Software Engineering

Event Management System

Submitted By: Stephanie Collins (T00208279)

Computing with Software Development

Date Submitted: 14/04/2020

Table of contents

[1. Introduction/overview 3](#_Toc38130512)

[2. Functional Components 4](#_Toc38130513)

[3. User Requirements 5](#_Toc38130514)

[3.1. EventSYS will manage Venues 5](#_Toc38130515)

[3.2. EventSYS will manage Events 5](#_Toc38130516)

[3.3. EventSYS will manage Bookings 5](#_Toc38130517)

[3.4. EventSYS will perform administrative reporting 5](#_Toc38130518)

[4. System Requirements 6](#_Toc38130519)

[4.1. System Level Use Case Diagram 6](#_Toc38130520)

[4.2. Manage Venues 7](#_Toc38130521)

[4.2.1. Register Venue 7](#_Toc38130522)

[4.2.2. Update Venue 10](#_Toc38130523)

[4.2.3. Delete Venue 13](#_Toc38130524)

[4.3. Manage Events 15](#_Toc38130525)

[4.3.1. Add Event 15](#_Toc38130526)

[4.3.2. Update Event 18](#_Toc38130527)

[4.3.3. Cancel Event 21](#_Toc38130528)

[4.4. Manage Bookings 23](#_Toc38130529)

[4.4.1. Sign Up Customer 23](#_Toc38130530)

[4.4.2. Top Up Account 25](#_Toc38130531)

[4.4.3. Make Booking 28](#_Toc38130532)

[4.4.4. Cancel Booking 32](#_Toc38130533)

[4.1. Perform Administration 34](#_Toc38130534)

[4.1.1. Analyse Venues 34](#_Toc38130535)

[4.1.2. Analyse Revenue 36](#_Toc38130536)

[5. System Model 38](#_Toc38130537)

[5.1. Level-0 DFD 38](#_Toc38130538)

[5.2. Level-1 DFD 39](#_Toc38130539)

[5.3. Level-2 DFD (Process P1: Manage Venues) 40](#_Toc38130540)

[5.4. Level-2 DFD (Process P2: Manage Events) 41](#_Toc38130541)

[5.5. Level-2 DFD (Process P3: Manage Bookings) 42](#_Toc38130542)

[5.6. Level-2 DFD (Process P3: Perform Administration) 43](#_Toc38130543)

[6. Data Model (Class Diagram) 44](#_Toc38130544)

[6.1. Class Diagram 44](#_Toc38130545)

[6.2. Relational Schema 45](#_Toc38130546)

[6.3. Database Schema 46](#_Toc38130547)

[7. Conclusion 48](#_Toc38130548)

# Introduction/overview

**Event System**

The system outlined in this document is an Event Management System for local venues and organisations to advertise upcoming events and sell tickets to customers.

A venue manager can register their venue on the system, update it and delete it if needs be. Once a venue has been added to the system, the managers will be able to register upcoming events taking place at their venue and sell tickets to customers through the system.

The venues will be charged a usage fee of 5% of the venue’s yearly sales. This allows for smaller non-profits, community resource centres, pubs and clubs to also use the system as an advertising mechanism free of charge when they are hosting free events e.g. Live music, family fun days etc.

This system will be an easily accessed go-to for customers to find an array of different types of upcoming events in their area. Customers can sign up to book ticketed events and top up their account when needed. Customers can book up to 4 tickets maximum at a time and will be charged a €2.50 booking fee on each booking.

The system will also allow venue managers to analyse the venue in terms of the revenue earned through the use of the system, their usage fee for the year, how many events they have hosted in a selected year, the number of tickets that were sold and how many events they have cancelled.

The system administrator can also perform an analysis to calculate the amount of revenue the system has made in booking fees and usage fees, the number of venues registered to the system and the amount of event tickets sold/booked on the system.

# Functional Components

This section presets the functional components of the event system.

There are 4 main components in the system including Managing Venues, Events and Bookings and Performing Administration.

# User Requirements

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

## EventSYS will manage Venues

* + 1. EventSYS will register a venue
    2. EventSYS will update a venue
    3. EventSYS will delete a venue

## EventSYS will manage Events

* + 1. EventSYS will add an event
    2. EventSYS will update an event
    3. EventSYS will cancel an event

## EventSYS will manage Bookings

* + 1. EventSYS will sign up customers
    2. EventSYS will allow customers to top up their accounts
    3. EventSYS will make bookings
    4. EventSYS will email tickets
    5. EventSYS will cancel bookings

## EventSYS will perform administrative reporting

* + 1. EventSYS will produce a yearly/monthly venue analysis
    2. EventSYS will analyse yearly revenue

# System Requirements

EventSYS consists of four main modules.

* The manage venues module provides functions to register a venue, update a venue and delete a venue.
* The manage events module provides functions to add an event, update an event and cancel events.
* The manage bookings module will allow customers to sign up, top up their account, make bookings and cancel bookings. This function will also send an email confirmation to customers when a booking has been made.
* The administration module will analyse venues and revenue.

## System Level Use Case Diagram

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

Venue Manager

Customer

Admin

## Manage Venues

This module provides functions to register a venue, update a venue and delete a venue. A venue is assigned a default status of Available when registered on the system to indicate that the venue is available to host events. If a venue is ‘deleted’ the status becomes “Unavailable”.

### Register Venue

This function allows a venue manager to register their venue on the system. Once a venue has been registered the venue manager will then be able to add events and sell tickets.

Venue Manager

<<includes>>

<<extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Register Venue** | |
| **Use Case Id** | 1 | |
| **Priority** | 1 | |
| **Source** | Venue Manager | |
| **Primary Business Actor** | Venue Manager | |
| **Other Participating Actors** |  | |
| **Description** | This function adds a venue to the system. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** A venue manager invokes the Register Venue function  **Step 4:** The manager enters the required data:   * VenueName * Street * Town * Capacity * Contact No * Email * Password   **Step 5:** The manager confirms that the venue is to be registered.  **Step 8:** The manager confirms that the details entered are correct. | **Step 2:** The system assigns the next VenueID and loads on UI.  **Step 3:** Display UI  **Step 6:** The system validates the data entered:   * Venue Name must not already exist * Street, Town and Email must not be null * Capacity must be number, max 4 digits * Contact No must be max 14 digits   **Step 7:** The System displays the details entered and asks the user to confirm.  **Step 9:** The System assigns a default status of “Available”.  **Step 10:** The System saves the data to the Venues File:   * Venue ID * VenueName * Street * Town * Capacity * Contact No * Email * Password * Status “Y”   **Step 11:**  Display confirmation message  **Step 12:** Reset UI |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **VenueName already exists** |  | **Step 6:** VenueName exists  **Step 7:** Display appropriate error message  **Step 8:** Return to step 4 |
| **Street/Town/ Email null** |  | **Step 6:** Information missing  **Step 7:** Display appropriate error message  **Step 8:** Return to step 4 |
| **Conclusions** | The Venue is added to the system | |
| **Post conditions** | Events may now be added to the system for this venue. | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### Update Venue

This function allows a venue manager to update the details of an available venue already registered on the system.

Venue Manager

<<includes>>

<<extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Update Venue** | |
| **Use Case Id** | 2 | |
| **Priority** | 2 | |
| **Source** | Venue Manager | |
| **Primary Business Actor** | Venue Manager | |
| **Other Participating Actors** |  | |
| **Description** | This function updates a venue on the system. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** A venue manager invokes the Update Venue function  **Step 4:** The Manager selects a venue from the list.  **Step 6:** The manager updates the relevant data:   * Name * Street * Town * Capacity * Contact No * Email * Password   **Step 7:** The manager confirms that the venue is to be updated.  **Step 10:** The manager confirms that the details entered are correct. | **Step 2:** The system loads the Update Venue form and a list of ‘Available’ Venues from the Venues file.  **Step 3:** Display UI  **Step 5:** The system loads the venue details of the selected venue to the text boxes from the venues file.  **Step 8:** The system validates the data entered:   * Venue Name must not already exist * Street, Town and Email must not be null. * Capacity must be number, max 4 digits * Contact No must be max 14 digits   **Step 9:** The System displays the details entered and asks the user to confirm.  **Step 11:** The System updates the data in the Venues File:   * VenueName * Street * Town * Capacity * Contact No * Email * Password   **Step 12:**  Display confirmation message  **Step 13:** Reset UI |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **VenueName already exists** |  | **Step 8:** VenueName exists  **Step 9:** Display appropriate error message  **Step 10:** Return to step 6 |
| **Street/Town/ Email null** |  | **Step 8:** Information missing  **Step 9:** Display appropriate error message  **Step 10:** Return to step 6 |
| **Conclusions** | The Venue is added to the system | |
| **Post conditions** | The Venue can now add events to the system | |
| **Business Rules** | Only venues with Status ‘Y can be updated | |
| **Implementation Constraints** |  | |

### Delete Venue

This function allows a venue manager to delete a venue from the system. The system will change the Status of the Venue to ‘N’ to deregister the venue. If there are any upcoming events at the venue chosen, the manager must cancel the event before deregistering the venue.

Venue Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Delete Venue** | |
| **Use Case Id** | 3 | |
| **Priority** | 3 | |
| **Source** | Venue Manager | |
| **Primary Business Actor** | Venue Manager | |
| **Other Participating Actors** |  | |
| **Description** | This function deletes a venue from the system by changing the Venue Status to “Unavailable”. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** A venue manager invokes the Delete Venue function  **Step 4:** The manager selects the VenueName to be removed.  **Step 7:** The manager confirms that the venue is to be deleted. | **Step 2:** The system retrieves a summary of ‘Available’ Venue Names from the Venues File and loads on UI.  **Step 3:** Display UI  **Step 5:** The System retrieves all details for the venue and displays on the UI for viewing only  **Step 6:** If there are upcoming events alert the user and return to step 4 otherwise ask manager to confirm that the venue is to be deleted.  **Step 8:** The System changes the status of the Venue to “N” in the Venues File.  **Step 9:**  Display confirmation message  **Step 10:** Reset UI |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Venue has upcoming events associated with it** |  | **Step 6:** Display message to cancel associated Events before deleting Venue.  **Step 7:** Return to step 4. |
| **Conclusions** | The Venue status is set to “N” in the Venues File. | |
| **Post conditions** | Events can no longer be scheduled for this venue | |
| **Business Rules** | A Venue cannot be deleted if there are upcoming Events. Events must be cancelled and bookings refunded before this can be done.  Only ‘Available’ venues can be deleted. | |
| **Implementation Constraints** |  | |

## Manage Events

### Add Event

This function will allow the venue manager to add an Event to the system. Once added it can then be booked by customers. An Event is assigned a default status of “Scheduled” once created. If an event is cancelled the status is then changed to “Cancelled”.

Venue Manager

<<includes>>

<<extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Add Event** | |
| **Use Case Id** | 4 | |
| **Priority** | 1 | |
| **Source** | Venue Manager | |
| **Primary Business Actor** | Venue Manager | |
| **Other Participating Actors** |  | |
| **Description** | This function adds an event to the system. | |
| **Preconditions** | The Venue has been registered on the system. | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** A venue manager invokes the Add Event function  **Step 5:** The manager enters the required data:   * Venue * Date   **Step 7:** The manager enters the required data:   * Title * Description (optional) * Type * Time * Price * TicketsAvailable   **Step 8:** The manager confirms that the event is to be added.  **Step 12:** The manager confirms that the details entered are correct. | **Step 2:** The system assigns the next EventID  **Step 3:** The system retrieves details of available venues from the Venues file and event types from the Types file and loads on UI.  **Step 4:** Display UI  **Step 6:**  The system determines if venue is available on the date required and if so displays the panel containing the event detail textboxes.  **Step 9:** The System assigns a default status of “Scheduled” to the event.  **Step 10:** The system validates the data entered:   * Title must be entered and max 50 characters. * Description is optional but must be a max 140 characters and. * TicketsAvailable must not be more than the capacity of the venue.   **Step 11:** The System displays the new event details and ask the user to confirm that the details are correct.  **Step 13:** The System saves the data to the Events File:   * EventID * VenueID * Title * Description(optional) * TypeID * Date * Time * TicketsAvailable * Price * Status “S”   **Step 14:**  Display confirmation message  **Step 15:** Reset UI |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Title/ type/ Date/ Time/ Tickets/ Price is null** |  | **Step 9:** Information missing  **Step 10:** Display appropriate error message  **Step 11:** Return to step 7 |
| **TicketsAvailable is more than the Venue’s capacity.** |  | **Step 9:** Capacity exceeded  **Step 10:** Display appropriate error message  **Step 11:** Return to step 7 |
| **Conclusions** | The event is added to the system | |
| **Post conditions** | Tickets for this event can now be booked. | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### Update Event

This function will allow a venue manager to update a scheduled Event. If any updates are made the customers with bookings for the updated event must be emailed and notified of the changes made.

Venue Manager

<<includes>>

<<extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Update Event** | |
| **Use Case Id** | 5 | |
| **Priority** | 1 | |
| **Source** | Venue Manager | |
| **Primary Business Actor** | Venue Manager | |
| **Other Participating Actors** |  | |
| **Description** | This function allows the venue manager to update event details on the system. It also sends an email to customers with bookings for the associated event. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** A venue manager invokes the Update Event function  **Step 4:** The manager selects the Event Title  **Step 6:** The manager updates the required data:   * Title * Description (optional) * Type * Date * Time * TicketsAvailable * Price   **Step 7:** The manager confirms that the event is to be updated.  **Step 10:** The manager confirms  that the details entered  are correct. | **Step 2:** The system loads a list of upcoming event titles from the Events file where the event status is “Scheduled” and loads the event types from the Types file.  **Step 3:** Display UI  **Step 5:** The system loads the information from the selected event to the text boxes.  **Step 8:** The system validates the data entered:   * Title must be entered and max 50 characters. * Description is optional but must be a max 140 characters and. * There cannot be another event in the same venue on the selected date. * TicketsAvailable must not be more than the capacity of the venue.   **Step 9:** The System displays  the event’s new details and  ask the user to confirm  that the details are correct.  **Step 11:** The System updates the event with the new data in the Events File:   * Title * Description(optional) * Type * Date * Time * TicketsAvailable * Price   **Step 12:**  Display confirmation message  **Step 13:** Reset UI |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Title/ type/ Date/ Time/ Tickets/ Price is null** |  | **Step 8:** Information missing  **Step 9:** Display appropriate error message  **Step 10:** Return to step 6 |
| **Event already taking place on the date specified in the selected venue** |  | **Step 8:** Clashing events  **Step 9:** Display appropriate error message  **Step 10:** Return to step 6 |
| **TicketsAvailable is more than the Venue’s capacity.** |  | **Step 8:** Capacity exceeded  **Step 9:** Display appropriate error message  **Step 10:** Return to step 6 |
| **Conclusions** | The event details are updated on the system | |
| **Post conditions** |  | |
| **Business Rules** | Only upcoming, scheduled events can be updated. | |
| **Implementation Constraints** |  | |

### Cancel Event

This function allows the venue manager to cancel an upcoming event. If an event is cancelled, the status is set to “N” and customers must be emailed and notified.

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Cancel Event** | |
| **Use Case Id** | 6 | |
| **Priority** | 3 | |
| **Source** | Venue Manager | |
| **Primary Business Actor** | Venue Manager | |
| **Other Participating Actors** |  | |
| **Description** | This function allows the venue manager to cancel an upcoming event on the system. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** A manager invokes the Cancel Event function  **Step 4:** The manager selects the Event Title and Reason for Cancellation  **Step 6:** The manager views the Event details and selects Delete.  **Step 8:** The manager selects ‘Yes’ to confirm that the event is to be cancelled. | **Step 2:** The system loads a list of upcoming event titles where the status is set to “S”.  **Step 3:** Display UI  **Step 5:** The system displays the selected event’s details.  **Step 7:** The system displays a message asking manager to confirm cancellation.  **Step 9:** The system sets:   * Status of the event to “C” in the Events File. * Increases the balance of the customers that had a booking for the event in the Customers file * The bookings for the cancelled event are deleted from the Bookings File.   **Step 10:**  Display confirmation message  **Step 11:** Reset UI |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Manager changes mind and selects ‘No’ when asked to confirm cancellation** | **Step 6:** The manager selects ‘No’ to confirm that the event is to be cancelled. | **Step 7:** Reset UI |
| **Conclusions** | The Event is updated in the Events File. | |
| **Post conditions** | The Event can no longer be booked by customers. | |
| **Business Rules** | Only scheduled, upcoming events can be booked. Customers with bookings related to the Event must be notified about the cancellation by email and a refund made to their account. | |
| **Implementation Constraints** |  | |

## Manage Bookings

### Sign Up Customer

This function allows a customer to sign up to the system. Once signed up a customer can book tickets for upcoming, scheduled events and top up their accounts to do so.

Customer

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Sign Up Customer** | |
| **Use Case Id** | 7 | |
| **Priority** | 1 | |
| **Source** | Customer | |
| **Primary Business Actor** | Customer | |
| **Other Participating Actors** |  | |
| **Description** | This function signs a customer up to the system. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** A customer invokes the Sign-Up Customer function  **Step 4:** The customer enters the required data:   * Forename * Surname * Email * Password * ContactNo   **Step 5:** The customer confirms that they would like to sign up.  **Step 10:** Customer selects ‘Yes’ to Top Up their Account. | **Step 2:** The system assigns the next CustID and loads on UI.  **Step 3:** Display UI  **Step 6:** The system validates the data entered:   * Customer email must not already exist * ContactNo must be max 14 digits * Password cannot be null and max 8 characters   **Step 7:** The System saves the data to the Customers File:   * CustID * FName * SName * Email * Password * ContactNo   **Step 8:**  The system displays a confirmation message to say the customer has been registered.  **Step 9**: The system displays a message asking the customer would they like to proceed to the top up form.  **Step 11:** The system closes the Sign-Up form**.**  **Step 12:** The system loads the Top Up Account form. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Customers email already exists in the Customers File** |  | **Step 6:** Account already exists  **Step 7:** Display appropriate error message  **Step 8:** Return to step 4 |
| **Password/ Name/ Contact No is required** |  | **Step 6:** Information missing  **Step 7:** Display appropriate error message  **Step 8:** Return to step 4 |
| **Conclusions** | The Customer is added to the Customer File. | |
| **Post conditions** | The Customer can now book events on the system. | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### Top Up Account

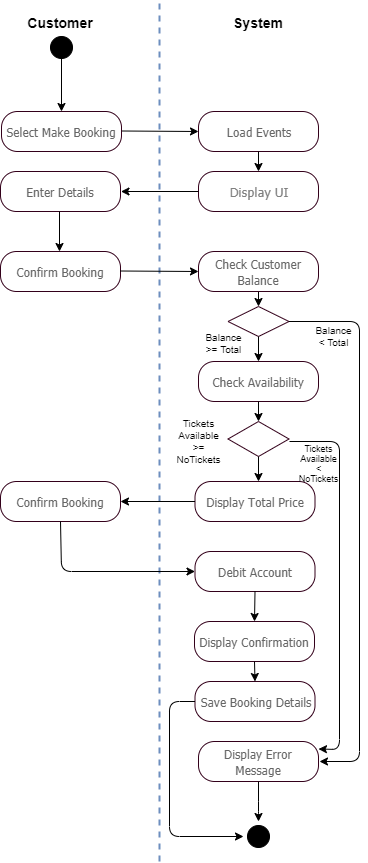
This function allows a customer to Top Up their account Balance so they can make bookings on the system.

Customer

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Top Up Account** | |
| **Use Case Id** | 8 | |
| **Priority** | 1 | |
| **Source** | Customer | |
| **Primary Business Actor** | Customer | |
| **Other Participating Actors** |  | |
| **Description** | This function allows a customer to top up their account balance. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** A customer invokes the Top Up Account function  **Step 4:** The customer enters their account data:   * Email * Password   **Step 5:** The customerselects “Sign In”.  **Step 7:** The customer enters the required data:   * Top Up Amount * Card Number * Card Expiry * CVV   **Step 8:** The customer confirms that they would like to top up. | **Step 2:** The system loads the Top Up Account Form.  **Step 3:** Display UI  **Step 6:** The system validates the login details and loads the account Balance, Forename and Surname to the textboxes from the record with the associated email and password in the Customer File.  **Step 9:** The systemvalidates the data:   * The Card Number must be 16 digits * The CVV must be 3 digits * The Top Up Amount must be more than 0 * The Expiry date must be more than today’s date.   **Step 10:** The System increases the Customers Balance by the amount entered in the Top Up Amount field.  **Step 11:**  Display confirmation message  **Step 12:** Reset UI |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Card holder name does not match Customer name** |  | **Step 9:** You cannot use someone else’s card, details must match account  **Step 10:** Display appropriate error message  **Step 11:** Return to step 7 |
| **Card number is not 16 digits, CVV is not 3 digits** |  | **Step 9:** Problem with card  **Step 10:** Display appropriate error message  **Step 11:** Return to step 7 |
| **Top Up Amount is null** |  | **Step 9:** Top Up Amount must be more than 0  **Step 10:** Display appropriate error message  **Step 11:** Return to step 7 |
| **Expiry Date has is in the past** |  | **Step 9:** Card is expired  **Step 10:** Display appropriate error message  **Step 11:** Return to step 7 |
| **Conclusions** | The Customer’s account balance is increased. | |
| **Post conditions** | The Customer can now use their balance to book events on the system. | |
| **Business Rules** | The Customer must top up with a card in their own name. | |

### Make Booking

This function allows the Customer to buy tickets for an available event. Once tickets have been bought, the system updates the number of tickets available and sends the customer an email to confirm their booking.



Customer

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Make Booking** | |
| **Use Case Id** | 9 | |
| **Priority** | 1 | |
| **Source** | Customer | |
| **Primary Business Actor** | Customer | |
| **Other Participating Actors** |  | |
| **Description** | This function allows a customer to make a booking for upcoming, scheduled events. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** A customer invokes the Make Booking function  **Step 5:** The customer selects an event.  **Step 7:** The customerenters their email and password and selects ‘Login’  **Step 9:** The customer enters the number of tickets they would like to purchase and selects ‘Book’. | **Step 2:** The system loads a list of upcoming Events where the Status is ‘S’ from the Events file.  **Step 3:** The system loads filter options of :   * types from the Types File * towns from the Venues File * venues from the Venue File     **Step 4:** Display UI  **Step 6:** The system displays a pop up for the customer to sign into their account.  **Step 8:** The system validates the customer’s details and displays their balance and options to make a booking.  **Step 10:** The system displays the total booking price.  **Step 11:** The systemvalidates the data:   * Number of tickets cannot be more than 4 * The customer’s account balance must exceed the total booking price. * There must be enough tickets available.   **Step 12:** The System must update the customer’s balance in the Customer file  **Step 13:** The system reduces the TicketsAvailable in the Events File.  **Step 14:** The system must generate a BookingID and the data is saved to the Bookings File   * BookingID * CustID * EventID * BookingDate * No Tickets * Price   **Step 15:**  Display confirmation message and BookingID and send email  **Step 16:** Reset UI |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Email and Password do not match an existing record in the Customers File** |  | **Step 6:** Invalid login  **Step 7:** Display appropriate error message  **Step 8:** Return to step 7 |
| **Unavailable funds** |  | **Step 9:** Please top up account  **Step 10:** Display appropriate error message  **Step 11:** Return to step 4 |
| **Tickets unavailable** |  | **Step 9:** Display the number of tickets available for the event  **Step 10:** Display appropriate error message  **Step 11:** Return to step 9 |
| **Conclusions** | The customer has booked tickets for an event. | |
| **Post conditions** |  | |
| **Business Rules** | The customer can book a maximum of 4 tickets at one time. The event they are booking must be in the future and have a status of “S”. | |
| **Implementation Constraints** |  | |

### Cancel Booking

This function allows a customer to cancel a booking. Once cancelled, their booking will be deleted from the system and the tickets available will be increased for the event. If the cancellation is made more than 24 hours before the event start time the customer will receive a full refund and their balance will be increased, otherwise they will not receive a refund.

Customer

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Cancel Booking** | |
| **Use Case Id** | 10 | |
| **Priority** | 3 | |
| **Source** | Customer | |
| **Primary Business Actor** | Customer | |
| **Other Participating Actors** |  | |
| **Description** | This function allows a customer to cancel a booking for an upcoming event. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** A customer invokes the Cancel Booking function.  **Step 4:** The customer logs in with the required data:   * Email * Password   **Step 5:** The customerselects “Find Bookings”.  **Step 8:** The customer selects the booking and “Cancel ”.  **Step 10:** The customer selects “Yes”. | **Step 2:** The system loads the Cancel Booking function.  **Step 3:** Display UI  **Step 6:** The systemvalidates the data:   * Email and password must be associated with a record in the Customers File   **Step 7:** The system loads a list of upcoming bookings associated with the customer.  **Step 9:** A confirmation message will ask the customer if they are sure they want to cancel and will notify them if they are due a refund.  **Step 11:** The System must update data:   * The customer’s balance must be increased by booking price if event is more than 24 hours away in the Customers File. * The Tickets Available must be increased in the Events File. * The system deletes the booking record from the Booking File.   **Step 12:**  Display confirmation message with message to say if account has been refunded or not.  **Step 13:** Reset UI |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Customer changes their mind.** | **Step 10:** The customer selects “No”. | **Step 11:**  Display message.  **Step 12:** Reset UI |
| **The event is less than 24 hours away** |  | **Step 9:** The customer is notified they will not be refunded if they proceed |
| **Conclusions** | The Booking is deleted from the system. | |
| **Post conditions** |  | |
| **Business Rules** | Customer is only refunded if the event is more than 24 hours away | |
| **Implementation Constraints** |  | |

## Perform Administration

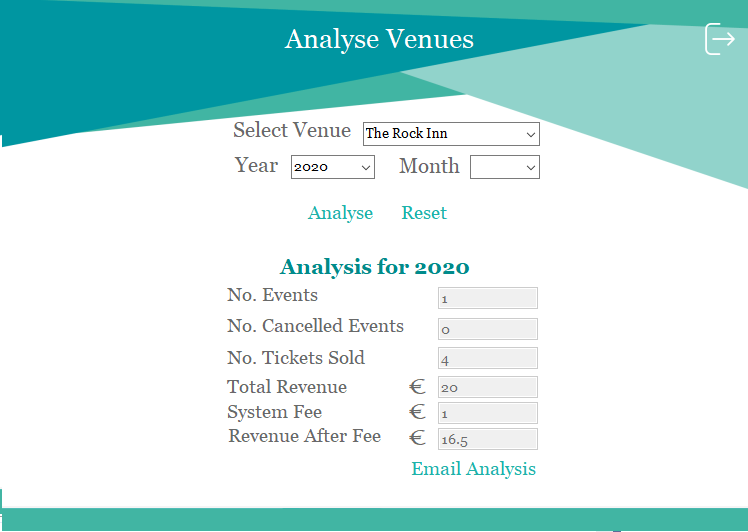
### Analyse Venues

This function will carry out an analysis of individual venues. The analysis will display the number of events that took place in the selected year/month, the number of cancelled events, the total revenue excluding system booking fee, the total number of tickets sold and the system usage fee for the year which is 5% of total revenue.

Venue Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Analyse Venue** | |
| **Use Case Id** | 11 | |
| **Priority** | 3 | |
| **Source** | Venue Manager | |
| **Primary Business Actor** | Venue Manager | |
| **Other Participating Actors** |  | |
| **Description** | This function allows a venue manager to carry out an analysis of their venue for a selected year/month. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** A venue manager invokes the Analyse Venue function.  **Step 4:** The manager selects the appropriate venue.  **Step 6**: The manager selects “Analyse”.  **Step 7:** The manager selects “Email Analysis”. | **Step 2:** The system loads a list of Venues to a combo box.  **Step 3:** Display UI  **Step 5:** The system loads a list of years in which the venue has events and/or bookings.  **Step 7:** The system finds the venue and produces the following information for the chosen year/month:   * Number of events that took place. * Number of cancelled events. * Total revenue excluding system booking fee. * Total number of tickets sold. * System usage fee for the year, 5% of total revenue. * Total revenue after Fees   **Step 8:** The system retrieves the venue email and sends the gathered information.  **Step 9:**  Display confirmation message.  **Step 10:** Reset UI |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **There are no events held or tickets sold by the selected venue in the selected year** |  | **Step 6:** No analysis available  **Step 7:** Display confirmation message  **Step 8:** return to step 4 |
| **Conclusions** | The manager receives an analysis of venue for the year. | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

A screenshot of a cell phone

Description automatically generated

Venue Analysis by year only

Venue Analysis by Year & Month

### Analyse Revenue

This function will analyse the yearly system revenue. It will display the yearly revenue made from booking fees, the yearly revenue from ticket sales, the total revenue for the year, the number of available venues and the number of tickets sold on system.

Admin

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Analyse Revenue** | |
| **Use Case Id** | 12 | |
| **Priority** | 2 | |
| **Source** | Admin | |
| **Primary Business Actor** | Admin | |
| **Other Participating Actors** |  | |
| **Description** | This function allows the system admin to calculate their yearly revenue. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The administrator invokes the Analyse Revenue function  **Step 4:** The administrator selects the year they would like to analyse from the list. | **Step 2:** The system loads a list of years where there have been Events and/or Bookings on the system to a combo box .  **Step 3:** Display UI  **Step 5:** The systemloads:   * a chart with data containing the total revenue in booking fees for each month applicable * The sum of all booking fees for the selected year. * a chart with data containing the total revenue in system fees for each month applicable * The sum of all system fees for the selected year. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
|  |  |  |
| **Conclusions** | The system administrator now has an analysis of yearly revenue. | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

A screenshot of a cell phone

Description automatically generated

# System Model

The following dataflow diagrams have been produced for the system:



## Level-0 DFD

**Customer**

**Venue Manager**

## Level-1 DFD

**Event Type File**

**D5**

Event Details

Venue Details

Event Types

**Venue Manager**

Venue Details

**Venues File**

**D2**

**P2**

**Manage Events**

**P1**

**Manage Venues**

Venue Details

Event Details

Venue   
Details

**Events File**

**D1**

Event Details

Event Details

Booking Details

**P3**

**Manage  
Bookings**

**P4**

**Perform Admin**

**Bookings File**

**D3**

Customer Details

Booking Details

**Customer File**

**D4**

Venue Details

**Venue Manager**

Booking Details

**Customer**

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

**Venue Manager**

Venue Details

Venue Details

**P1.1**

**Register Venue**

Venue Details

Venue Details

**P1.3**

**Delete Venue**

Venue Details

Updated Venue Details

Venue Details

Updated Venue Details

**Venue Manager**

**P1.2**

**Update Venue**

Venue Status

**Venues File**

**D2**

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

**Venue Manager**

Event Details

Event Details

**P2.1**

**Add Event**

Event Details

Event Details

Details

**P2.3**

**Cancel Event**

Event Types

Event Types

**Event Type File**

**D5**

Event Details

Updated Event Details

Event Details

Updated Event Details

**Manager**

**P2.2**

**Update Event**

Event Status

**Events File**

**D1**

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

Booking Details

Customer Details

**Customer**

**P3.1**

**Sign Up Customer**

Customer Details

**P3.3**

**Make Booking**

**Customer File**

**D4**

Event Details

Venue Name

Customer Balance

**Events File**

**D1**

**Venues File**

**D2**

Booking Details

Venue Name

**Bookings File**

**D3**

**P3.3**

**Cancel Booking**

**P3.2**

**Top Up Account**

Updated Details

Booking Details

Booking Details

Customer Balance

**Customer**

Updated Details

Top up Account

## Level-2 DFD (Process P3: Perform Administration)

**Venue Manager**

Venue Details

Venue Analysis

Venue Details

**P4.2**

**Analyse Revenue**

Venue Details

**Venues File**

**D2**

**P4.1**

**Analyse Venues**

Event Details

**Events File**

**D1**

Booking Details

Booking Details

**Bookings File**

**D3**

# Data Model (Class Diagram)

|  |
| --- |
| Booking |
| BookingID\* BookingDate  NoTickets  BookingTotal |

## Class Diagram

1..\*

1

is booked

1..\*

Makes a

1

has a

EventType  
TypeID\*  
Description

|  |
| --- |
| Event |
| EventID\* Title  Description EventDate  Time  TicketsAvailable Price  Status  1..\*  0..\* |

|  |
| --- |
| Customer  0..1 |
| CustID\* FName  Sname  Email  Password  ContactNo  Balance |

adds an

|  |
| --- |
| Venue  0..1 |
| VenueID\* VenueName  Street  Town  Capacity  ContactNo  Email  Status  Password |

## Relational Schema

Venues (VenueID, VenueName, Street, Town, Capacity, ContactNo, Email, Status, Password)  
EventType (TypeID, Description)  
Events (EventID, VenueID, Title, Description, TypeID, EventDate, Time, TicketsAvailable, Price, Status)  
Customers (CustID, FName, Sname, Email, Password, ContactNo, Balance)  
Bookings (BookingID, CustID, EventID, BookingDate, NoTickets, BookingTotal)

## Database Schema

**Schema:** EventSystem

**Relation:** EventType  
**Attributes**:   
TypeID numeric (2) not null  
Description varchar2 (15) not null

Primary Key TypeID

**Relation:** Venues  
**Attributes**:   
VenueID numeric (4) not null  
VenueName varchar2 (30) not null unique  
Street varchar2 (30) not null  
Town varchar2 (20) not null  
Capacity numeric (4) not null  
ContactNo varchar2 (14) not null  
Email varchar2 (30) not null  
Status varchar2 (1) default = ‘Y’   
Password varchar2 (8) not null  
  
Primary Key VenueID

**Relation:** Events  
**Attributes**:   
EventID numeric (4) not null unique  
VenueID numeric (4) not null  
Title varchar2 (50) not null  
Description varchar2 (140)   
TypeID numeric (4) not null  
EventDate date not null  
Time varchar2(5) not null  
TicketsAvailable numeric (3)  
Price numeric (5,2)   
Status char (1) default = ‘S’  
  
Primary Key EventID   
Foreign Key VenueID References Venues  
Foreign Key TypeID References EventTypes

**Relation:** Customers  
**Attributes**:   
CustID numeric (6) not null unique  
FName varchar2 (20) not null  
Sname vacrchar2 (20) not null  
Email varchar2 (30) not null unique  
Password varchar2 (8) not null  
ContactNo varchar2 (14) not null  
Balance numeric (6,2) default = 0  
  
Primary Key CustID

**Relation:** Bookings  
**Attributes**:   
BookingID numeric (8) not null unique  
CustID numeric (6) not null  
EventID numeric (4) not null  
BookingDate date not null  
NoTickets numeric (1) not null  
BookingTotal numeric (6,2) not null  
  
Primary Key BookingNo  
Foreign Key VenueID References Venues  
Foreign Key CustID References Customers

# Conclusion

Throughout the process of designing my Event System I have learnt the importance of requirement gathering particularly in terms of designing a system that can easily adapt to changes and that abides by business rules. However, keeping in mind that precise planning can also prevent the need for changes.

I am pleased with this release of the system and the adjustments made in producing a better statistical analysis for administration to get an overview of the system’s finances. The validation throughout the system ensures not only that the business rules are all met but also prevents the system from crashing due to invalid data types. The system design has also been completely revamped since the last release making it much more simplistic. With the use of appearing and disappearing panels I have also increased the ease of use by guiding the user through each function seamlessly. These modifications enhance non-functional requirements such as Usability.

I feel as though the system suits its purpose as intended and with a fee that works from a percentage of the revenue generated through the system, local businesses, especially small and non-profit business will benefit largely from the services provided.