Concert Management

Software Requirements Specification

1.0

9/11/2023

Megha Manoj
Lead Software Engineer

Chelsy Clevia Fernandes
Lead Software Engineer

Prepared for
IS 341 —Software Engineering
Instructor: Dr. Pramod Gaur, Ph.D.

Revision History

Date	Description	Author	Comments
9/11/2023	<Version 1>	Megha Manoj	-

Document Approval

The following Software Requirements Specification has been accepted and approved by the following:

Signature	Printed Name	Title	Date
	Megha Manoj	Lead Software Eng.	9/11/2023
	Chelsy Clevia Fernandes	Lead Software Eng.	9/11/2023
	Dr. Pramod Gaur	Instructor, IS 341	9/11/2023

Table of Contents

1. Introduction

The following document has been created to record the functionalities, relevant details and changes incorporated in the project 'Concert Management'.

1.1 Purpose

The purpose of this SRS is to provide an overview of the designed concert management software built based on the requirements of the entertainment industry with respect to customer as well as employee requirements.

1.2 Scope

This subsection should:

(1) The product to be developed is a web application for entertainment companies which operates using server-side Python code, HTML, JavaScript and SQLite.

(2) The created product makes use of data collection and information retrieval to manage artists, venues and customers.

(3) The application of the developing product is as follows:

- (a) *Access control of the application via 2 login pages to separate usage between employee and customers.*
- *(b) Web pages for users to view available concerts and show dates.*
- *(c) The employees have the authority to edit any information present on the site.*
- *(d) Employees can view bookings made by users.*

1.3 Overview

The concert management system has the following functions:

- *(1) Login page accessible to both employees as well as the customers.*
- *(2) The customers can*
  - *(i) Create an account*
  - *(ii)Book their private event slots*
  - *(iii) View upcoming events*

- *(3) The employees can*
  - *(i) Create events*
  - *(ii) Delete events*
  - *(iii) View client bookings*

2. General Description

2.1 Product Perspective

The concert management system is a management system for an event management company to organize and manage their concerts. The aim of this management system is to allow clients to register their membership and book private concerts based on the available events and allow company employees to manage the concerts.

2.2 Product Functions

2.2.1. Creation of events

- Create new concert events.
- Add details like event name,time and date.

2.2.2. Deletion of events

2.2.3. User management

- View client bookings.

2.2.4. Event Booking

- View upcoming concerts and relevant information.
- Book an event based on requested timing.

2.3 User Characteristics

1. Customers:

- *Role: Visitors interested in events booking.*
- *Characteristics:*
- *Create user accounts.*
- *Login through verified account username and password.*
- *Seek free concert information.*
- *Book private concert slots.*

2. Employees:

- *Role: Authorized administrators.*
- *Characteristics:*
- *Special login portal with authorization.*
- *Manage concert information.*
- *Ensure accurate content.*
- *Check concert bookings by clients.*

3. General Users (Customers and Employees):

- *Role: All visitors.*
- *Characteristics:*
- *Access concert information.*
- *Login.*
- *Expect a user-friendly interface.*
- *Expect responsive performance.*

2.4 General Constraints

- *The entered email address for password retrieval may not be correct.*
- *The django server used may not be active during code maintenance.*

- *The user interface must be user friendly and responsive. The application must adapt to the screens.*
- *The application must be able to handle several simultaneous users.*
- *The large images/videos displayed must be loaded quickly.*
- *The web application should have the capability to update itself if a concert has been canceled.*

2.5 Assumptions and Dependencies

- *Once registered, users can freely gain access to all upcoming events listed*
- *The information regarding event timings and artist availability are constantly updated by employees.*
- *The databases in SQLite store all relevant user account information.*

3. Specific Requirements

This will be the largest and most important section of the SRS. The customer requirements will be embodied within Section 2, but this section will give the D-requirements that are used to guide the project's software design, implementation, and testing.

Each requirement in this section should be:
- *Correct*
- *Traceable (both forward and backward to prior/future artifacts)*
- *Unambiguous*
- *Verifiable (i.e., testable)*
- *Prioritized (with respect to importance and/or stability)*
- *Complete*
- *Consistent*
- *Uniquely identifiable (usually via numbering like 3.4.5.6)*

3.1 External Interface Requirements

3.1.1 User Interfaces

3.1.2 Hardware Interfaces

Users must possess devices compatible with the latest version of Google Chrome or any other web browser.

3.1.3 Software Interfaces

Visual studio code 2022 is required for hosting the project due to its ease of use and access to documentation.

3.2 Functional Requirements

The following features have been designed specifically with respect to the management nature of the organization

3.2.1 User registration and authentication

3.2.1.1 Introduction

There are 2 HTML sites designed with respect to the login process:

- User login page: to authenticate users and control their level of access to the application features.
- User registration form: Creation of account for new users to use for their booking processes.

3.2.1.2 Inputs

The inputs of each page are as follows:

- User login page:
  - Username
  - Password
- User registration form:
  - Username
  - Email address
  - Gender
  - Password
  - Password confirmation

3.2.1.3 Processing

The provided information for registration is saved in SQLite as a record for each user account.

For authenticating login details the password and format of the email address will be compared with that of the stored details. The employees will be provided a unique email address and password by the company.

3.2.1.4 Outputs

Upon successful login, the user will be led to a home page.

3.2.1.5 Error Handling

Any error upon login will be highlighted by the box where incorrect information was entered.

3.2.2 User home page

3.2.2.1 Introduction

This particular page consists of 2 tabs leading to viewing concert information, concert booking and account logout respectively.

3.2.2.2 Inputs

There are no inputs to this particular feature, rather links are present which direct the users to the previously mentioned pages.

3.2.2.3 Processing

Rendering of web pages is performed for directing the users to another page.

3.2.2.4 Outputs

The web page leads to other information and booking pages.

3.2.2.5 Error Handling

Any error upon login will be deflected by reloading the page.

3.2.3 User event booking

3.2.3.1 Introduction

Customers can reserve their private slots for an upcoming event if unable to attend the scheduled ones.

3.2.3.2 Inputs

In order to book a ticket, the users are required to enter the following details:

- Concert ID
- Date
- Time

3.2.3.3 Processing

The submitted details are collected through the django text boxes and added to the database.

3.2.3.4 Outputs

User is redirected to the same page.

3.2.3.5 Error Handling

Any error will result in automatic reload by browser..

3.2.4 View upcoming events

3.2.4.1 Introduction

Customers can view events which they can attend via their membership with the entertainment organization.

3.2.4.2 Inputs

- There is no input for viewing the available concerts.

3.2.4.3 Processing

The data regarding event details is loaded from SQLite dataframe.

3.2.4.4 Outputs

User is redirected to the same page.

3.2.4.5 Error Handling

Any error will result in automatic reload by browser.

3.2.5 Admin registration and authentication

3.2.5.1 Introduction

A distinct login page for admin where employees can access the concert management portal with credentials provided by the company.

3.2.5.2 Inputs

The inputs of each page are as follows:

- User login page:
  - Username: admin
  - Password: admin123

3.2.5.3 Processing

For authenticating login details the password and format of the email address will be compared with that of the stored details.

3.2.5.4 Outputs

Upon successful login, the admin will be led to an admin home page.

3.2.5.5 Error Handling

Any error upon login will be highlighted by the box where incorrect information was entered.

3.2.6 Admin home page

3.2.6.1 Introduction

This particular page consists of 2 tabs leading to viewing client information,editing concert information and account logout respectively.

3.2.6.2 Inputs

There are no inputs to this particular feature, rather links are present which direct the users to the previously mentioned pages.

3.2.6.3 Processing

Rendering of web pages is performed for directing the users to another page.

3.2.6.4 Outputs

The web page leads to other data management pages.

3.2.6.5 Error Handling

Any error upon login will be deflected by reloading the page.

3.2.7 View booked slots

3.2.7.1 Introduction

Employees can view client information and their booked slots for verification of details.

3.2.7.2 Inputs

- There is no input for viewing the available concerts.

3.2.7.3 Processing

The data regarding event details is loaded from SQLite dataframe.

3.2.7.4 Outputs

User is redirected to the same page.

3.2.7.5 Error Handling

Any error will result in automatic reload by browser.

3.2.8 Edit concert information

3.2.8.1 Introduction

Employees can create records of upcoming concerts, update and delete them.

3.2.8.2 Inputs

- Event name
- Venue
- Date
- Time

3.2.8.3 Processing

The data regarding event details is stored into SQLite dataframe.

3.2.8.4 Outputs

User is redirected to the same page.

3.2.8.5 Error Handling

Any error will result in automatic reload by browser.

3.3 Use Cases

3.3.1 User Registration and Authentication:

- Use Case 1: User Registration

  Actors: New users (customer)

  Description: New users can create an account by providing their first name, last name, email address, and password through the user registration form.

- Use Case 2: User Authentication

  Actors: Registered users (customer)

  Description: Registered users can log in by entering their email address and password on the user login page, and their access to application features is controlled based on their authentication.

- Use Case 3: Forgot Password

  Actors: Registered users

  Description: Users who have forgotten their password can access a new password via their email address by using the "Forgot password" page.

3.3.2 Events Page:

- **Use Case 4: View concert information**

  Actors: Customer

  Description: Users can view information about a list of upcoming concerts.

- **Use Case 5: View Concert information**

  Actors: employee

  Description: Users can access information about upcoming concerts.

3.3.3 **Slot Booking**:

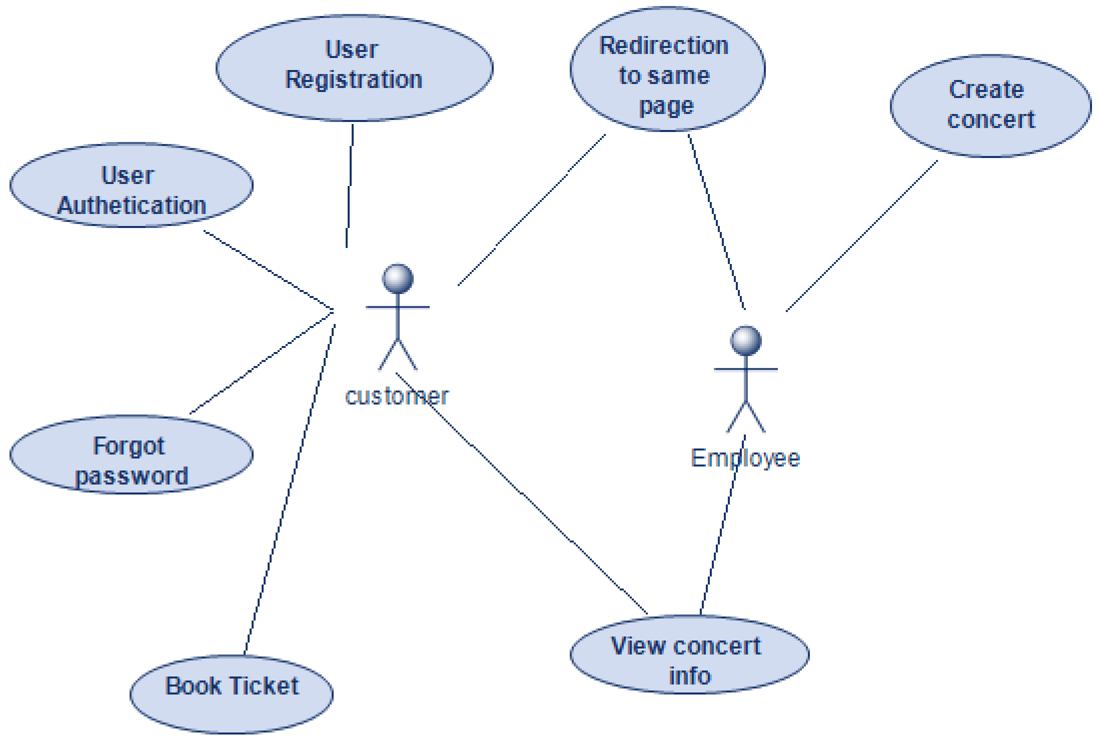
- **Use Case 6: Book Ticket**
  Actors: Customers
  Description: Customers can book their preferred time slot and date for events listed as upcoming.

- **Use Case 7: Redirection to same page**
  Actors: Customers
  Description: After submitting booking details, customers are redirected after their booking has been successfully made.

3.3.4 **Admin concert updates**:

- **Use Case 8: Create concert**
  Actors: Admin (employee)
  Description: Admin can create concerts which take place in the upcoming week with details related to venue and date mentioned.

- **Use Case 9: Redirection to same page**
  Actors: Employee
  Description: After submitting booking details, customers are redirected after their booking has been successfully made.



Use case Diagram

3.4 Classes / Objects

3.4.1 User

3.4.1.1 Attributes

- Username - String
- Email-id - String
- Gender- String
- Password - String

3.4.1.2 Functions

- Slot booking- Function for accessing event details- the date and name of the event after submission of booking form has been completed.
- Access control - Enter login details to access the user home page.
- View events- View upcoming events.

3.4.2 Employee

3.4.1.1 Attributes

- Username - String
- Password - String

3.4.1.2 Functions

- Edit event details - Function for modifying the details about an event.
- Access control - Verify the login details submitted and direct access based account verification.

3.5 Non-Functional Requirements

3.5.1 Performance

- ***Responsive Time:*** The system should respond to the actions of the user within a specific time frame.
- ***Scalability:*** The system must be capable of managing a growing volume of users and events without noticeably losing performance.
- ***Concurrency:*** It should not have performance bottlenecks when handling a certain number of concurrent users.

3.5.2 Reliability

- ***Data Integrity:*** Make sure that data is reliable and consistent over its entire life.
- ***Fault Tolerance:*** The system must go on to function even if there are hardware or software failures.

3.5.3 Availability

- The system must be available 24//7 with no interruption in its access.
- ***Uptime requirement:*** Should specify the required minimum uptime i.e. the time for which the system would be available.
- ***Downtime Allowance:*** The system downtime should be

3.5.4 Security

- ***Authentication:*** Users must be able to securely login and their access to the functionalities and data must be based on their permissions and roles.
- ***Authorization:*** Make sure that users can only access data and carry out only authorized actions.
- ***Data Encryption:*** To prevent unwanted access, sensitive data, such as payment information, should be encrypted.

3.5.5 Maintainability

It ensures that the system is simple to update, modify and fix over time.

- ***Documentation:*** Maintain a detailed documentation that has system architecture, code and data structures.
- ***Modularity:*** The system needs to be built using a modular architecture, where various parts are kept apart and arranged rationally.
  A clear and well-defined interface for each module or component will make it simpler to replace or update certain system components without affecting the overall system.
- ***Quality of Code:*** To make sure the codebase is clear, understandable, and well-structured, enforce coding standards and best practices. To make code maintenance easier, use meaningful variable and function names, comments, and version control.

3.5.6 Portability

- ***Platform Independence:*** The concert management system should be platform-independent, which means it should be able to run on several operating systems (such as Windows, macOS, and Linux) without requiring much modification.
- ***Database Compatibility:*** The system should be able to work in different database management systems.
- ***Web Browser Compatibility:*** The interface must be compatible with the web browsers like Chrome, Edge etc.
- ***API and Integration Support:*** The system should be able to provide APIs for integration with external services which would enhance its ability to adapt and interoperability.

3.6 Inverse Requirements

- ***User Registration and Authentication:***
  - *The system must not allow users to register without providing a valid email address.*
  - *The system must not allow users to change their email address without proper authentication.*
- ***User Concerts Page:***
  - *The system must not allow unauthorized users to edit the concert page template.*
  - *The system must not display concert information if they contain incomplete or erroneous data.*
- ***Concert Slot Booking:***
  - *The system must not allow users to book tickets for events that have already passed.*
  - *The system must not allow users to book tickets for events that are marked as canceled or unavailable.*

3.7 Design Constraints

- *User feedback: Fixing bugs present in the application based on usability may take time if the format of the websites is ambiguous, as the HTML templates may require modification.*

3.8 Logical Database Requirements

- ***User Data Collection***
    - *Define a SQLite data frame named "users" to store user data.*
        - *Fields within the "users" collection:*
        - *first_name: String*
        - *last_name: String*
        - *email: String (unique)*
        - *password: String (hashed)*

- ***Artist Data Collection***
    - *Create a SQLite collection named "artists" to store artist-related information.*
        - *Fields within the "artists" collection:*
        - *name: String*
        - *genre: String*
        - *description: String*

- ***Event Data Collection***
    - *Establish a SQLite collection named "events" to manage event information.*
        - *Fields within the "events" collection:*
        - *name: String*
        - *date: Date*

- ***Ticket Booking Data Collection***
    - *Create a SQLite collection named "ticket_bookings" to store ticket booking records.*
        - *Fields within the "ticket_bookings" collection:*
        - *event: Reference to the "events" collection (event ID)*
        - *user: Reference to the "users" collection (user ID)*
        - *experience_type: String*

- ***Indexing and Querying***
- *Define unique indexes on the email field in the "users" collection to ensure email uniqueness.*
- *Consider indexing fields used frequently for searching or sorting, such as event.date.*
- *Implement efficient queries for user authentication by searching for a user with a given email and verifying the hashed password.*
- *Enable queries to retrieve artist information by artist name, genre, or other relevant criteria.*
- *Allow queries to list events by date or other relevant parameters.*
- *Support queries to retrieve ticket booking information for a specific user or event.*
- ***Data Validation and Constraints***
    - *Implement data validation rules to ensure that required fields are present and data types are correct (e.g., email format validation).*
    - *Enforce constraints to prevent data inconsistencies or invalid references (e.g., ensuring that a ticket booking references an existing event and user).*
    - *Define rules to maintain data integrity, such as cascading updates or deletions when an artist or event is modified or deleted.*

4. Test Cases

To ensure proper working of the program the following test cases were implemented:

- **Admin login redirection**
  - Positive Test Case:
    Input: User attempts to access the admin login page from the user login page.
    Expected Output: Successful redirection to the admin login page.
  - Negative Test Case:
    Input: User attempts to access the admin login page from the user login page but is not an admin.
    Expected Output: Display an error message or remain on the user login page, indicating unauthorized access.

- **User login**
  - Positive Test Case:
    Input: Valid username, email, and password.
    Expected Output: Successful registration message.
  - Negative Test Case:
    Input: Existing username or email.
    Expected Output: Error message indicating the unavailability of the username or email.

- **Concert updation**
  - Positive Test Case:
    Input: Valid event ID and updated event details.
    Expected Output: Successful update message, event details changed.
  - Negative Test Case:
    Input: Invalid event ID or incomplete updated details.
    Expected Output: Error message indicating the issue with event updating.

- **Event Booking:**
  - Positive Test Case:
    Input: Valid event ID and user details.
    Expected Output: Successful booking confirmation.
  - Negative Test Case:
    Input: Invalid event ID or incomplete user details.
    Expected Output: Error message indicating the issue with the booking.

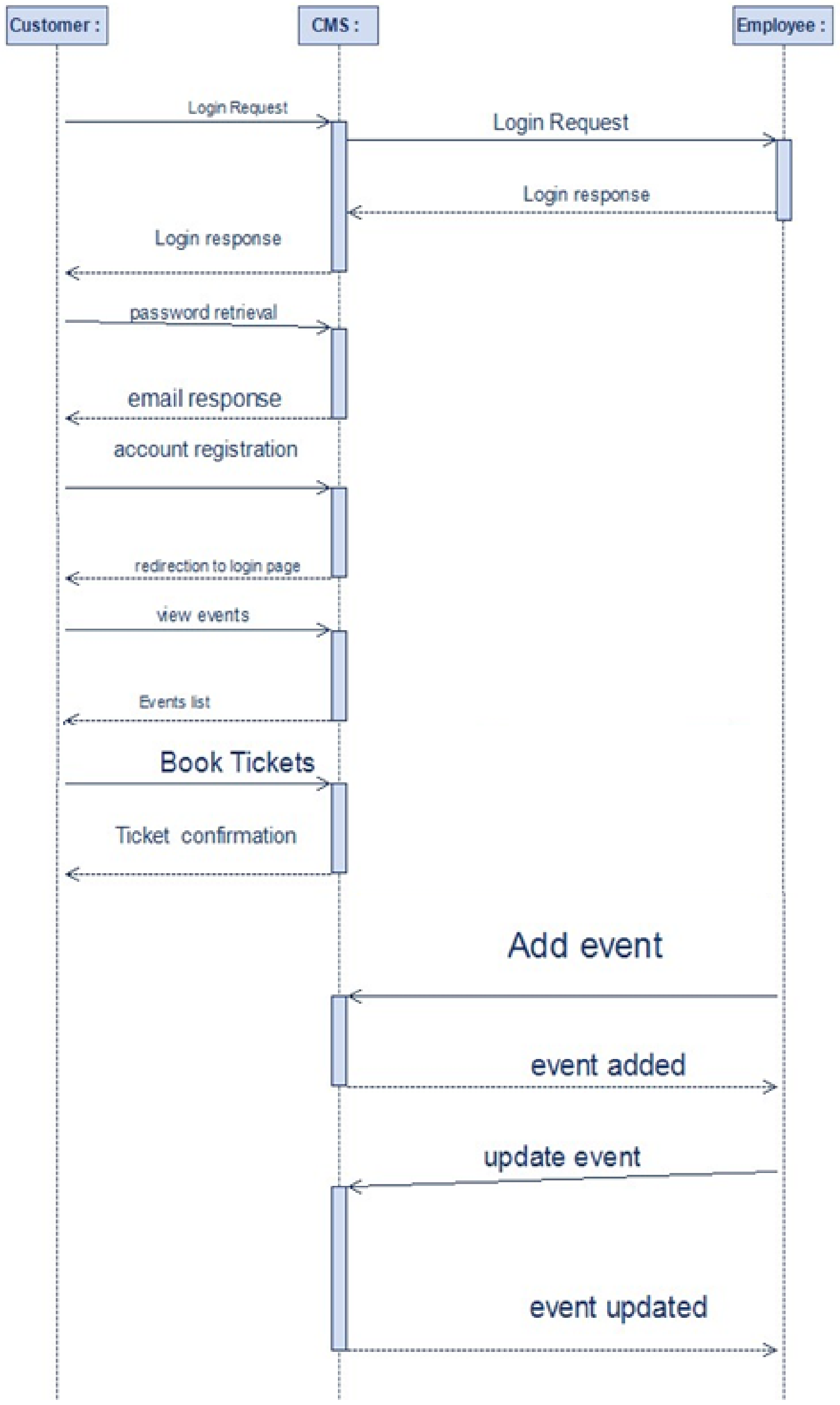
- **User Registration:**
  - Positive Test Case:
    Input: Valid username, email, and password.
    Expected Output: Successful registration message.
  - Negative Test Case:

Input: Existing username or email.
Expected

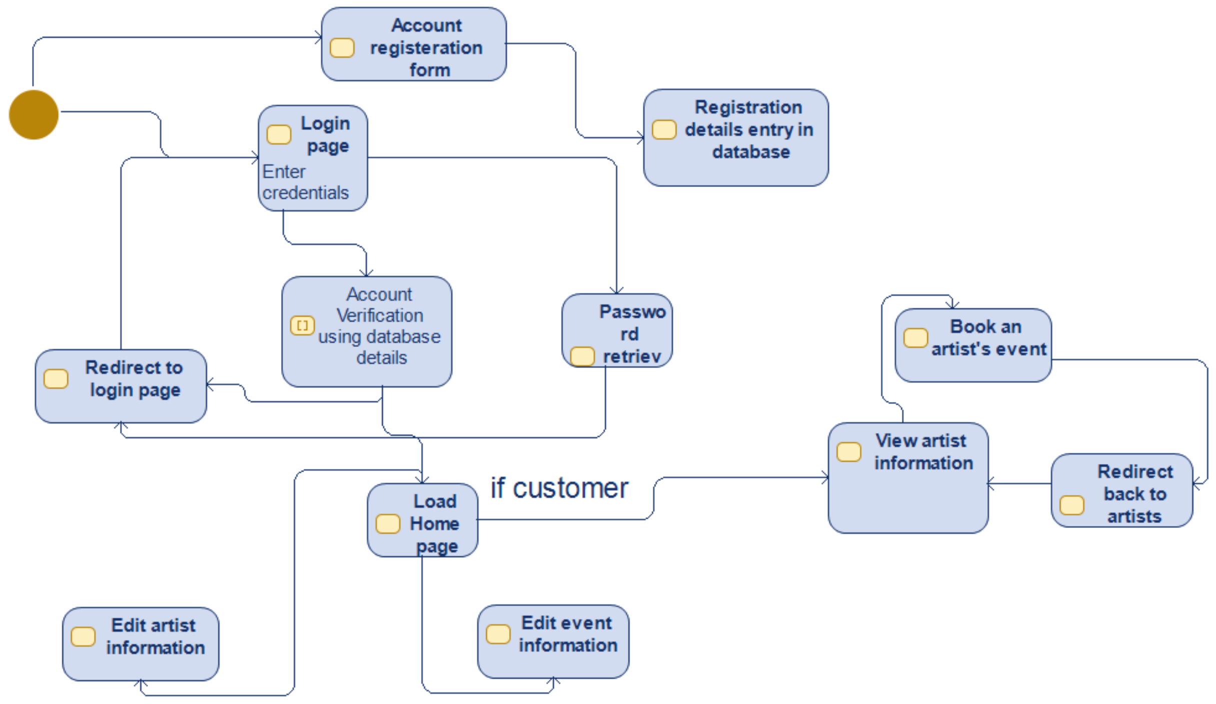
- **Viewing Events from User Account**
    - Positive Test Case:
      Input: User logs into their account and navigates to the "View Events" section.
      Expected Output: Display a list of upcoming events that the user can view and potentially book.
    - Negative Test Case:
      Input: User attempts to view events without logging in.
      Expected Output: Redirect the user to the login page or display an error message prompting them to log in before viewing events.
- **Viewing Booked Events from Admin Account**
    - Positive Test Case:
      Input: Admin logs into their account and navigates to the "View Booked Events" section.
      Expected Output: Display a list of events that have been successfully booked by users, including relevant details such as user information, event details, and booking status.
    - Negative Test Case:
      Input: Admin attempts to view booked events without logging in.
      Expected Output: Redirect the admin to the login page or display an error message prompting them to log in before viewing booked events.

5. ANALYSIS MODELS

5.1 Sequence Diagrams



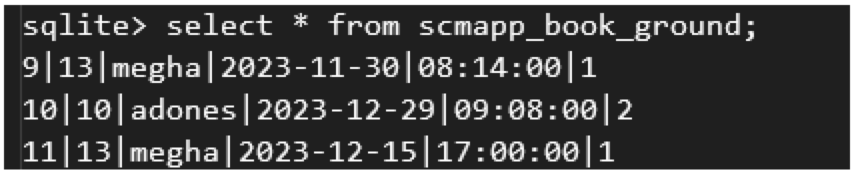
5.2 Data Flow Diagrams (DFD)



6. Databases

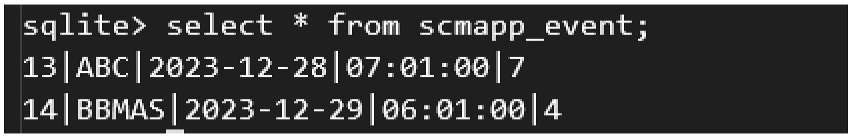
There are 4 main databases implemented using built-in SQLite in this project.

- Client private slot booking
  IDl Event ID l username l date l time l displayed event ID



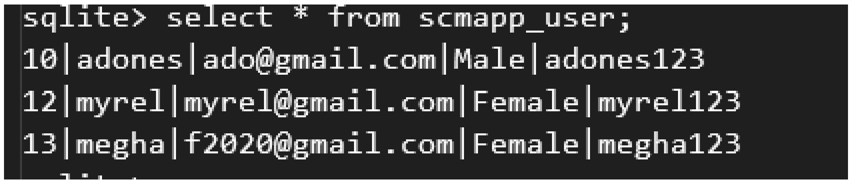
```
sqlite> select * from scmapp_book_ground;
9|13|megha|2023-11-30|08:14:00|1
10|10|adones|2023-12-29|09:08:00|2
11|13|megha|2023-12-15|17:00:00|1
```

- Events table
  Event ID l event l date l time l duration



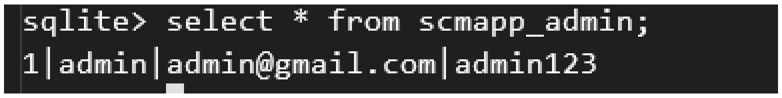
```
sqlite> select * from scmapp_event;
13|ABC|2023-12-28|07:01:00|7
14|BBMAS|2023-12-29|06:01:00|4
```

- Registered users
  ID l username l gmail l gender l password



```
sqlite> select * from scmapp_user;
10|adones|ado@gmail.com|Male|adones123
12|myrel|myrel@gmail.com|Female|myrel123
13|megha|f2020@gmail.com|Female|megha123
```

- Registered admins
  ID | username | gmail | password



```
sqlite> select * from scmapp_admin;
1|admin|admin@gmail.com|admin123
```

7. Change Management Process

The project can be modified by the lead software engineers upon occurrence and detection of any bugs. They are allowed to make the necessary edits in code or entirely remove or replace functions which are found unfit for future utilization of the product. In case of hardware failure, shifting of device or source-code editor is permitted.

A. Appendices

A.1 Appendix 1

1. "SRS Event Management System", Hitesh Srivastava, https://www.academia.edu/38112670/SRS_Event_Management_System
2. https://www.scribd.com/document/500793536/srs-event-management-system

3. https://www.evernote.com/shard/s566/client/snv?isnewsnv=true&noteGuid=a7c73d85-73fd-a0dc-dec1-b79d965245cb&noteKey=wTd7wHIF5wK7IGk6MrOQ2K920xT4Ylqd9YiL_iyuXCNGL3vOWQbqcByGeA&sn=https%3A%2F%2Fwww.evernote.com%2Fshard%2Fs566%2Fsh%2Fa7c73d85-73fd-a0dc-dec1-b79d965245cb%2FwTd7wHIF5wK7IGk6MrOQ2K920xT4Ylqd9YiL_iyuXCNGL3vOWQbqcByGeA&title=Django

4. https://www.youtube.com/watch?v=8hY-tVsD2nQ&ab_channel=PearlEventManagementServices
5. https://visualstudio.microsoft.com/vs/
6. https://www.project.co/project-management-for-events/