Test Suite

for

Students' Auditorium Management Software (SAMS)

Version 1.0 approved

Prepared by

Rushil Venkateswar (20CS30045)
Rahul Mandal (20CS30039)
Sourabh Soumyakanta Das (20CS30051)

Indian Institute of Technology, Kharagpur

23rd March 2022

Revision History

Name	Date	Reason For Changes	Version

1 Introduction

This is the detailed document for the test cases that we will be using for testing the software.

2 Unit Testing

This is a process in software development in which the smallest testable parts of a software are tested individually for proper operation.

2.1 White Box Testing

2.1.1 Seat class

- On inputting 17, "Ordinary" in the seat class constructor __init__(), a seat class object is created.
- On calling print(), it prints "Seat Number: 17, Seat Type: Ordinary"
- On calling Allot() with an int input 24 (AllotmentStatus is false and transactionID is empty), it makes the AllotmentStatus true and assigns a transactionID "24".
- On calling Allot() again on the same seat with input other than 24, let's say 30, it returns false since the seat is already alloted a transactionID.
- On calling isAvailable(), it returns false since it is already booked.
- On calling cancel(), it changes AllotmentStatus to false, transactionID to empty string.
- On calling cancel() again, it returns false since the seat isn't booked anymore.
- On calling isAvailable(), it returns true.

2.1.2 Show Class

- On inputting required arguments in __init__() function of show class, it constructs a show object without errors. Inputting dates in different formats as specified will result in an error. For example: (23-03-2022, "ShowName", 80, 20, 200, 450) will create a new show.
- On calling ShowAvailableSeats() function, program will display all the seats with available seats in green color.
- On calling PercentageOccupied() function, program will display the percentage of seats occupied for the given show.

2.1.3 Auditorium Class

- Calling __init__() function for auditorium class will create new object for auditorium class.

- Calling addShow() function will add a new show. If the timing clashes with an already existing show, then it will throw an error.

Example: addShow("NewShow")

- On calling findShow() with argument as show name, program will display that show in results.

Example: findShow("ExistingShow")

2.1.4 Transaction Class

- On inputting appropriate arguments in __init__() function for transaction class, the ticket is booked or cancelled as entered.
- Calling print() will print the transaction receipt of booking or cancellation

2.1.5 Ledger Class

- On calling __init__() function, program will create a new object for ledger class
- On calling printTransaction() function will display all the transaction for the auditorium

Example: printTransaction("ShowName")

- Calling addExpense() function with appropriate arguments will add expense of the provided show in the ledger.

Example: addExpense("ShowName",23-03-2022)

- Calling addRevenue() function with appropriate arguments will add revenue of the provided show to ledger.
  Example: addRevenue("ShowName",23-03-2022)
- Calling printBalanceSheet() function will print the balance sheet for the show provided.

Example: printBalanceSheet("ShowName",23-03-2022)

2.1.6 Employee Class

- On calling __init__() function with correct arguments (i.e., id and password) will create a new employee object.

Example: __init__("id","pass")

2.1.7 Salesperson Class

- On calling __init__() function with appropriate arguments will create a new salesperson class with provided commission.

Example: __init__("id"," pass",5) this will create salesperson object with 5% commission.

- On calling InsertSales() function will create a new sale for that salesperson in transaction array
- On calling getSales() function will print all the sales of salesperson within provided dates.

Example: getSales(20-03-2022, 23-03-2022)

2.2 Black Box Testing

2.2.1 Testing of Login page

- The login page for each type of user is checked, with appropriate error message thrown for wrong password or ID.
- OTP based 2FA is done, but this has not been tested.

2.2.2 Printing Balance Sheet

- This operation should cover all expenses incurred from the beginning.
- In case of many entries in the balance sheet, the software should not crash.

2.2.3 Seat Availability

- The software should correctly display the number of available seats currently across all auditoriums for a given show.
- The UI would display the seats available in green and seats taken in red.

2.2.4 Booking and Printing of Tickets

- The software should generate a ticket having the booking details (seat number, auditorium number, show timing) as well as the name of the person it has been booked for.
- The ticket should be emailed to the user as well, but this will not be tested by us as this functionality is implemented by an external vendor.

2.2.5 Cancellation of Ticket

- Software should allow cancellation of ticket with appropriate changes made to the number of vacant seats and the Ledger of the auditorium.

3 Interface Testing

3.1 Home Page

- There are two options, one for login (intended for employees and manager) and another to display the availability of seats (intended for all).
- Clicking on login option redirects to the login page.
- Clicking on display availability redirects to another page which gives a graphical overview of the seats available based on the show names and timings entered.

3.2 Login Page

- If the credentials are correct, it redirects to the dashboard or login homepage of the respective user (Manager or Salesperson or Accounts Clerk).
- In case incorrect credentials are entered, it shows invalid login and prompts user to enter again.

3.3 Salesperson Menu

- Salesperson searches for the show in the box provided.
- If the program finds any matching shows, then it displays on the screen.

3.4 Book/Cancel Menu

- The sales person selects the seats to be booked/cancelled.
- If the seat was not already booked then it gets booked.
- If the seat was already booked then it gets cancelled.

3.5 Audit Clerk Menu

- The clerk enters the expense name, date and amount of expense.
- The expense gets added to the database and a message of the same is displayed.

3.6 Spectator Menu

- The spectators can search for the shows. All the matchings will be displayed.
- If the box is left empty and searched, then the program returns all the shows scheduled.

3.7 Manager Menu

- Show manager can perform following functions after successful login:

1. Create Salesperson ID
2. View Transactions per salesperson
3. Create new Shows
4. View percentage of seats occupied
5. View full schedule of auditorium
6. Access the balance sheet of auditorium

3.8 View transaction

- Selecting a salesperson from a list
- After selecting, all the transactions performed by salesperson will be visible on screen.

3.9 Auditorium Schedule

- This will show the list of all scheduled shows in the auditorium.

3.10 Add Salesperson Menu

- A unique ID and password along with commission rate is entered.
- After validation is successful, the salesperson is added to the system and program displays the message for the same.

3.11 Add Shows Menu

- Name of the show, dates, timings, seats type and numbers are entered.
- The show gets added to the schedule and a message is displayed for the same.
- If any discrepancies in time and dates occur then a corresponding error message is displayed.

4 System Testing

4.1 Functional Validation Testing

The complete software package will be tested in this section to see if the functions are working as expected cohesively. Additionally, mapping of correct buttons to correct underlying class functions will be checked to see if everything is working as planned.

4.2 Performance Testing

Changes made during the runtime of the website should reflect on the database stored on the disk. Handling of sudden, unexpected crashes should be done such that all changes made to the database right before the crash are reflected in the database.
The software should be able to handle large quantities of user data. If a show has a large number of seats to offer then the software should be able to handle booking, and ticket generation along with database management for all users efficiently.
Calculation of expenditure and updating of ledger along with calculation of the commission for the salespeople for a long period of time should be done quickly and the software should be robust enough to handle it efficiently.

5 Use Case Testing

- This type of testing is covered under Unit and Interface testing.