Requirements Engineering 2

Concert Event Management System

Submitted By: Michal Kubiak (T00205890)

Computing with Software Development

Date Submitted: 11/04/2020

**Table of Contents**

1. Introduction/overview 4

2. Functional Components 5

3. User Requirements 6

3.1. Event Management System 6

3.2. Event Management System will manage Venues 6

3.3. Event Management System will manage Events 6

3.4. Event Management System will manage Bookings 6

3.5. Event Management System will perform Administration 6

4. System Requirements 7

4.1. System-Level Use Case Diagram 8

4.2. Log In 9

4.3. Sign Up 11

4.4. Manage Venues 14

**4.4.1.** **Add Venue** 14

**4.4.2.** **Update Venue** 16

**4.4.3.** **Delete Venue** 19

4.5. Manage Events 21

**4.5.1.** **Add Event** 21

**4.5.2.** **Update Event** 24

**4.5.3.** **Cancel Event** 27

4.6. Manage Bookings 29

**4.6.1.** **Add Booking** 29

**4.6.2.** **Cancel Booking** 32

4.7. Perform Admin 34

**4.7.1.** **Analyse Revenue** 34

**4.7.2.** **Analyse Venue** 36

5. System Model 38

5.1. Level-0 DFD 38

5.2. Level-1 DFD 38

5.3. Level-2 DFD (Process P1: Manage Venues ) 40

5.4. Level-2 DFD (Process P2: Manage Events) 41

5.5. Level-2 DFD (Process P3: Manage Bookings) 42

6. Data Model (Class Diagram) 44

6.1. Class Diagram 44

6.2. Relational Schema 45

6.3. Database Schema 46

7. Conclusion 48

# Introduction/overview

The system described in this document is an Event System for venues and events to be organised and for the sale of tickets to customers.

A manager with access to this system can register new venues, update and delete them. An event can only be added once there are venues in the system and thena n event can be added to a certain venue. The manager can also make bookings for any event for a customer.

Customers can buy a maximum of 10 tickets at once, and they have to be registered with the system as their emailw ill be checked in the booking section.

All events can run once a day per venue, so two events cannot happen at one venue in the same day.

Two of the same events (title wise) can be made but not at the same venue.

The manager can also analyse the revnue of a specific venue for a specific year or the total revenue of all venues ina specific year. Number of events, tickets sold and total revenue.

All dates in the system are set to the YYYY-MM-DD format to compare them and validate that events that are to be added have to be in the future and to get all active events from sysdate onwards, this format is easiest and fastest to compare the string of date.

This system is designed for the Manager and Staff, a separate system would be created for the Custumers where they would only be able to make bookings and check their bookings / cancel their bookings and see the list of currently active upcoming events.

As the bookings that show up in My Bookings function will show up from the CustomerID that is passed when logged in, the system should always be started on log in at least once to set that CustomerID and avoid a null object exception.

# Functional Components

This section presents the functional components of the proposed software system.

When opened the software allows you to either log in or sign up. There are 4 main components in the proposed system each with 2 or 3 sub-components after log in has been proceeded.

# User Requirements

This section describes the user requirements (functional components) as high-level abstract statements.

## Event Management System

## Log In lets you log in to either a user account or the admin account

## Sign Up lets you sign up as another user account

## Event Management System will manage Venues

## Add Venue will add a venue to the Venue file

## Update Venue will update the details of an venue in the Venue file

## Remove Venue will set venue to not available in the Venue file

## Event Management System will manage Events

## Schedule Event will add an event in the Event file

## Update Event will update the details of an event in the Event file

## Cancel Event will set the event cancelled status to true in the Event file

## Event Management System will manage Bookings

## Add Booking will add a booking to the Bookings file

## Cancel Booking will delete a booking from the Bookings file

## Event Management System will perform Administration

## Revenue Analysis will calculate the yearly revenue from all venues

## Venue Analysis will calculate a yearly revenue from one venue

# System Requirements

* Event Management System consists of a log in form, sign up form and four main modules.
* The Log In module provides a function of logging in to the system
* The Sign Up module lets you sign up as a user to be able to make bookings
* The Venues module provides functions to add venues, update venues and remove venues (set to unavailable).
* The Events module provides functions to add events, update events and cancel events (set to unavailable).
* The Bookings module provides a function to add a booking and delete a booking.
* The Admin module provides functions to analyse the yearly revenue of all venues and also any single venue.
* For this project we assume all events run for the entire day.

## System-Level Use Case Diagram

The following system-level use case diagram illustrates the high-level system requirements.

Staff

Customer

## Log In

This module provides a function of either logging in or signing up.

Staff

Customer

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Log In** | |
| **Use Case Id** | 1 | |
| **Priority** | 1 | |
| **Source** | Manager / Customer | |
| **Primary Business Actor** | Manager / Customer | |
| **Other Participating Actors** | N/A | |
| **Description** | This function signs an existing user into the system | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The manager or customer invokes the Log In function (Start of application)  **Step 3:** Manager or customer enters their account data:   * Username (char 25) * Password (char 25)   **Step 4:** The manager or customer selects Log In | **Step 2:** Display UI  **Step 5:** The system validates the data entered:   * All fields must be entered * Username must exist * Password must match the Username’s password in the database   **Step 6:** The system loads a main menu form. If an admin such as a manager logs in, the system displays all of the system functions. If a customer logs in they only have access to the booking function. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Username does not exist** |  | **Step 5:** System cannot find this username in the database  **Step 6:** Display appropriate error message  **Step 7:** Return to step 3 |
| **Password does not match the username’s password in the database** |  | **Step 5:** Password does not match account’s password  **Step 6:** Display appropriate error message  **Step 7:** Return to step 3 |
| **Conclusions** | User is now moved to the Main Menu form. | |
| **Postconditions** | User can now select the modules and functions they would like to use. | |
| **Business Rules** | Managers have unlimited access to system functions, while customers have limited access to the boking function only | |
| **Implementation Constraints** |  | |

## Sign Up

This module allows the customer to sign up and saves their details in the User’s File, it then returns the customer to the log in form.

Customer

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Sign Up** | |
| **Use Case Id** | 2 | |
| **Priority** | 2 | |
| **Source** | Customer | |
| **Primary Business Actor** | Customer | |
| **Other Participating Actors** | N/A | |
| **Description** | This function signs a customer up to the system | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The customer invokes the Sign Up function  **Step 3:** The customer enters the required data:   * Username (char 25) * Password(char 25) * ConfirmPassword(char 25) * Email(char 40) * ConfirmEmail(char 40)   **Step 5:** The customer accepts the terms and agreements  **Step 6:** The customer confirms the data entered | **Step 2:** DisplayUI  **Step 7:** The system validates the data entered:   * All fields must be entered * Username must not already exist * Passwords must match * Email must be valid * Emails must match * Terms and agreements must be accepted   **Step 8:** The system saves the booking details in the users file:   * Username * Password * Email   **Step 8:** The system displays a confirmation message  **Step 9:** The system proceeds to the log in form |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Customer Username already exists in the database** |  | **Step 7:** Username already exists  **Step 8:** Display appropriate error message  **Step 9:** Return to step 3 |
| **Customer Email already exists in the database** |  | **Step 7:** Email already exists  **Step 8:** Display appropriate error message  **Step 9:** Return to step 3 |
| **Field of text missing / not filled in** |  | **Step 7:** Field of text missing  **Step 8:** Display appropriate error message  **Step 9:** Return to step 3 |
| **Conclusions** | The customer is added to the users table in the database | |
| **Postconditions** | The customer can now use the booking modules on the system | |
| **Business Rules** | Usernames and emails must be unique | |
| **Implementation Constraints** |  | |

## Manage Venues

This module provides functions to add venues, update venues and set venues status to unavailable if need be.

### **Add Venue**

This function adds a venue and its details to the venue's file

Manager

<<includes>>

<<extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Add Venue** | |
| **Use Case Id** | 3 | |
| **Priority** | 1 | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | N/A | |
| **Description** | This function adds a venue and its details to the venue's file | |
| **Preconditions** | The manager must be signed into the system | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The manager invokes the Add Venue function  **Step 4:** The manager enters the required data:   * VenueName (char 80) * Street (char 30) * Town (char 30) * County (char 30) * ManagerName (char 50) * ManagerEmail (char 60) * Manager Mobile (char 20) * MaxCapacity (int 5)   **Step 5:** The manager confirms that the venue is to be added | **Step 2:** System loads all Active Venues from the venue’s file in order of VenueID  **Step 3:** Display UI  **Step 6:** The system validates the data entered:   * All fields must be entered * Venue name must not already exist   **Step 6:** The system saves the venue details in the Venues table and assigns a default Status of ‘Active’ to the Venue :   * VenueID (NextID) * VenueName * Street * Town * County * ManagerName * ManagerEmail * ManagerMobile * Status (Active – ‘Y’) * MaxCapacity   **Step 7:** The system displays a confirmation message  **Step 8:** The system resets the UI |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Null Fields** |  | **Step 6:** Fields Empty  **Step 7:** Display an appropriate error message  **Step 8:** Return to step 3 |
|  |  |  |
| **Venue Name already Exists** |  | **Step 6:** Venue Already Exists  **Step 7:** Display “This Venue already Exists”  **Step 8:** Return to step 3 |
| **Conclusions** | The venue has been recorded in the Venues file. | |
| **Postconditions** | Events may now be added for this venue | |
| **Business Rules** | The venue name must not already exist in the database | |
| **Implementation Constraints** |  | |

### **Update Venue**

This function allows the manager to update a specific venue by selecting the Venue ID and changing the venue details. This is then updated in the Venues File.

Manager

<<includes>>

<<extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Update Venue** | |
| **Use Case Id** | 4 | |
| **Priority** | 2 | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | N/A | |
| **Description** | This function updates a venue and its details in the venue's file | |
| **Preconditions** | The manager has logged into the system | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The manager invokes the Update Venue function  **Step 4:** The Manager selects a venue to update from the grid.  **Step 6:** The manager updates the required data:   * VenueName (char 80) * Street (char 30) * Town (char 30) * County (char 30) * ManagerName (char 50) * ManagerEmail (char 60) * Manager Mobile (char 20) * MaxCapacity (int 5)   **Step 7:** The Manager confirms that the venue is to be updated. | **Step 2:** System loads all Active Venues to a grid from the venue’s file in order of VenueID also with an option to change the order to VenueName.  **Step 3:** Display UI  **Step 5:**  System loads the selected venue’s details from the venues file to the text boxes provided.  **Step 8:** The system validates the data entered:   * All fields must be entered * Venue Name must be unique   **Step 9:** The system saves the venue details in the Venues table :   * VenueName * Street * Town * County * ManagerName * ManagerEmail * ManagerMobile * Status (Active – ‘Y’) * MaxCapacity   **Step 9:** The system displays a confirmation message  **Step 10:** The system resets the UI |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Null Fields** |  | **Step 7:** Fields Empty  **Step 8:** Display an appropriate error message  **Step 9:** Return to step 3 |
| **Venue Name already Exists** |  | **Step 7:** Venue Already Exists  **Step 8:** Display “This Venue already Exists”  **Step 9:** Return to step 3 |
| **Conclusions** | The venue details have been overwritten in the database. | |
| **Postconditions** | Events may now continue to be added for this venue | |
| **Business Rules** | Only venues with at ‘Y’ Status can be updated | |
| **Implementation Constraints** |  | |

### **Delete Venue**

This function allows a manager to ‘delete’ a venue from the system, essentially the venue cannot be deleted because of associated records so the system assigns a status of ‘unavailable’ to the selected venue in the venue's file .

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Remove Venue** | |
| **Use Case Id** | 5 | |
| **Priority** | 3 | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | N/A | |
| **Description** | This function allows the manager to ‘delete’ a venue and assigns a status of ‘unavailable’ to the venue. | |
| **Preconditions** | The manager must be logged in to the system. | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The manager invokes the Remove Venue function  **Step 4:** The manager selects a venue to be deleted from the drop down menu.  **Step 6:** The manager confirms that the venue is to be deleted. | **Step 2:** System loads all ‘Active’ Veneus from the venues file.  **Step 3:** Display UI  **Step 5:** The system retrieves and displays the details of the selected venue.  **Step 7:** System sets the status of selected venue to ‘N’ in the venues file  **Step 8:** The system displays a confirmation message  **Step 9:** The system resets the UI |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Manager selects ‘No’ when asked to confirm venue** **cancellation** | **Step 6:** The manager selects ‘No’ when asked to confirm that the venue is to be cancelled. | **Step 7:** Reset UI |
| **Conclusions** | The venue’s availability status is set to ‘N’ in the database. | |
| **Postconditions** | Events cannot be scheduled for this venue. | |
| **Business Rules** | Venue must not be already set to unavailable (false) | |
| **Implementation Constraints** |  | |

## Manage Events

This module provides functions to add events, update events and cancel upcoming events.

### **Add Event**

This function adds an event to the events file

Manager

<<includes>>

<<extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Schedule Event** | |
| **Use Case Id** | 6 | |
| **Priority** | 1 | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | N/A | |
| **Description** | This function adds an event in the events file | |
| **Preconditions** | A venue has been added to the system and a manager has logged on to the system. | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The manager invokes the Schedule Event function  **Step 4:** The manager enters the required data:   * Venue (Combo Box) * EventTitle (Char 50) * Description (Char 140) * EventDate (Date) * EventTime (Char 5) * TicketsAvailable (Int 5) * Price (Float 5)   **Step 5:** The manager confirms that the event is to be scheduled | **Step 2:** System retrieves all ‘active’ venues from the venues file and ‘active’ events from the events file  **Step 3:** Display UI  **Step 6:** The system validates the data entered:   * All fields must be entered * VenueID must not already occur in the event file with the same EventID and EventDate (as all events run all day in one venue) * Number of tickets cannot be more than the capacity of the selected venue   **Step 7:** The system updates the event details in the Events file and sets a default status of ‘Active’:   * VenueID * VenueName * Title * Description * EventDate * EventTime * TicketsAvailable * Price * Status (‘Y’)   **Step 8:** The system displays a confirmation message  **Step 9:** The system resets the UI |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Fields Empty** |  | **Step 6:** Fields empty  **Step 7:**  The system displays appropriate error message  **Step 8:** Return to step 4 |
| **An event already exists with the same date and venueID** |  | **Step 6:** VenueID already exists in another Event with same date  **Step 7:**  Display an error message saying “Cannot hold two events in the same venue on the same day”  **Step 8:** Return to step 4 |
| **TicketsAvailable is more than the Venue’s capacity.** |  | **Step 6:** Capacity exceeded  **Step 7:** Display appropriate error message  **Step 8:** Return to step 4 |
| **Conclusions** | The event has been scheduled. | |
| **Postconditions** | Bookings can now be made for this event | |
| **Business Rules** | VenueID must not already occur in the event file with the same EventID and EventDate (as all events run all day in one venue) | |
| **Implementation Constraints** |  | |

### **Update Event**

This function allows a manager to update an existing event in the events file

Manager

<<includes>>

<<extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Update Event** | |
| **Use Case Id** | 7 | |
| **Priority** | 2 | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | N/A | |
| **Description** | This function updates an existing event in the events file | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The manager invokes the Update Event function  **Step 4:** The manager enters the required data:   * Venue (Combo Box) * EventTitle (Char 50) * Description (Char 140) * EventDate (Date) * EventTime (Char 5) * TicketsAvailable (Int 5) * Price (Float 5)   **Step 5:** The manager confirms that the event is to be scheduled | **Step 2:** System retrieves a list of ‘active’ venues from the venues file, and a list of ‘active’ events from the events file  **Step 3:** Display UI  **Step 6:** The system validates the data entered:   * All fields must be entered * VenueID must not already occur in the event file with the same EventID and EventDate (as all events run all day in one venue) * Number of tickets cannot be more than the capacity of the selected venue   **Step 7:** The system saves the event details in the Events table:   * VenueID (From combo selection) * VenueName (From combo selection) * Title * Description * EventDate * EventTime * TicketsAvailable * Price * Status (Active ‘Y’)   **Step 8:** The system displays a confirmation message  **Step 9:** The system resets the UI |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Fields Empty** |  | **Step 6:** Fields empty  **Step 7:**  The system displays appropriate error message  **Step 8:** Return to step 4 |
| **An event already exists with the same date and venueID** |  | **Step 6:** VenueID already exists in another Event with same date  **Step 7:**  Display an error message saying “Cannot hold two events in the same venue on the same day”  **Step 8:** Return to step 4 |
| **TicketsAvailable is more than the Venue’s capacity.** |  | **Step 6:** Capacity exceeded  **Step 7:** Display appropriate error message  **Step 8:** Return to step 4 |
| **Conclusions** | The event details have been updated. | |
| **Postconditions** | Events with this ID may now be booked again | |
| **Business Rules** | Events must not exist with the same date and the same Venue ID, Any bookings made for this event must be notified of any changes | |

### **Cancel Event**

This function allows the manager to ‘cancel’ an upcoming event the system sets the canceled status to true in the events file

Manager

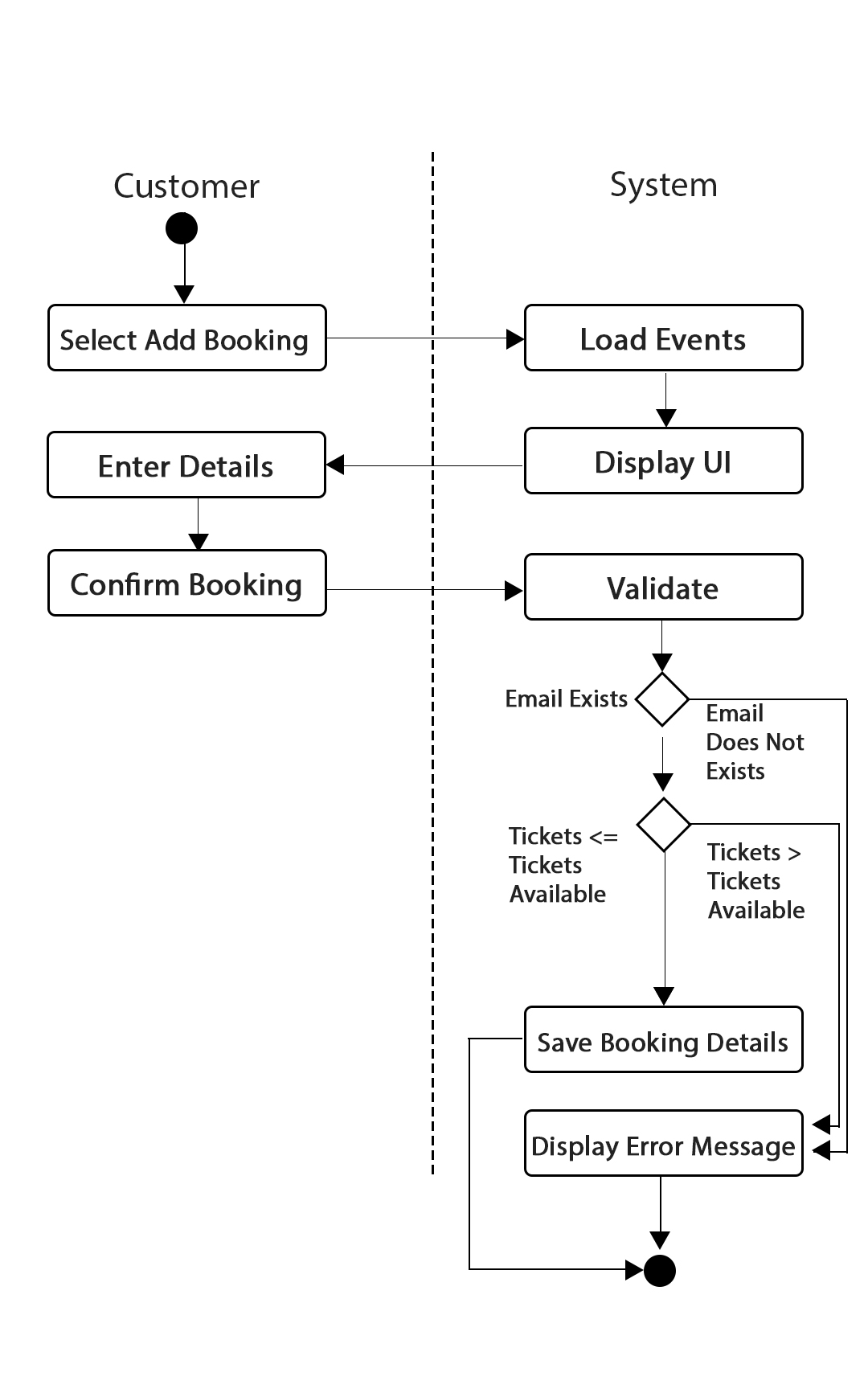
|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Cancel Event** | |
| **Use Case Id** | 8 | |
| **Priority** | 3 | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | N/A | |
| **Description** | This function allows a manager to ‘cancel’ an event and the system sets the status of an event to canceled (true) in the events file | |
| **Preconditions** | A manager has logged on to the system | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The manager invokes the CancelEvent function  **Step 4:** The manager selects an event to be deleted from combo box  **Step 6:** The manager confirms that the event is to be canceled. | **Step 2:** System retrieves ‘Active’ Events from the events file. (Active events cannot be in the past, as all events with the Date < SysDate have their status set to ‘N’ (inactive))  **Step 3:** Display UI  **Step 5:** The system retrieves and displays all information for the selected event  **Step 7:** System sets:   * the status of selected event to ‘N’ in the events file (Canceled)   **Step 8:** The system displays a confirmation message  **Step 9:** The system resets the UI |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Manager selects ‘No’ when asked to confirm event cancellation** | **Step 6:** The manager selects ‘No’ when asked to confirm that the event is to be cancelled. | **Step 7:** Reset UI |
| **Conclusions** | The events status has been changed to canceled (‘N’) in the events file. | |
| **Postconditions** | Bookings cannot be made for this event from now – All bookings made so far must have their status changed to refunded | |
| **Business Rules** | Bookings cannot be permanently deleted after an event is canceled in order to track the revenue precicely | |
| **Implementation Constraints** |  | |

## Manage Bookings

The Bookings module provides a function to add a booking, update a booking and delete a booking.

### **Add Booking**

This function allows a customer or a manager to make a booking in the bookings file.



Customer

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Add Booking** | |
| **Use Case Id** | 9 | |
| **Priority** | 1 | |
| **Source** | Manager / Customer | |
| **Primary Business Actor** | Manager / Customer | |
| **Other Participating Actors** | N/A | |
| **Description** | This function allows a customer or manager to make a booking in the bookings file | |
| **Preconditions** | A manager or customer must be logged into the system | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** A manager or customer invokes the Add Booking function  **Step 4:** The manager or customer selects an event  **Step 5:** If manager is signed in with admin, the manager enters the customers email to make a bookings and selects no. of tickets. (If customer is signed in they just select the amount of tickets they wish to buy)   * Email (char 30) * No Tickets (Combo)   **Step 6:** The manager or customer confirms that the venue is to be added | **Step 2:** System loads all ‘active’ events from the events file (upcoming events with status ‘Y’)  **Step3:** Display UI  **Step 7:** The system validates the data entered:   * No of tickets must be less than tickets available for this event   **Step 8:** The system saves the booking details in the booking table:   * BookingID (ID generated on load by NextID method) * CustID * EventID * BookingDate * NoTickets * BookingTotal   **Step 9:** The system displays a confirmation message  **Step 10:** The system resets the UI |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Cust / Manager selects no. of tickets** |  | **Step 7:** Tickets selected is more than available tickets for event  **Step 8:** Display appropriate error message  **Step 9:** Return to step 4 |
| **Conclusions** | The booking has been recorded in bookings file. | |
| **Postconditions** | This booking can be used to attend and event or it can be deleted, revenue from this booking is added into the Bookings file for revenue analysis | |
| **Business Rules** | Email must be already registered | |
| **Implementation Constraints** |  | |

### **Cancel Booking**

This function allows a manager or customer to cancel an upcoming booking and the system sets a status of cancelled to an event in the events file

Manager

Customer

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Cancel Booking** | |
| **Use Case Id** | 11 | |
| **Priority** | 3 | |
| **Source** | Manager/Customer | |
| **Primary Business Actor** | Manager/Customer | |
| **Other Participating Actors** | N/A | |
| **Description** | This function allows a customer to cancel a booking for an upcoming event | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The manager or customer invokes the My Bookings / Search Bookings function  **Step 3:** If signed in as an admin, the manager enters a customer email to view/cancel the customer’s bookings. If signed in as a customer it displays their active (upcoming) bookings.  **Step 5:** Customer or Manager selects a booking  **Step 6:** Customer or Manager confirms cancelation | **Step 2:** Display UI  **Step 4:** The System loads bookings with a status of ‘Y’ (Active) that are associated with the provided customer email  **Step 7:** System updates the booking with the status ‘N’  **Step 8:** System resets UI |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Email not found** |  | **Step 4:** Email not found in Users File  **Step 5:** Display Error Message  **Step 6:** Return |
| **Conclusions** | The particular booking has been canceled and is awaiting to be refunded | |
| **Postconditions** | This booking will no longer be visible in My Bookings/Bookings Search | |
| **Business Rules** | Bookings cannot be deleted as the revenue function needs to know about which ones were refunded in order for the analysis to be accurate, therefore the status is changed instead | |
| **Implementation Constraints** |  | |

## Perform Admin

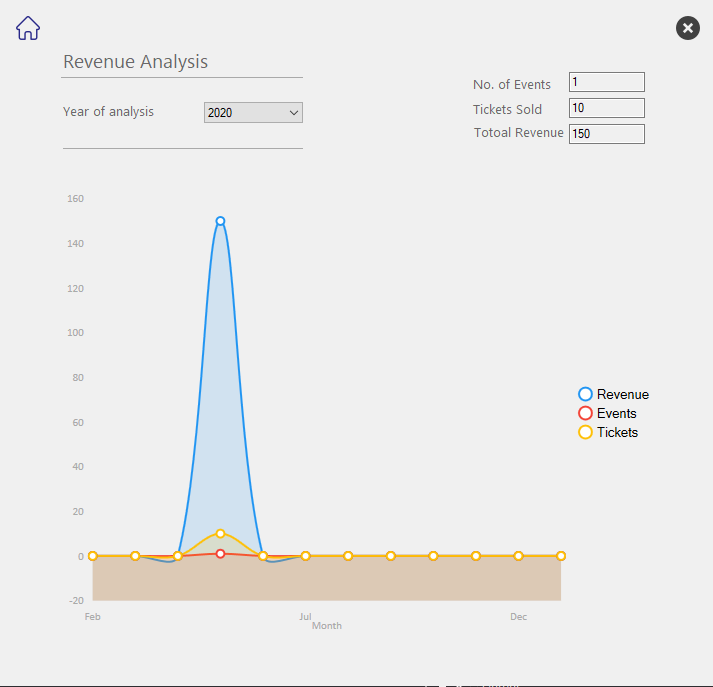
This module will allow the manager and only the manager to log in to the admin panel which consists of two functions, Analyse Revenue and Analyse Venue.

### **Analyse Revenue**

This function allows the administrator to check the revenue of all events in all venues over a period of a year.

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Analyse Revenue** | |
| **Use Case Id** | 12 | |
| **Priority** | 12 | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | N/A | |
| **Description** | This function would allow the admin to check the revenue of all events in all venues over a period of a year. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step1:** The manager invokes the Analyse Revenue function  **Step 3:** Manager selects the year to be analysed from the list (This displays all the revenue for every month for the selected year) | **Step 2:** Display UI  **Step 4:** System loads the amount of events for all venues that year, the number of tickets sold and the total revenue from the Events, Venues and Bookings file. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
|  |  |  |
| **Conclusions** | Revenue for the year is calculated and displayed | |
| **Postconditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

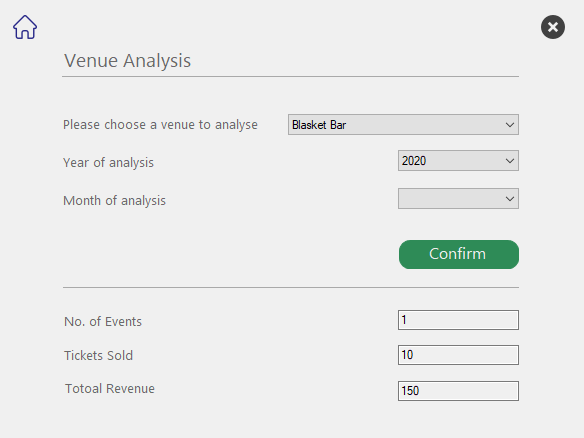


### **Analyse Venue**

This function allows the manager to check the revenue of all events in single venue over a period of a year.

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Analyse Venue** | |
| **Use Case Id** | 13 | |
| **Priority** | 13 | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | N/A | |
| **Description** | This function would allows the manager (admin) to check the revenue of all events in a single venue over a period of a year. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step1:** The manager invokes the Analyse Venue function  **Step 3:** Manager selects the venue to be analysed and selects the year to analyse. They may also specify the month to analyse if required  **Step 4:** Manager clicks confirm button | **Step 2:** Display UI  **Step 5:** System loads the amount of events for this venue that year, and the number of tickets sold and sums the total revenue from the Events, Veneus and Bookigns file. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Venue or Year not selected** |  | **Step 5:** Venue or Year not selected  **Step 6:** Display appropriate message  **Step 7:** Reset UI |
| **No of tickets, events or revenue found or null** |  | **Step 5**: No information available for selected year/month  **Step 6**: Display “No results found” or 0 |
| **Conclusions** | Revenue, tickets and number of events for a particular venue have been displayed. Analysis has been performed. | |
| **Postconditions** |  | |
| **Business Rules** | Venue and Year must be selected | |
| **Implementation Constraints** |  | |



# System Model

The following dataflow diagrams have been produced for the system:

## Level-0 DFD

BookingID

CUSTOMER

Booking Details

Event

Management

System

## Level-1 DFD

Venue Details

P4

PERFORM

ADMIN

Venue Details

P1

MANAGE

VENUES

D1 VENUES FILE

Event Details

D2 EVENTS FILE

Venue Details

Event Details

P5

LOG IN /

SIGN UP

P2

MANAGE

EVENTS

Customer Details

D3 CUSTOMERS FILE

Event Details

CUSTOMER

Customer Details

P3

MANAGE

EVENTS

Booking Details

Booking Details

D4 BOOKING FILE

## Level-2 DFD (Process P1: Manage Venues )

Venue Details

New Venue Details

P1.3

CANCEL

VENUE

P1.1

ADD

VENUE

D1 VENUES FILE

Venue Status

Venue Details

P1.2

UPDATE

VENUE

Updated Venue Details

## Level-2 DFD (Process P2: Manage Events)

D1 VENUES FILE

Venue Details

Updated Event Details

Event Details

Event Status

Event Details

D2 EVENT FILE

P2.1

ADD

EVENT

P2.3

CANCEL

EVENT

Event Details

P2.2

UPDATE

EVENT

Venue Details

D1 VENUES FILE

## Level-2 DFD (Process P3: Manage Bookings)

**Level-2 DFD (**Process P4: Perform Admin**)**

P3.2

CANCEL

BOOKING

P3.1

ADD

BOOKING

Booking Details

D4 BOOKINGS FILE

Event Details

D2 EVENTS FILE

Booking Status

Booking ID

Booking Details

Booking Details

CUSTOMER

P4.1

ANALYSE

VENUE

Venue Details

Venue Details

D1 VENUES FILE

Event Details

Event Details

D2 EVENTS FILE

Booking Details

Booking Details

D4 BOOKINGS FILE

P4.2

ANALYSE

REVENUE

# Data Model (Class Diagram)

## Class Diagram

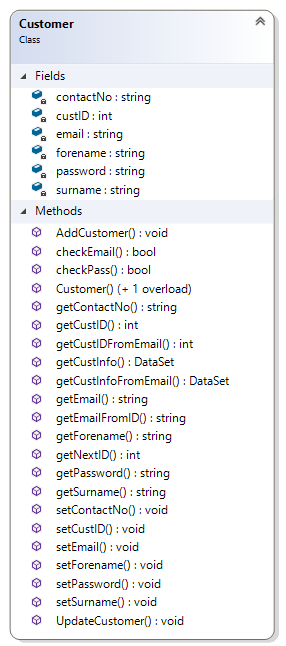
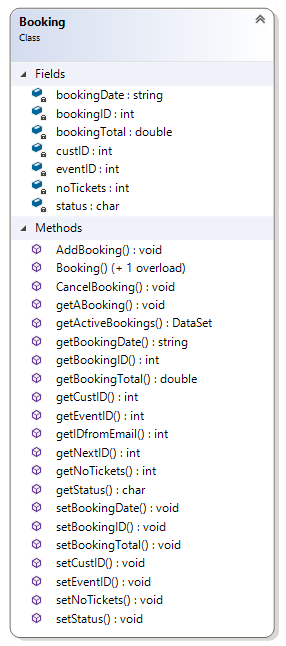
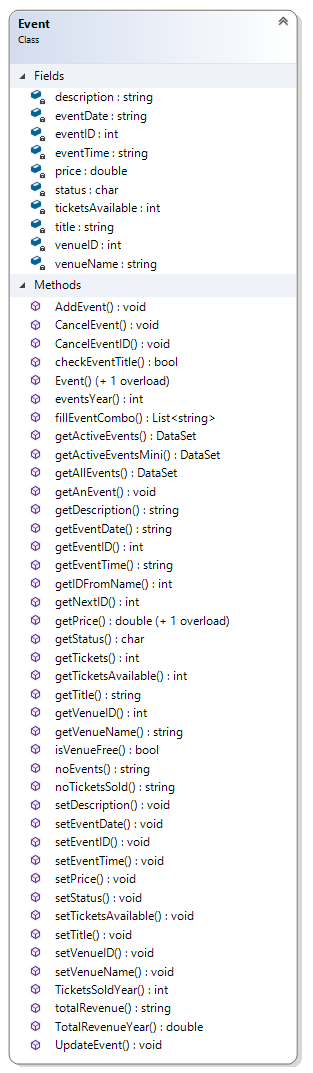
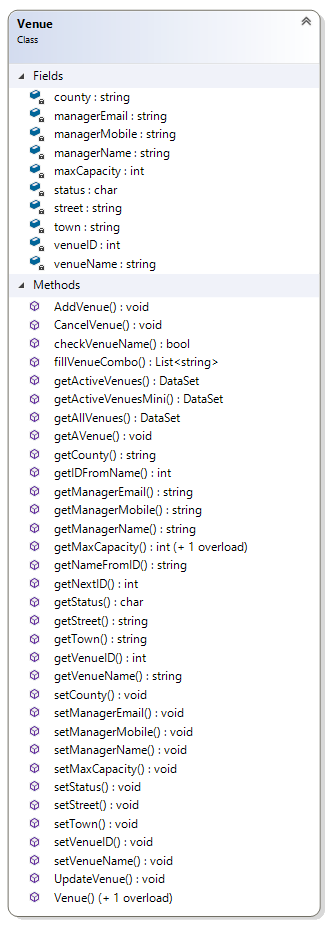
Object Model – UML Class Diagram

Class diagram shows objects & attributes

(Created using class diagram options in Visual Studio)

0..\*

adds an



1

1..\*

is booked

0..1

Makes a

1..\*

0..1

## Relational Schema

The relational schema for the data requirements - Using ***bracket notation***

***Venues****(VenueID, VenueName, Street, Town, County, ManagerName, ManagerEmail, ManagerMobile, Status, MaxCapacity)*

***Events****(EventID, VenueID, VenueName, Title, Description, EventDate, EventTime, TicketsAvailable, Price, Status)*

***Customer****(CustID, Forename, Surname, Email, Password, ContactNo)*

***Bookings****(BookingID, CustID, EventID, BookingDate, NoTickets, BookingTotal, Status)*

## Database Schema

A definition of the database to be implemented.

This includes primary key, foreign key and other constraints to be implemented.

**Schema: EventSYS**

**Relation: Venues**

**Attributes:**

VenueID numeric(4) NOT NULL ,

VenueName varchar2(80) NOT NULL,

Street varchar2(30) NOT NULL,

Town varchar2(30) NOT NULL,

County varchar2(30) NOT NULL,

ManagerName varchar2(50) NOT NULL,

ManagerEmail varchar2(60) NOT NULL,

ManagerMobile varchar2(20) NOT NULL,

Status char(1) DEFAULT 'Y',

MaxCapacity numeric(5) NOT NULL

Primary Key VenueID

**Relation: Events**

**Attributes:**

EventID numeric(4) NOT NULL,

VenueID numeric(4) NOT NULL,

VenueName varchar2(80) NOT NULL,

Title varchar2(50) NOT NULL,

Description varchar2(140),

EventDate varchar2(30) NOT NULL,

EventTime varchar2(20) NOT NULL,

TicketsAvailable numeric (5) NOT NULL,

Price numeric(5,2) NOT NULL,

Status char(1) DEFAULT 'Y' NOT NULL,

Primary Key EventID

Foreign Key VenueID References Venues

**Relation: Customers**

**Attributes:**

CustID numeric(5) NOT NULL,

forename varchar2(20) NOT NULL,

surname varchar2(20) NOT NULL,

Email varchar2(40) NOT NULL UNIQUE,

Password varchar2(8)NOT NULL,

ContactNo varchar2(14) NOT NULL,

Primary Key CustID

**Relation: Bookings**

**Attributes:**

BookingID numeric(8) NOT NULL,

CustID numeric(5) NOT NULL,

EventID numeric(4) NOT NULL,

BookingDate varchar(10) NOT NULL,

NoTickets numeric(2) NOT NULL,

BookingTotal numeric(6,2) NOT NULL,

Status char(1) Default 'Y' NOT NULL,

Primary Key BookingID

Foreign Key CustID References Customers

Foreign Key EventID References Events

# Conclusion

While designing the prototype of my Event Management System I learned a lot of new skills when it comes to object oriented programming and the structure of a basic application hat uses a database. My design skills came in handy as I tried my best to make the application look somewhat *modern.* I planned most of the functionality and drew blueprints of individual components and forms, but as the project went on a lot fo the methods and functionality was addictionally needed for the system to work propeprly.

I can say I am happy with how the end product turned out and I think most of the validation is in place to prevent most of the causes of a potential crash. Business rules are met and the referential integrity is also in place.

As to some other improvements I would implement in a real release are more design tweaks making it more user friendly and developing a Flutter or a React app based on this idea.

This project is best suited for local business that would like to market and advertise small events and even big venues trying to host a concert. I would like to thinkt he ease of use would be up to a certain level and standard where even non tech-savvy individuals would have no problem navigating and operating this sysem.