SOFTWARE REQUIREMENTS SPECIFICATION

for

Student Auditorium Management System (SAMS)

Prepared by
Aaditya Agrawal (19CS10003)
Debanjan Saha (19CS30014)
Deep Majumder (19CS30015)

March 13, 2021

Contents

Revision History

Name	Date	Reason For Changes	Version

1 Introduction

1.1 Purpose

This document is to define the functional and non-functional requirements of the Student Auditorium Management System (SAMS). This software consists of a backend API server and a frontend application, which together allow for efficiently managing shows, selling and refunding tickets and maintaining a balance sheet.

1.2 Document Conventions

Following defintions and abbreviations are used in this document:

1. SAMS: Students' Auditorium Management System
2. SM: Show Manager
3. SP: Sales Person
4. AC: Accounts Clerk
5. User: User corresponds to either SM or AC or SP or customer.

1.3 Intended Audience and Reading Suggestions

This document mainly focuses on the design analysis and requirement specifications for this software. Software developers, testers, documentation writers are always welcome to read this document. However, it is not necessary for the users of this software to read this document. Nevertheless, we would encourage everyone to look into this document to get to know how we create the software from ground zero.

For the readers, we would encourage you to read the content in the same order in which we have documented this and not to skip in between, so that you get a smoother understanding of this software.

1.4 Project Scope

The software can be used in any auditorium to organise sale of tickets of various shows. This software consists of following functions:

1. Adding new events as per availability of the Auditorium, and editing events which are already present.
2. Allocating Balcony and Ordinary Seats for sale or to offer as complementary gifts. Also fixing the price of different seats.
3. Booking and Cancellation of seats for an event.
4. Printing Ticket for booking and cancellation of a seat of an event.
5. Querying the number of available seats of different classes for an event.
6. Querying the percentage of seats booked for various classes of seats and the amount collected in each case.
7. Booking available seat for a particular show
8. Creating new authorized sales person's and clerk's log in accounts.
9. Recording all the transactions including the sales person ID.
10. Preparing balance sheets for each event and also for the entire year.

1.5 References

1. IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications. IEEE Computer Society, 1998.
2. SE Lectures (Provided by Prof. Partha Pratim Das)

2 Overall Description

2.1 Product Perspective

This software is made to help provide an efficient methods for students and administration to book auditoriums for various events. It is build to add new events as per availability of the Auditorium, and edit events which are already present. The software is flexible and can be extended for various types of seats and other functionalities. It is a software that can be deployed in schools and colleges to reduce errors often caused due to physical selling of tickets. One could book a ticket directly from any place as the software can be hosted on a web server. An user of the software can book seats , cancel already booked seats for an event and get printed ticket for booking and cancellation. User can check number of available and book seats for an event. User can create new authorized sales person's and clerk's log in accounts. It also records all the transactions and prepares balance sheets for each event and also for the entire year. The manager can create new authorized sales person's and clerk's log in accounts.

2.2 Product Functions

The software is intended for the use of four types of users - show manager,accounts clerk,sales persons and customers(spectators). The functions for each type of user provided by this software are listed as follows:

1. **Show Manager**
   a) Add New Show: The SM can add a new show with a date,timings based on the availability of the auditorium. Fixing a particular number of shows on a date and fixing the number and price of the balcony and ordinary seats for sale in a particular show are exclusive rights of SM. Offering seats as complimentary gifts to the students' society and VIPs are also handled by the SM.
   b) Manage Account for SP and AC: The SM is able to create new account for authorised SP and AC and will be able to change passwords for the all SP and AC.
   c) Query: The SM can query the percentage of seats booked of each type and also the amount collected for each show
   d) Check Balance Sheet: The SM can access different types of balance sheets(based on a show or annual sheet) for the auditorium.
2. **Accounts Clerk**

a) Add Expenditure: The AC can add expenditures for a show with a description for the expense.

3. **Sales Person**
   a) Book Ticket: SP will be able to book and authorise tickets for a customer and make a sale for the show.
4. **Customer**
   a) Cancel Tickets: The Customer will be able to cancel their tickets before the starting of the show.
   b) Query for Seats: The Customer will be able to check the availability of different classes of seats.

2.3 User Classes and Characteristics

We have used following classes in our software :

1. Show: Show is a class to handle shows hosted by auditorium. It stores show ID, show date and time, seats list and price of different types of seats.
2. Ticket: Ticket holds information about which show it is for, the type of seat booked, total price and the userId.
3. Transaction: It is a class to store information about flow of money, it stores the details about a transaction of money.It stores the reason for transaction, amount, date and time. It also stores the Id of user involved in the transaction.
4. Expenditure: A show has various expenditures incurred including payment of artists.
5. User: A user has his personal details like UserId, Name, email,etc.

2.4 Operating Environment

We are testing this to work on the following tech stack- JDK11 and above, MongoDB 4.4 and above, Apache Maven 3.6 and above and node 14.5 and above. We have tested it to run on reasonably recent modern browser like Google Chrome 88 and above, firefox 86 and above with amd64 architecture.

2.5 Design and Implementation Constraints

1. Total number of seats in the auditorium must be fixed at any time.
2. Total number of seats booked by a customer can not exceed a beyond a considerable logical value.

3. SM can not be fired. In other words, there must be a SM present in the auditorium to manage the system.

4. Tickets can not be sold by spectators or transfered to others.

2.6 User Documentation

The frontend of the software will be kept sufficiently intuitive and straight forward for an internet literate person to use.

2.7 Assumptions and Dependencies

We have assumed the following:

- Computers are available to all types of users and users are computer literate and internet literate.
- Only one SM is present at a time.

Our dependencies are as follows:

- The software is being developed in linux operating system.

3 External Interface Requirements

3.1 User Interfaces

<Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.>

3.2 Hardware Interfaces

<Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and communication protocols to be used.>

3.3 Software Interfaces

<Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.>

3.4 Communications Interfaces

<Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.>

4 System Features

<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>

4.1 Login

- **Goal in Context**: Open view relevant to the user on giving a correct login credentials.
- **Preconditions**: User should exist with given username.
- **Successful End Condition**: Opens the view corresponding to the user.
- **Failed End Condition**: If password or username isn't correct, it shows a dialog printing the same.
- **Actors**: User
- **Trigger**: The user enters their username and password, and clicks the Login button.

4.2 Create Show

- **Goal in Context**: Create a show
- **Preconditions**: There shouldn't be a show with clashing time
- **Successful End Condition**: A show with given details is created
- **Failed End Condition**: If preconditions are not met, then no show isn't created
- **Actors**: SM
- **Trigger**: In manager dashboard, there is a button for Creating a show.

4.3 Create Accountant

- **Goal in Context**: Create an AC account.
- **Preconditions**: SM should be logged in.
- **Successful End Condition**: An account for AC is created
- **Failed End Condition**: There shouldn't exist an User with clashing username.
- **Actors**: SM
- **Trigger**: In manager dashboard, there is a button for Creating an accountant.

4.4 Create Salesman

- **Goal in Context**: Create a SP account.
- **Preconditions**: SM should be logged in.
- **Successful End Condition**: An account for SP is created
- **Failed End Condition**: There shouldn't exist an User with clashing username.
- **Actors**: SM
- **Trigger**: In manager dashboard, there is a button for Creating a Salesman.

4.5 View Shows

- **Goal in Context**: Display list of all shows.
- **Preconditions**: Shows should be created.
- **Successful End Condition**: List of all shows displayed.
- **Failed End Condition**: No fail conditions.
- **Actors**: SM
- **Trigger**: In manager dashboard, there is a button for Creating a view shows.

4.6 Show Stats

- **Goal in Context**: Show all statistics related to particular shows.
- **Preconditions**: A show should be selected from view shows.
- **Successful End Condition**: Statistics like percentage of seats booked from each category are shown.
- **Failed End Condition**: No fail conditions.
- **Actors**: SM
- **Trigger**: In manager dashboard, there is a button for Showing Statistics of a show already selected.

4.7 Show Balance Sheets

- **Goal in Context**: Show balance sheet of a particular shows.
- **Preconditions**: A show should be selected from view shows.
- **Successful End Condition**: Statistics like percentage of seats booked from each category are shown.
- **Failed End Condition**: No fail conditions.
- **Actors**: SM
- **Trigger**: In manager dashboard, there is a button for Showing Stastics of a show already selected.

4.8 Show Balance Sheets

- **Goal in Context**: Show all statistics related to particular shows.
- **Preconditions**: A show should be selected from view shows.
- **Successful End Condition**: Statistics like percentage of seats booked from each category are shown.
- **Failed End Condition**: No fail conditions.
- **Actors**: SM
- **Trigger**: In manager dashboard, there is a button for Showing Statistics of a show already selected.

4.9 View Accountants

- **Goal in Context**: Show list of all AC
- **Preconditions**: AC should be created.
- **Successful End Condition**: List of ACs is shown.
- **Failed End Condition**: No fail conditions
- **Actors**: SM
- **Trigger**: In manager dashboard, there is a button for viewing accountants.

4.10 View Salespersons

- **Goal in Context**: Display list of all SP.
- **Preconditions**: Account for SP should be created beforehand.
- **Successful End Condition**: List of SP is shown.
- **Failed End Condition**: No fail condition.
- **Actors**: SM
- **Trigger**: In SM dashboard, there is a button to view all SPs present in the system.

4.11 View Yearly Balance Sheet

- **Goal in Context**: Show annual balance sheet for the auditorium.
- **Preconditions**: The year selected should be a valid year in which at least one ticket was sold using SAMS.
- **Successful End Condition**: It will display balance sheet for a particular year.
- **Failed End Condition**: No fail condition.
- **Actors**: SM
- **Trigger**: In SM dashboard, there is a button to view yearly balance sheet.

4.12 Signup

- **Goal in Context**: Create an account for the SM
- **Preconditions**: Only one SM account should be created for managing SAMS.
- **Successful End Condition**: A new account is created for an SM.
- **Failed End Condition**: If one SM already exists, SAMS will not create another account for SM.
- **Actors**: A system admin or the SM.
- **Trigger**: The system admin will enter details of the SM in a Signup Page and press signup button.

4.13 View Shows

- **Goal in Context**: Display a list of all available shows.
- **Preconditions**: There should be at least one available show.
- **Successful End Condition**: List of all available shows is displayed.
- **Failed End Condition**: No fail condition.
- **Actors**: Customer
- **Trigger**: In Customer dashboard, there is a button to view all shows.

4.14 View Tickets

- **Goal in Context**: Show available tickets for a particular show.
- **Preconditions**: There should be an existing show to view tickets for it.
- **Successful End Condition**: Show tickets for selected show.
- **Failed End Condition**: No fail condition.
- **Actors**: Customer
- **Trigger**: In Customer dashboard, there is a button to view tickets for a show.

4.15 Cancel Tickets

- **Goal in Context**: Cancel tickets for a particular show.
- **Preconditions**: At least one ticket should be booked for a show in order to be able to cancel it and it should be done before the show starts.
- **Successful End Condition**: Ticket for selected show gets cancelled with automatic deduction of money based on date of cancellation based on policy of the auditorium.
- **Failed End Condition**: No fail condition.
- **Actors**: Customer
- **Trigger**: In Customer dashboard, there is a button to cancel tickets for a show.

4.16 Book Ticket

- **Goal in Context**: Book a ticket for a customer.
- **Preconditions**: There should be at least one booking request from a customer for a particular show.
- **Successful End Condition**: One or multiple tickets are booked for a customer for a show.
- **Failed End Condition**: No fail condition.
- **Actors**: SP
- **Trigger**: In SP dashboard, there is a button to book tickets for a show for a particular customer.

4.17 Add expenditure

- **Goal in Context**: Add various types of expenditure for a particular show
- **Preconditions**: No precondition.
- **Successful End Condition**: An expenditure with details is added for a show.
- **Failed End Condition**: No fail condition.
- **Actors**: AC
- **Trigger**: In AC dashboard, there is a button to add various expenditures for a show.

5 Other Nonfunctional Requirements

5.1 Performance Requirements

<If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.>

5.2 Safety Requirements

<Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product's design or use. Define any safety certifications that must be satisfied.>

5.3 Security Requirements

<Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements. Refer to any external policies or regulations containing security issues that affect the product. Define any security or privacy certifications that must be satisfied.>

5.4 Software Quality Attributes

<Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.>

5.5 Business Rules

<List any operating principles about the product, such as which individuals or roles can perform which functions under specific circumstances. These are not functional requirements in themselves, but they may imply certain functional requirements to enforce the rules.>

6 Other Requirements

<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>

6.1 Appendix A: Glossary

<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.>

6.2 Appendix B: Analysis Models

<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.>

6.3 Appendix C: To Be Determined List

<Collect a numbered list of the TBD (to be determined) references that remain in the SRS so they can be tracked to closure.>