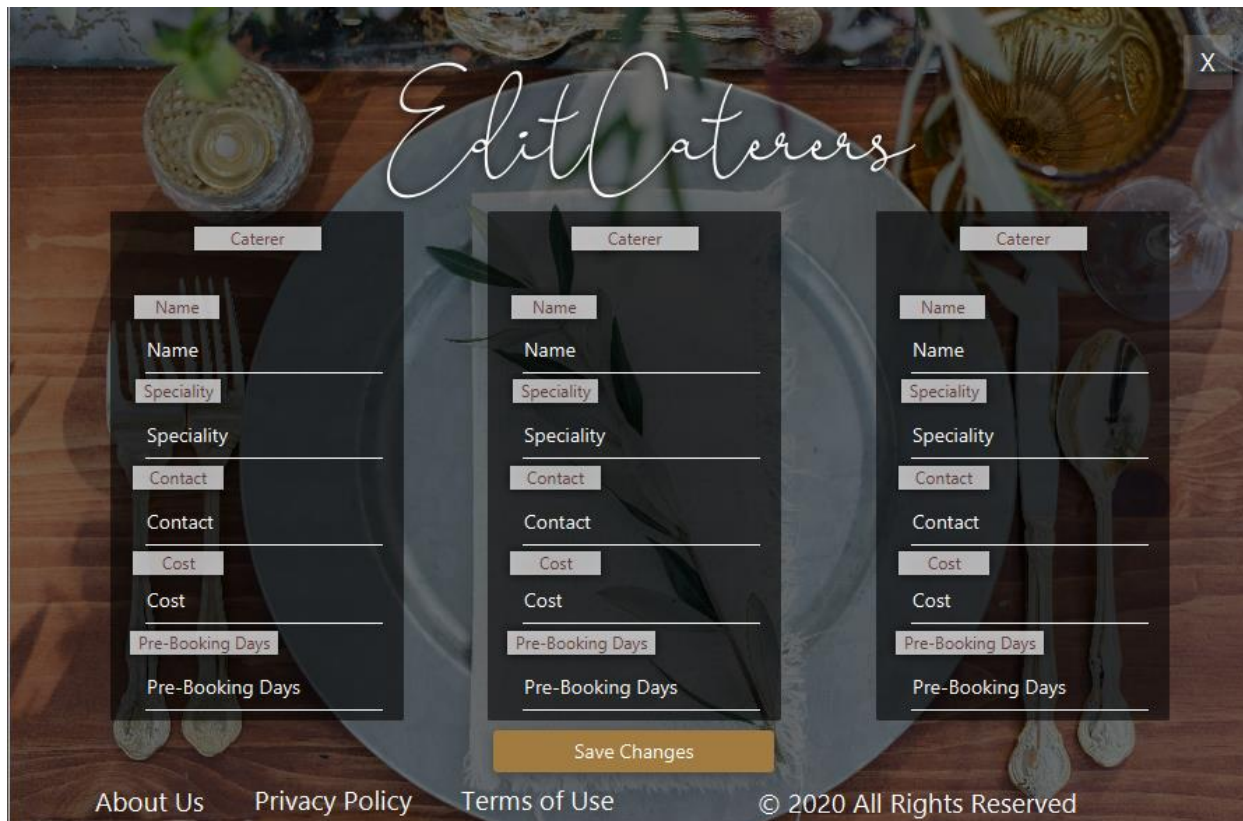
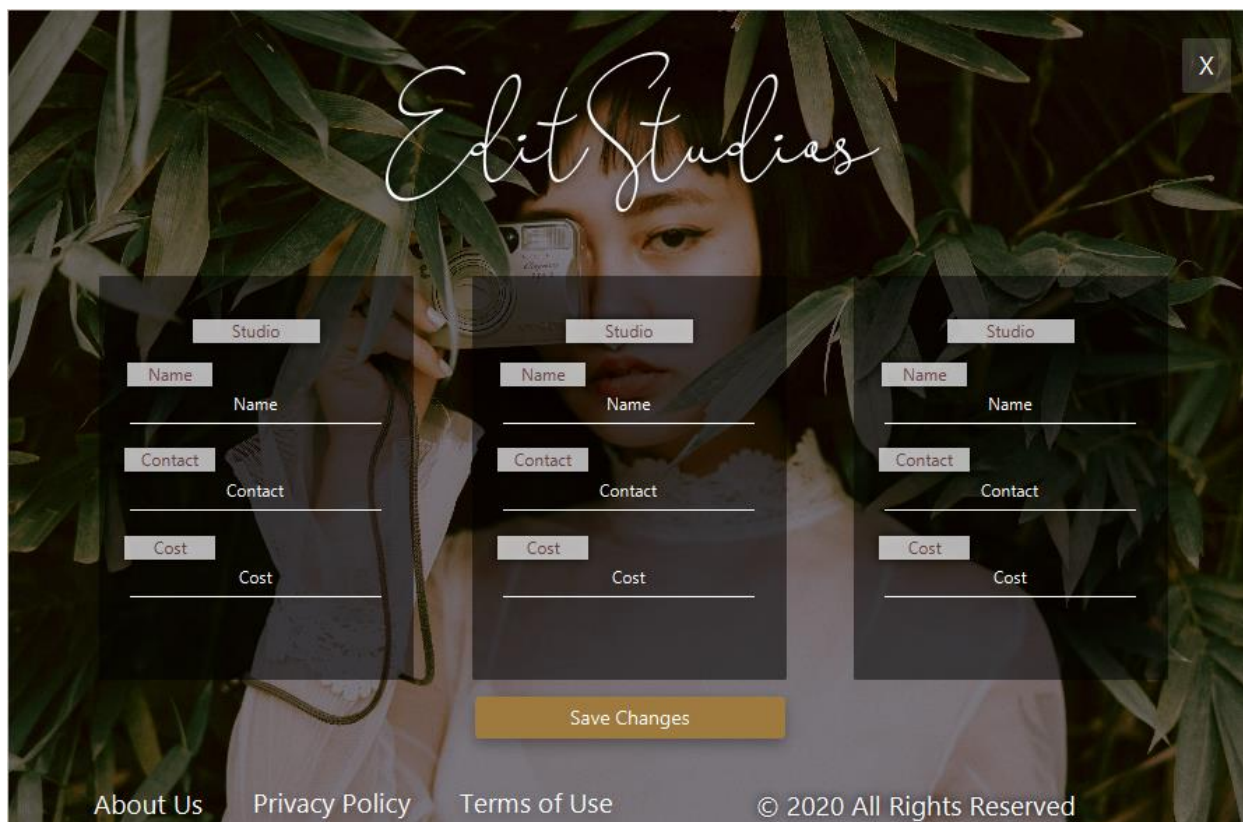


Fig 14. Edit Employee Info Screen



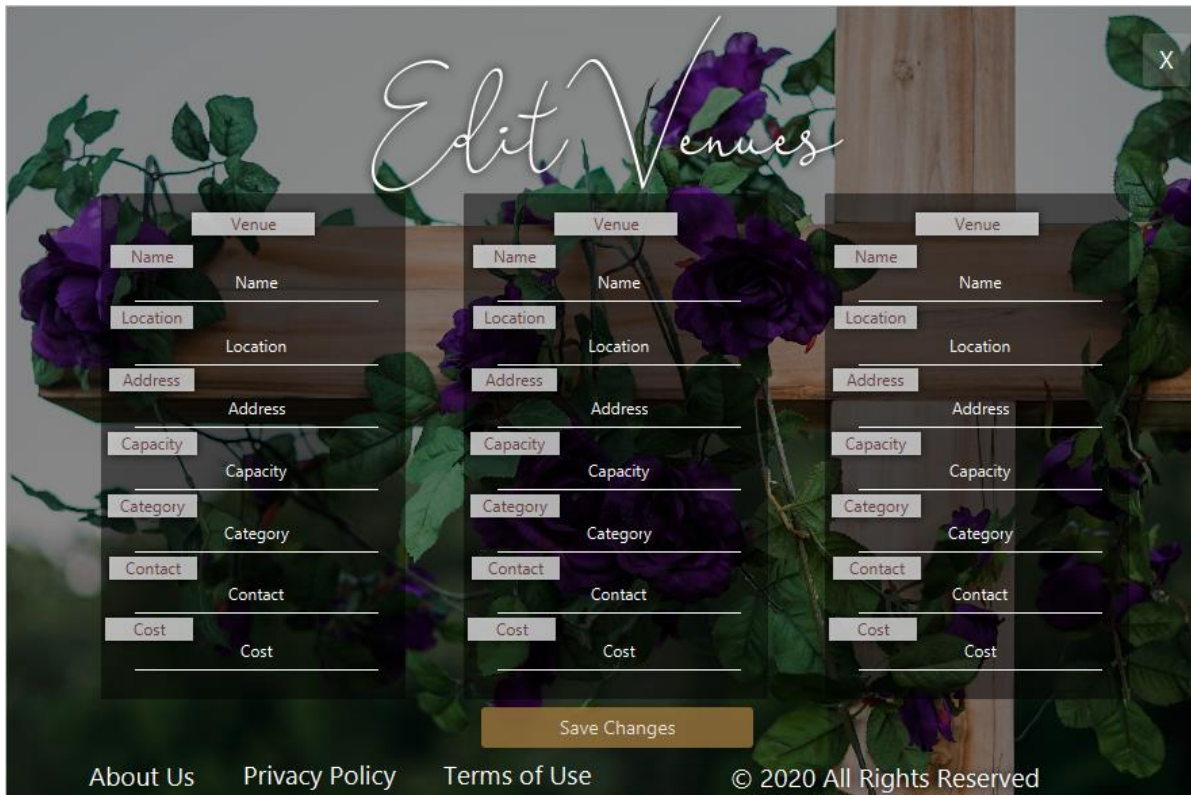
The 'Edit Caterers' screen features a background image of a dining table with glassware and a plate. At the top, the title 'Edit Caterers' is written in a white script font. A close button 'X' is in the top right corner. The screen contains three identical dark grey form panels, each titled 'Caterer'. Each panel has the following fields: 'Name' (with a label above and below the input), 'Speciality' (with a label above the input), 'Contact' (with a label above the input), 'Cost' (with a label above and below the input), and 'Pre-Booking Days' (with a label above the input). A yellow 'Save Changes' button is positioned below the middle panel. At the bottom, there are links for 'About Us', 'Privacy Policy', and 'Terms of Use', followed by the copyright notice '© 2020 All Rights Reserved'.

Fig 15. Edit Caterers Screen



The 'Edit Studios' screen features a background image of a person holding a camera, partially obscured by green foliage. At the top, the title 'Edit Studios' is written in a white script font. A close button 'X' is in the top right corner. The screen contains three identical dark grey form panels, each titled 'Studio'. Each panel has the following fields: 'Name' (with a label above and below the input), 'Contact' (with a label above the input), and 'Cost' (with a label above and below the input). A yellow 'Save Changes' button is positioned below the middle panel. At the bottom, there are links for 'About Us', 'Privacy Policy', and 'Terms of Use', followed by the copyright notice '© 2020 All Rights Reserved'.

Fig 16. Edit Studios Screen



The 'Edit Venues' screen features a dark background with a floral pattern. It contains three identical, semi-transparent white forms arranged horizontally. Each form has a title bar 'Venue' with a close button 'X' on the right. The forms contain the following fields: Name, Location, Address, Capacity, Category, Contact, and Cost. Below the forms is a 'Save Changes' button. At the bottom, there are links for 'About Us', 'Privacy Policy', 'Terms of Use', and a copyright notice '© 2020 All Rights Reserved'.

Fig 17. Edit Venues Screen



The 'View Events' screen has a dark background with a floral pattern. It features a search bar at the top right with the placeholder text 'Emp ID, First Name, Last Name, Wage Rate'. Below the search bar is a table with 8 columns: ID, Name, Type, Date, Guests Expected, Price, and Timings (Start, End). The table contains 11 rows of event data. Row 5 is highlighted in yellow. At the bottom, there is a table with 4 columns: Event ID, View Event, Approve Event, and Delete Event. At the very bottom, there are links for 'About Us', 'Privacy Policy', 'Terms of Use', and a copyright notice '© 2020 All Rights Reserved'.

ID	Name	Type	Date	Guests Expected	Price	Timings
						Start End
20001	Sara's Bridal S...	Bridal Shower	2020-12-24	150	255000	7 pm 11 pm
20002	Ali's Enagement	Enagement	2020-12-31	34	246800	1 pm 4 pm
20003	Sana's Birthday	Birthday	2020-12-25	100	540000	7 pm 11 pm
20004	Saneya's Enag...	Enagement	2020-12-30	90	493000	4 pm 9 pm
20005	Mishal's Bridal...	Bridal Shower	2020-12-26	50	185000	7 pm 11 pm
20006	Ali's Nikkah	Nikkah	2020-12-29	34	231800	12 pm 2 pm
20007	Abeera's Treat	Thanks Giving	2020-12-27	10	507000	7 pm 11 pm
20008	Aimen's Enage...	Enagement	2020-12-28	100	290000	9 pm 12 pm
20009	Saleha's Bridal...	Bridal Shower	2021-01-01	23	229600	3 pm 6 pm
20010	Musa's Gradu...	Graduation	2021-01-06	120	224000	1 pm 4 pm
20011	Haccan's Valima	Wedding	2021-01-02	700	380000	1 pm 5 pm

Fig 18. View All Events Screen

View Employees

Search
Emp ID, First Name, Last Name, Wage Rate

ID	First Name	Last Name	CNIC	DOB	Email	Contact	Wage		Account	Points	Manager ID
							Rate	Type			
20001	Shafia	Manager	1242-532...	2000-03-16	dummy@...	0330-336...	1000	Monthly	934545	800	
20002	Ahmed	Khan	3323-322...	2001-04-12	ahmed@...	0300-675...	2000	Hourly	765434567	600	20001
20003	Ali	Arshad	42201-49...	2002-03-03	doublea...	0300-444...	1000	Monthly	456945543	0	20002
20004	Queen	Elizabeth	Nothing	2004-05-22	hermajest...	Unavailable	100	Daily	65434654	150	20002
20005	Garbage	Man	144-2948...	2001-05-17	IamTrash...	0300-668...	1000	Daily	794-3553	475	20002

Refresh List

First Name

Last Name

CNIC

DOB

Email

Contact

Wage Rate

Wage Type

Account

Manager ID

Add Employee

Emp ID

Edit Employee

Delete Employee

Fig 18. View All Employees Screen

Chapter 17

Database Description

A database is a collection of information that is organized so that it can be easily accessed, managed and updated. Computer databases typically contain aggregations of data records or files, containing information about sales transactions or interactions with specific customers.

For our EMS, we started off with Oracle 11g as 2 of the members had it installed and configured in their systems already. We made all the tables, columns, backend etc.

During the final stages of our project, we wanted it to run on every system, so we started by installing Oracle 11g on 2 remaining systems, however it kept failing. We tried to search online for issue but still no use. So, at this stage, we decided to convert all of our code into a different database. We had 3 databases in mind: MySQL, MariaDB, Firebase. After a bit of research and analysis, we found MySQL to be easiest and the closest to the Oracle 11g so we wouldn't need to change our code entirely. So, we shifted our code to MySQL.

The query language used was SQL.

We are using:

✓ **MySQL 8.0**

✓ **10 Tables**

Following is the details of the tables along with their columns:

Table	Columns
CATERING	<ul style="list-style-type: none"> (catering_id, name, contact, specialty, days, charges)
CUSTOMER	<ul style="list-style-type: none"> (cust_id, name, cnic, age, phone_no, email, account_number, priority_status)
CUSTOMERPASS	<ul style="list-style-type: none"> (cust_id, password) cust_id references cust_id in CUSTOMER
EMPLOYEE	<ul style="list-style-type: none"> (emp_id, name, dob, email, phone_no, cnic, account_number, wage_type, wage_rate, points, mgr_id) mgr_id references emp_id in EMPLOYEE
EMPLOYEEPASS	<ul style="list-style-type: none"> (emp_id, password) emp_id references emp_id in EMPLOYEE
EVENT	<ul style="list-style-type: none"> (event_id, name, type varchar(30), event_date, guests, total_cost, starting_time, ending_time, cust_id, venue_id, studio_id, menu_id, catering_id, media_id, approved) media_id references media_id from MEDIA_REQUIREMENTS studio_id references studio_id from STUDIO catering_id references catering_id from CATERING venue_id references venue_id from VENUE menu_id references menu_id from MENU cust_id references cust_id from CUSTOMER
MEDIA_REQUIREMENTS	<ul style="list-style-type: none"> (media_id, photography, videography, album, drone, crane)
MENU	<ul style="list-style-type: none"> (menu_id, rice, bread, protein, coke, miranda, sprite, water, dryfruit, sugarfree, icecream, cake, cost)
STUDIO	<ul style="list-style-type: none"> (studio_id, name, contact_info, cost)
VENUE	<ul style="list-style-type: none"> (venue_id, name, location, address, max_capacity, description, category, contact_info, cost)

Chapter 18

Implementation Description

This chapter includes the details of the implementation of our backend code.

This includes the following parts:

1. Development Setup:

The setup that we used to implement our software is as follows:

- **Programming Language:**
 - For this we decided to go with **JAVA**, as it is not only cross-platform, but we also got to learn a new programming language.
- **Frontend:**
 - For the frontend, we decided to use **FXML** as it is not only easy to use but also easy to adjust.
- **Hardware Interface:**
 - The interface of our hardware was **Windows 10**.
- **Database:**
 - The database that we used was **MySQL**.
- **Framework:**
 - We used **Java FX** for our front end and as our framework.
- **Tools:**
 - We used **IntelliJ IDEA 2020.3** by JetBrains as it is very responsive and easy to use.
- **Libraries Included:**
 - We used **java sdk 15.0.1**, **Jfoenix 9.0.10** and **MySQL java connector 8.0.22**.

2. Implementation:

For implementation of this software, we started off by following basic OOP concepts. Instead of writing all of the back-end code in a single file, we decided to make multiple files. Each file has a different class having different functionalities. We used controllers to link the front-end to the backend, which is then further linked to the database to ensure smooth flow of information.

We also left 2 types of comments:

- a. A comment at start of each file describing the purpose of the file, its functionalities and the flow of the file/class.
- b. Small comments inside the code for the sole purpose of better understanding, so that this code can be understood and modified by anyone.

We also used inheritance in our classes just for the sake of ease and efficiency. Along with this, the names of variables have also been kept meaningful, so that a reader can simply understand the use of those variables by simply reading the names of the variables.

Moreover, for error handling, we added a small beep which is played whenever an error occurs, followed by a popup message having the description of the error message. This pop up will prompt the user that an error has occurred and will also guide the user regarding the steps to further prevent this error.

The overall implementation of the software was very fluent and efficient keeping in mind all of the efficient programming practices.

3. Error Handling:

In order for a perfect and authentic execution, we implemented numerous error handling conditions in our software. A few of these checks/error handling conditions are as follows:

- ✓ No input fields should be left blank
- ✓ No empty strings should be entered in input fields
- ✓ Only numeric input is accepted for numeric values (*i.e., Account number, guests etc.*)
- ✓ No negative values entered
- ✓ Proper email address format should be followed
- ✓ A customer can book only 1 event at a time
- ✓ Customer cannot view another customer's event
- ✓ Minimum age to register is 18
- ✓ Each email can have a single account
- ✓ Log in fails with incorrect email/password
- ✓ A caterer, venue and a studio must be selected before an event is booked
- ✓ Venue can not be selected if its capacity is less than the number of guests
- ✓ Number of guests must be more than 0
- ✓ Selected date for event must be after the current date
- ✓ A single event per date
- ✓ CNIC cannot have alphabets
- ✓ Selected event id (*for viewing, approving, editing event*) must exist.

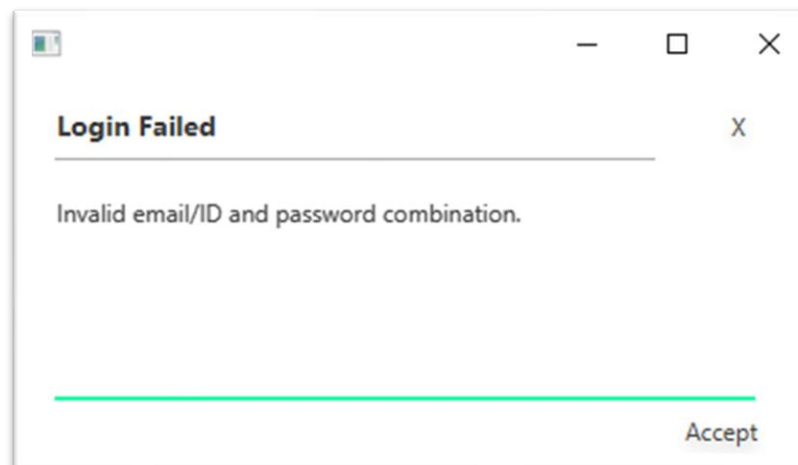
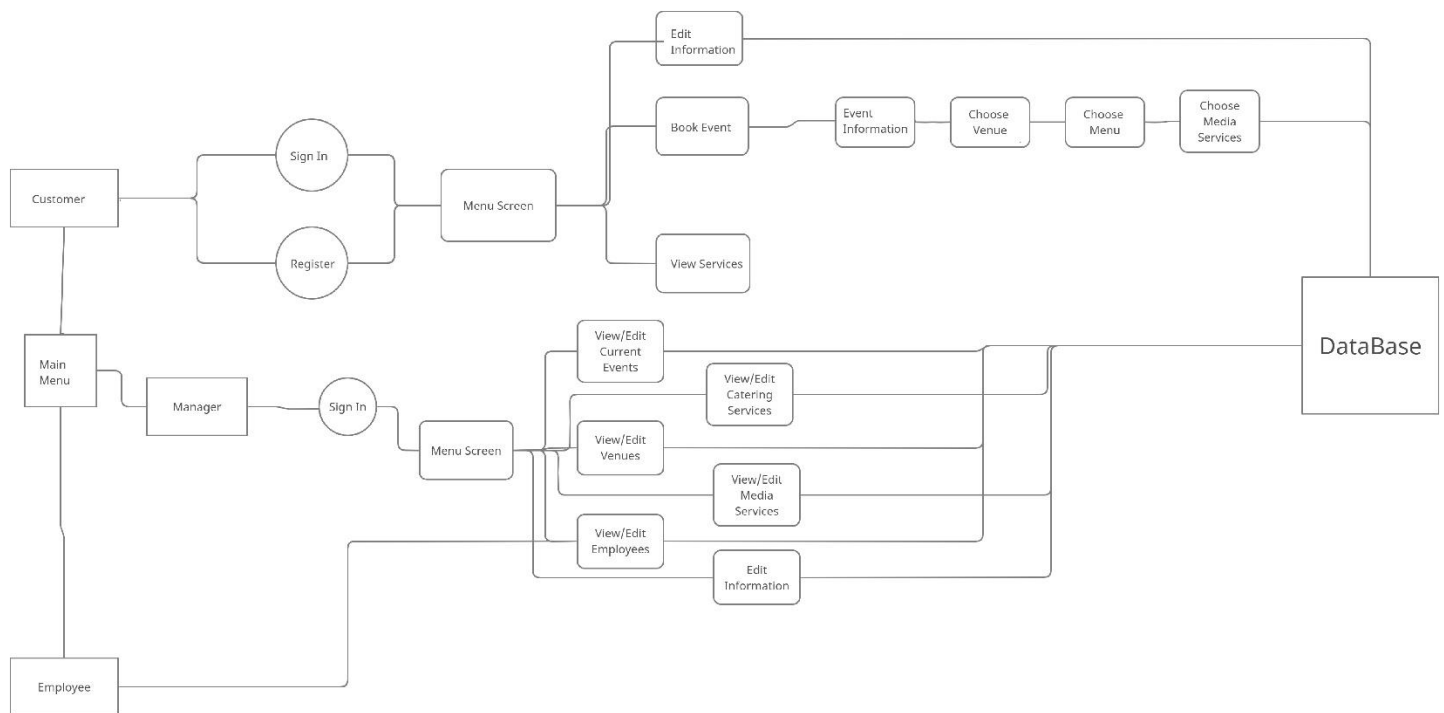


Fig 19. Error Pop-up Screen

Chapter 19

Flow Diagram



Chapter 20

Testing

We performed different types of tests on our software. A few of those tests are as follows:

✓ White Box Testing:

- ✓ In white box testing, a close examination of the logical parts is done to check different conditions, loops and the execution of the program. This enables the developers to understand if there are any logical errors such as a loop iterating more than it should etc. We performed white box testing on our system which proved to be successful.

✓ Black Box Testing:

- ✓ In black box testing, a fixed set of inputs and outputs are used to test the software and the results are compared with the manually calculated results to see if the software is working well. Our software passed this test too.

✓ Alpha Testing:

- ✓ Alpha testing is also known as acceptance testing. In this, a system is designed as for a single user and that user keeps testing until both the user and developer agree that the software runs perfectly for that user. In our case, the user was a fellow friend and the test proved to be successful.

✓ Beta Testing:

- ✓ Before official launch of any product into the market, beta testing is carried out. In this, the product is given to potential people who use the products and point out if any flaws. During this test, we provided our software to 3 different people from different universities and from different domains. One issue was caught which was later on fixed.

✓ Validation Testing:

- ✓ In this testing, we made sure that all the functional and performance requirements are met.

Chapter 21

Results

After performing the above-mentioned tests, we came up with following results:

1. Percentage of Completion:

We have completed our project 100%. We have met all functional requirements which we discussed earlier.

2. Percentage of Accuracy:

Our project is working 100% accurate. It fulfills all the functional and non-functional requirements along with proper error handling.

3. Percentage of Correctness:

As we tested all the requirements using different test cases in the black box testing, and we cleared the error that was brought to light, we can confidently say that our project is 100% correct.

Chapter 22

Conclusion

Our project is only a humble venture to satisfy the needs to manage their project work. Several user-friendly coding techniques have also been adopted. This package shall prove to be a powerful package in satisfying the requirements of both the customer and the firm. The objective of software planning is to provide a framework that enables the manager to make reasonable estimate made within a limited time frame at the beginning of the software project and should be updated regularly.

At the end, we will conclude by saying that we made efforts in following areas:

- ✓ A description of introduction and context of project and its relation to the work already done in this area
- ✓ Project Scope, purpose and stakeholders discussed
- ✓ We defined the project on which we worked
- ✓ We described the requirement specification of the system and actions that can be done on these things
- ✓ We designed user interface and ensured privacy of information.
- ✓ Finally, the system was implemented and tested accordingly.