Creighton
UNIVERSITY

System Test Plan

For

Theater Ticket Sales System

by: Devin Chavez and Benjamin Walter

Date: 4-13-2022

Table of Contents

1. Introduction

1.1 Purpose

This document is a test plan for Theater Ticket Sales System Testing, produced by the System Testing/Development team. It describes the testing strategy and approach to testing the team will use to verify that the application meets the established requirements of the business prior to release.

1.2 Objectives

- Meets the requirements and specifications.
- Supports the intended business functions and achieves the required software standards.

2. Functional Scope

Testing the Theater Ticket Sales System is in the scope of this test plan. The following components and functions will be tested:

1. User access to the theater catalog
2. Browsing of the catalog
3. Selection of plays
4. Selection of Seats
5. Adding Seat to Cart
6. Accessing Cart
7. Remove from Cart
8. Purchase Cart
9. Input information for transaction
10. Receive ticket after transaction
11. Availability of seats purchased

3. Overall Strategy and Approach

3.1 Testing Strategy

Theater Ticket Sales System Testing will include testing of all functionalities that are in scope (Refer Functional Scope Section) identified. System testing activities will include the testing of new functionalities, modified functionalities, functionality access, testing of internal & external interfaces.

3.2 System Testing Entrance Criteria

System testing criteria consists of the following:

- Complete or partially testable code
- Requirement are known and observed
- Test Case is defined for test
- Test Environment is established
- Testing is approved by both Project members

3.3 Testing Types

3.3.1 Usability Testing

User interface attributes, cosmetic presentation and content will be tested for accuracy and general usability. The goal of Usability Testing is to ensure that the User Interface is comfortable to use and provides the user with consistent and appropriate access and navigation through the functions of the application (e.g., access keys, consistent tab order, readable fonts etc.)

3.3.2 Functional Testing

The objective of this test is to ensure that each element of the component meets the functional requirements of the business as outlined in the:

- Business / Functional Requirements
- Business rules or conditions

3.4 Suspension Criteria and Resumption Requirements

This section will specify the criteria that will be used to suspend all or a portion of the testing activities on the items associated with this test plan.

3.4.1 Suspension Criteria

Testing will be suspended if the incidents found will not allow further testing of the system/application under-test. If testing is halted, and changes are made to the hardware, software or database, it is up to the Testing/Project Manager to determine whether the test plan will be re-executed, or part of the plan will be re-executed.

3.4.2 Resumption Requirements

Resumption of testing will be possible when the functionality that caused the suspension of testing has been retested successfully.

3.5 Test Data

Test data requirements are drawn up based on the functional requirements that are due for testing. The testing team will identify test cases that can be grouped into test scenarios and detail the data required to complete the testing activities.

4. Execution Plan

4.1 Execution Plan

The execution plan will detail the test cases to be executed. The Execution plan will be put together to ensure that all the requirements are covered. The execution plan will be designed to accommodate some changes if necessary, if testing is incomplete on any day. All the test cases of the projects under test in this release are arranged in a logical order depending upon their inter dependency.

5. Defect Reporting

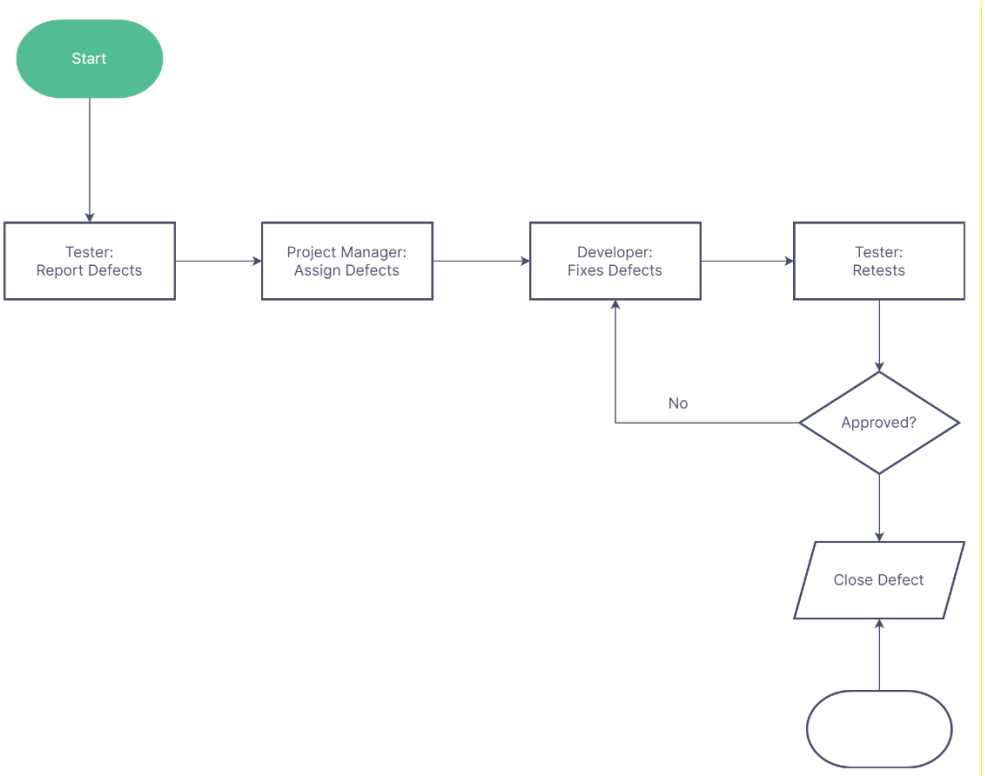
5.1 Defect Tracking

Defect Tracking will be completed by finding the defect by testing the system for each functionality. The defect will then be tracked by the tester that first found the defect and reported so that it can be fixed.

5.2 Defect Reporting and Reports

Defects will be reported until March 8th. Defects will be reported be generated by Devin and Benjamin to be reviewed and analyzed.

5.3 Defect Management Process



5.4 Defect Severity Definitions

Critical	The defect causes a catastrophic or severe error that results in major problems and the functionality rendered is unavailable to the user. A manual procedure cannot be either implemented or a high effort is required to remedy the defect. Examples of a critical defect are as follows: • System abends • Data cannot flow through a business function/lifecycle • Data is corrupted or cannot post to the database
Medium	The defect does not seriously impair system function can be categorized as a medium Defect. A manual procedure requiring medium effort can be implemented to remedy the defect. Examples of a medium defect are as follows: • Form navigation is incorrect • Field labels are not consistent with global terminology
Low	The defect is cosmetic or has little to no impact on system functionality. A manual procedure requiring low effort can be implemented to remedy the defect. Examples of a low defect are as follows: • Repositioning of fields on screens • Text font on reports is incorrect

6. Environment

6.1 Environment

The Test Environment will consist of the following:

- Personal Computers to access system
- Windows and Linux operating systems will be used.
- Git Hub repository
- Browser
- Eclipse

7. Test Schedule

System testing is scheduled for a period of 3 weeks. The test team will complete the execution of all the tests during the first 1 weeks. The defects retesting and regression testing will occur in the last week of System Testing. The run dates for defect retesting period may be changed according to the need to retest and close the defects.

8. Assumptions

We assume that we will find some defect from the User interface. Some defects will also come from the interaction of modules. Specifically, the interaction of the user module and the catalog module. Both the user module and the catalog modules have the most interaction with the rest of the system, so, this is where we will focus most.

9. Risks and Contingencies

A major risk is to fix a defect with the system but cause a bigger system failure. In this case, we will avoid this by having multiple versions of the system. Another risk comes from the fact that the testing team and the development to are the same. This can cause problems with black box testing and biases.