TEST PLAN

STUDENTS AUDITORIUM MANAGEMENT SYSTEM (SAMS)

Prepared by

Aditi Mishra (19CS10015)

Yash Agarwal (19CS10066)

Monal Prasad (19CS30030)

Submitted On

27 March 2021

CONTENTS

INTRODUCTION

This document is a high-level overview defining testing strategy for the Students' Auditorium Management System. 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 con- structed. 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.

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 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.

INTERFACE TESTING

There are two primary modules that need to be integrated: the User Interface module and the System module which connects to the local database. The two components, once integrated form the complete Student's Auditorium Management 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.

User Interface Testing

This module provides a simple GUI where the user can perform the different actions (functions). This module will be tested after the backend testing is completed to check if each interface (like a menu command) is functioning properly, and in general, to test if the interface can correctly access and use/modify the system.

Test (Connection Module)

The Controllers provide functions to send requests to the servers and then obtain rel- evant data from the database to return to the GUI. This module is to be tested separately from the GUI by printing out the results to the Console. In testing this module we followed the big bang integration testing method i.e. testing one class first and then test it again when all the classes have been integrated.t

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 white-box method of testing.

Function Validation Testing

The integrated “SAMS” will be tested based on the requirements to ensure that we built the right application. In doing this test, we will try to find the errors in the inputs and outputs, that is, we will test each function to ensure that it properly implements the parsing procedures, and that the results are expected.

In addition, these features are to be 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 is to be conducted to evaluate the fulfillment of a system with specific perfor- mance requirements. It will be done using white-box testing method. Following things will be tested:

- Creating an event which requires a large number of seats to see how much time it takes to store and retrieve information about that show from the database.
- Booking a large number of seats and storing their 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 balance sheet generation.

USE CASE TESTING

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

- If the entered username or password does not match with any user's info available in the database then an error message is displayed.
- If the user tries to login as a salesperson or accountant but the respective accounts are not created by the show manager then an error message is displayed.
- While adding a salesperson/accountant , 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 the book ticket button without selecting an event or if no events are created, then an appropriate popup is displayed.
- If a salesperson tries to cancel a ticket of an event whose event date has passed then an appropriate error message is displayed.
- If the show manager tries to query event's data and statistics like event's expenditures 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 without selecting an employee, transactions or expenditures an error is displayed.

SOFTWARE RISK ISSUES

- Backup and Recovery of the local login, accounts and auditorium databases, must be carefully checked.
- The ability to restart the application in the middle of the process is a critical factor to application reliability. This is especially true in the case of the database files as it needs to be protected locally.
- Database security and access must be defined and verified, especially for files shared between the Accounts clerk and the Show Manager.

FEATURES NOT TO BE TESTED

The following is a list of the areas that will not be specifically addressed. All testing in these areas will be indirect as a result of other testing efforts.

- Accessing and backing up database files has not been tested at the unit level, because the system testing will bring out any issues related to those modules.

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 the client.

TEST DELIVERABLES

- Acceptance test plan
- System/Integration test plan
- Unit test plans/turnover documentation
- Test logs and turnover reports

ENVIRONMENTAL NEEDS

The following elements are required to support the overall testing effort at all levels within the reassigned sales project:

- Access to the master control tables (databases) for controlling the production/testing environment on both production and development systems.
- Access to the backup/recovery process.

SCHEDULE

Time has been allocated within the project plan for the following testing activities. The specific dates and times for each activity are defined in the project plan timeline. The persons

required for each process are detailed in the project timeline and plan as well.

A. Review of Requirements document by test team personnel (with other team members) and initial creation of Inventory classes, sub-classes and objectives.
B. Development of Master test plan by test manager and test with time allocated for at least two reviews of the plan.
C. Review of the System design document by test team personnel. This will provide the team with a clearer understanding of the application structure and will further define the Inventory classes, sub-classes and objectives.
D. Development of System/Integration and Acceptance test plans.
E. Unit test time within the development process.
F. Time allocated for both System/Integration and Acceptance test processes.