Students' Auditorium Management Software (SAMS)

TEST PLAN

Apurv Kumar (14CS10006) & Aniket Choudhary (14CS30004)

Contents

Introduction

Objective

This document is a high-level overview defining testing strategy for the Supermarket Automation software. Its objective is to communicate project-wide quality standards and procedures. It portrays a snapshot of the project as of the end of the planning phase. This document will address the different standards that will apply to the unit, integration and system testing of the specified application. Testing criteria under the white box, black box, and system-testing paradigm will be utilized. This paradigm will include, but is not limited to, the testing criteria, methods, and test cases of the overall design. Throughout the testing process the test documentation specifications described in the IEEE Standard 829-1983 for Software Test Documentation will be applied.

Testing Strategy

Testing is the process of analyzing a software item to detect the differences between existing and required conditions and to evaluate the features of the software item. Specific test plan components include :

- Purpose for this level of test,
- Management and technical approach,
- Pass/Fail Criteria,
- Hardware/Software requirements,

Scope

Testing will be performed at several points in the life cycle as the product is constructed. Testing is a very dependent activity. As a result, test planning is a continuous activity performed throughout the system development life cycle. Test plans must be developed for each level of product testing.

Tests

Unit Testing

Unit Testing is done at the source or code level for language-specific programming errors such as bad syntax, logic errors, or to test particular functions or code modules. The unit test cases shall be designed to test the validity of the programs correctness. Unit testing includes testing all the classes in the program and the graphical user interface.

White Box Testing

In white box testing, the UI is bypassed. Inputs and outputs are tested directly at the code level and the results are compared against specifications. This form of testing ignores the function of the program under test and will focus only on its code and the structure of that code. Test cases are generated that not only cause each condition to take on all possible values at least once, but that cause each such condition to be executed at least once.

Black Box Testing

Black box testing typically involves running through every possible input to verify that it results in the right outputs using the software as an end-user would. We have decided to perform Error guessing and Boundary Value Analysis testing on our application.

Interface Testing

There are two primary modules that were needed to be integrated: the Graphic User Interface module and the controller module which connects to the local database. The two components, once integrated form the complete Student's Auditorium Automation Software. The following describes these modules as well as the steps that will need to be taken to achieve complete integration. We will be employing an incremental testing strategy to complete the integration.

Graphic User Interface Testing

This module provides a simple GUI where the user can perform the different actions (functions). This module was tested separate from the backend to check if each interface (e.g. Log in button) is functioning properly, and in general, to test if the mouse-event actions were working properly. The testing was performed by writing a stub for each element in the interface.

Test(Connection Module)

The Controllers provide functions to send requests to the servers and then obtain relevant data from the database to return to the GUI. This module was tested separate from the GUI by printing out the results to the Console. In testing this module we followed the incremental testing method i.e. testing one function first and then keep adding additional function and test it again until all the required functions are tested.

The following interfaces were tested:

- AccountantAdd
- AccountantEdit
- AccountantHome
- AddShows
- SalesPersonAdd
- SalesPersonEdit
- SalesPersonHome
- SeatStatus
- ShowManagerDetails
- ShowManagerFrame
- Start

System Testing

The goals of system testing are to detect faults that can only be exposed by testing the entire integrated system or some major part of it. Generally, system testing is mainly concerned with areas such as performance, security, validation, load/stress, and configuration sensitivity. But in our case we focused only on function validation and performance. And in both cases we used the black-box method of testing.

Function Validation Testing

The integrated "SAMS" was tested based on the requirements to ensure that we built the right application. In doing this test, we tried to find the errors in the inputs and outputs, that is, we tested each function to ensure that it properly implements the parsing procedures, and that the results are expected.
In addition, we tested:

- The interfaces to ensure they are functioning as desired (i.e. check if each interface is behaving as expected, specifically verifying the appropriate action is associated with each mouse click event).
- The interaction between the GUI and the backend controller classes. In this case the data will be inserted and check if they are sent to the server properly and the expected results are obtained.

Performance Testing

This test was conducted to evaluate the fulfillment of a system with specified performance requirements. It was done using black-box testing method. Following things were tested:

- Creating an event which requires large number of seats to see how much time it take to store and retrieve information of that show from the database.
- Booking large number of seats and storing there information to the database to see how much time it takes to retrieve them from the database.
- Calculating the expenditures statistics for a very large history to test the performance of expenditure table generation.

Use Case Testing

The following use cases were verified to cover as many corner cases as possible:

- If user tries to login as a Show Manager, but the show manager account is not available the an error message is displayed and user is asked to create one and is taken to the managerdetails interface.
- If entered username or password does not match with any user's info available in database then error message is displayed.
- If user tries to see event's seat status in start interface without selecting an event or event is not available then error message is displayed.
- If the user tries to login as salesperson or accountant but the respective accounts are not created by show manager then an error message is displayed.
- In change login details interface, if the entry for mobile number is not a number, then error message is displayed.
- In change login details interface, if the entry for email id does not follow a valid email id format, then error message is displayed.
- In the change login details or add salesperson/accountant interface, if the user leaves the username and/or password field as blank, then an error message is popped.
- In the add shows interface if the entered event date and time lies before the current date and time, a invalid event date-time error is displayed.
- If the salesperson selects book ticket button without selecting an event or if no events are created, then an appropriate popup is displayed.
- If accountant selects add expenditure button without selecting an event first, then an error message is displayed.
- If salesperson tries to cancel a ticket of an event whose event date has passed then an appropriate error message is didplayed.
- If the show manager tries to query event's data and statistics like event's sales, event's expenditures, event's status or event's seat status without selecting or creating an event first, then an appropriate message is displayed.
- If the show manager tries to query employee's performance data like total sales, yearly balance sheets etc without creating any employee, transactions or expenditures an error is displayed.

Pass or Fail Criteria

Suspension Criteria

The test is considered suspended if any of the following is encountered:

1. The software crashes,
2. The software produces incorrect output,
3. The software takes more than expected time to produce the output.

Approval Criteria

The test is considered approved if software produces correct output as per the demand of client.