Eastern New Mexico University

Team 4

Garry Callis, Jeru Sanders, Stephen Washington, and Michael Gonzales

Project: Stage 3

Software Test Plan

April 17, 2022

INTRODUCTION

Purpose

a. The purpose of this document is to design a test plan for development and system testing. The client has provided the requirements for the desired software. The goal of this document is to address the requirements the client has specified and to design a test plan that meets those requirements.

Objectives

b. We are developing custom software for the Portales theatre project. This custom software must accommodate a list of requirements provided by the client. Our objective is to have a minimum of one test per each requirement provided. In this document we will discuss the plan to test each individual requirement. The software should meet the standards listed below.

    i. Needs to meet specified client requirements

    ii. Supports all functions needed for the business

    iii. Client must be satisfied with the product

FUNCTIONAL SCOPE

In our project “Theatre Portales” we utilized certain modules that accomplish certain tasks and requirements. Listed below are the modules that have been utilized in our system so far. Each module, both front-end and back-end, will need to be tested against the requirements provided by the client. This will ensure that all requirements within the system are met.

Components to test:

- Front End Development
    - SCREEN_NONE: The start of the application. Does not display anything.
    - SCREEN_LOGIN: Prompts users for username and password or guest login.
    - SCREEN_WAITING_FOR_SERVER: Displays a message while trying to establish connection to server.
    - SCREEN_SHOW_LIST: Displays the list of upcoming shows.
    - SCREEN_SEAT_LIST: Displays the seats for the shows along with their availability and price.
    - SCREEN_PAYMENT_INFO: Displays form for the customer to fill in and submit their payment information for processing.
    - SCREEN_CART: Displays the seats they have selected along with total price.
    - SCREEN_RECEIPT: Displays a confirmation message and receipt of transaction.
    - SCREEN_SHOW_EDITOR: Displays a menu to edit shows attributes.
    - SCREEN_SEAT_EDITOR: Displays a menu to edit seats attributes.
- Back End Development
    - Connection.inc.php: This file connects to the database, is included at the top of every file.
    - DeleteShow.inc.php: This file deletes a specified show from the database.

- DeleteUser.inc.php: This file Deletes a specified user from the database.
- GetShow.inc.php: This file fetches and echos the shows from the database.
- GetShowReport.inc.php: This file fetches information on echo back the sales information.
- LoadSeats.inc.php: This file fetches and displays the seats and their attributes.
- Login.inc.php: This file uses a username and a password to grant users access to the site.
- NewShowInsert.inc.php: This file inserts a new show into the database.
- NewUserInsert.inc.php: This file inserts a new user into the database.
- SelectSeats.inc.php: This file gets the selected seats and changes the availability of the seats.
- Selectshows.inc.php: This file selects the show.
- UserProfile.inc.php: This file has user classes and functions/methods to be called with the class.

OVERALL STRATEGY AND APPROACH

Testing Strategy

As a group we have decided to utilize all necessary stages of testing to ensure our software meets all our clients' needs. To do this, we will perform development, release, and user testing.

Development testing:

This has involved testing functions as we write them, making sure that we are getting the expected result. Development testing will be a continuous process as we are developing and editing our system.

Integration testing:

We are currently developing the front and back end of custom software. Therefore, we have been performing development testing as we are building the software. To be more specific we are performing and will continue to perform integration testing. This involves thoroughly testing how the different systems work with each other. For example: Create a user, sign in, and delete the user.

Release Testing:

After our system is completed, we will then perform release testing. This will involve testing our entire system. This stage will guarantee that our system is working smoothly as a whole and hopefully reduce the number of problems with user testing.

In our final stage of testing, we will perform user testing. Prior to creating our system, we were given a list of requirements by the client. This testing will be performed by the user or client, using the individual environment. This will guarantee that the software meets all requirements needed to work sufficiently.

Testing Criteria

To test the system, we need to address all system requirements. As previously mentioned, the purpose of this project is to create software for the theatre “Los Portales.” Listed below are the requirements for the project. Refer to SDD document in Stage 2 for more information.

- System:
    - 8 rows and 12 column seats
    - Seats must all fit on the screen nicely
    - Can’t be too small to touch
    - Leave some border on the end
    - All buttons must be visible
    - Can’t cover buy button at the bottom
    - Available seats are green
    - Unavailable seats are red
    - Seating for future plays
    - Online and accessible with mobile devices
        - Screen rotation
        - Touch buttons
        - Virtual keyboard
        - Screen size
- Admin:
    - Add new plays (time and date)
    - They can also edit and delete plays
    - Change individual seat prices
    - By selecting one or more seats and changing them

    - Convenient to change all the seats at once to the same price
    - Can generate a report with
        - How many seats were sold for a specific play and date
        - Requires username and password to access the management area.

- Customers:
    - Be able to create an account with an email address and password.
    - Be able to edit their information (ex: name, age, address, telephone, and email)
    - Registered customers can buy available seats for any play
    - Can select and deselect multiple seats
    - Can add a list of selected seats to a shopping cart
    - Can move from play to play to buy more seats
- Checkout:
    - Total due
    - Report on transactions
    - Transaction ID (consecutive number)
    - Name of play or plays
    - Dates/Time
    - Seat numbers
    - Customer needs to make changes to shopping cart
    - The system must ask the customer for a Creditcard number and simulates the sale.
    - System must inform user of the status of the sale
    - An option to the customer, the system must send an email with purchase details (Optional)

Testing Types

- Development testing
    - Unit testing:
        - The client has an internal unit testing suit to test its connection to the server and its API contract.
        - These unit tests will help inform the design of the backend systems
        - The following is a list of tests
            - User profiles:
                - Create a user
                - Log in as that user, verify that login is successful
                - Delete the user
                - Verify that login fails since the user is deleted
            - Show management:
                - Create a show
                - Change that show's date to exactly 24hrs in the future
                - Get the shows from the server
                - Verify that the newly created show is on the list
                - Moving the show's date to the past
                - Get the shows and verify that the show is no longer on the list
                - Delete the show
            - Seats
                - Confirm available seats are green prior
                - Click a seat, once clicked the seat should be yellow
                - Proceed to cart, then proceed to select another seat

    - Unavailable seats should be red.
- Purchase
    - Enter payment information
    - Verify if it works
    - Should display a successful popup if correct

Usability testing

This type of testing will cover the user interface mainly. This type of testing will focus on the cosmetics of the system. There are certain things that we need to keep in mind when performing this type of testing.

- Ease of use
- Cosmetically pleasing
- Overall experience
- Performance

To test the usability of the system, we will first go over the system with the software development team. Making sure that the interface is easy to use, cosmetically pleasing, and our experience using the software is good. Listed below are the steps needed to perform our usability test.

Development Testing

- When writing the code during development, each change or addition is tested individually during implementation, developers should try to test all reasonably testable and common edges cases and explicitly write code to handle error conditions inline (no exceptions)
- During the development stage developers should also try to test all external behaviors and component interactions when making changes to coupled components. They will be able to

make changes and fix bugs in the interface quickly. This is possible since the developers will have the most information about the code.

Release Testing

- Our second step will be to test the system with a third party. These individuals may or may not have software development experience. These testers will help uncover and diagnose user interface and experience issues. At the end of their purchasing experience, they will grade the system on ease of use, cosmetic appearance, and overall experience. As well as giving critique and recommendations on how to improve these areas if they believe improvements need to be made. This will provide the software development team with unbiased test results.
- Third party testers will use both the desktop and mobile apps to create an account and make a purchase. This will provide information as to whether the mobile and desktop visual experience is adequate and expose any discontinuities between the platforms experience.
- After the results from the third-party usability testing are completed, the software development team will make the changes necessary to improve the system user experience and aesthetics. Then the software development team will perform regression testing, guaranteeing that the changes made are working correctly and that the system still performs the same functionality.

Acceptance Testing

- The client will run the system software utilizing the final online app.
- They will test the creation and editing of shows.
- They will also test the report and give comments on the format.

REQUIREMENT EXECUTION PLAN

System Display Requirements		
Requirements	Type (due date)	Testing Phase
8 rows and 12 column seats	Visual Inspection	1
Buttons must all fit on all screens nicely with padding and not overlapping	Visual Inspection Layout Testing Mobile testing	1 2 3
Available, selected, and taken seats are color coded (green, yellow, red)	Visual Inspection	1-4
Mobile devices are supported	Visual Inspection Mobile testing	2 2
Touch screen works for buttons	Visual Inspection Mobile testing	2 2
Virtual keyboard and input text works	Visual Inspection Mobile testing	3 3
Screen rotation works	Visual Inspection Mobile testing	3 3
Shows seating for future plays	Visual Inspection	1
Customer Requirements		
Users can create accounts with an email address and password	Integration testing Unit testing	3-4 4
Users can log in to their accounts	Visual inspection Integration testing Unit testing	2 2 4
Customers can edit their profile details (name, age, address, tele, email)	Integration testing Unit testing	3-4 4
Customers can select and deselect seats. You can add a list of seats to the cart.	Visual inspection Integration testing	1 1
Customers can browse and add tickets to the cart for checkout	Visual inspection Integration testing Unit testing	1 3 4
Customers can add tickets to multiple plays to the cart and check them out in one transaction	Integration testing Unit testing	3 3-4
Users can edit tickets and view the current total in the cart	Visual inspection Integration testing	3-4 3-4
Users see the tickets and shows they purchase tickets for after checkout	Visual inspection Integration testing	2-4 4
Admin		
Can create, edit, and delete plays	Integration testing Unit testing	4 4
Can set the prices of seats individually and by selecting multiple seats	Integration testing Unit testing	4 4
Can generate a report of purchases	Integration testing	3-4

Checkout		
The system requires customers to put in credit card information and simulate a sale. Informing user of the status of the sale (ex) purchase is successful).	Visual inspection Integration testing Unit testing	1 2-3 4
Generate a receipt/report of purchase	Visual inspection Integration testing	1 2-4
Receipt/report displays correct information (transaction ID, Name of play, Date/Time, Seat numbers)	Visual inspection Integration testing	1 2-4
Optional email with receipt information (Ask professor for to clarify).	Integration testing Unit testing	4 4

Phase 1: Initial planning and requirement understanding phase

Phase 2: Design specification phase

Phase 3: Risk analysis and testing phase

Phase 4: Code and demo phase

DEFECT REPORTING

The purpose of this section is to explain the log levels used to report errors, messages, and other information from the program to developers for use in debugging. All these messages are purely cosmetic; they are used both in debug and release mode except logVerbose(), which is disabled in release mode.

- Logging
  - The client application logs data for several reasons
    - Information data that may be relevant in the future, like build date/version
    - Noncritical issues like performance warnings
    - Critical issues like http and server error
    - And fatal issues like malformed server responses
  - To facilitate this, we will have a logger module that will have the following functions
    - logVerbose() for logging verbose information like startup, ui element creation, screen transitions, server message send/receive
    - logPrint() for actively debugging using print messages for custom programming information
    - logWarning() for casual warnings like performance messages
    - logPanic() for potentially fatal errors
  - Server logging
    - We plan to implement server logging. This is accomplished by using a simple php file that takes a string argument and writes it to a dedicated table for development telemetry information.

ENVIRONMENT

Client testing environments

- Code is merged and versioned on Github.
- It cloned and worked on locally.
- You can simulate an Apache server using XAMPP.
    - You can use the PHP files, but the php files cannot access the SQL server (unless you were to configure it yourself locally).
    - We use development tricks like adding a shortcut login button to the login screen to bypass interaction with the server for cosmetic changes.
- When doing integration testing, we connect with our webhost's ftp server (000webhost.com) using FileZilla.
    - We only need to update the client.js (and the assets folder every now and then) to test a new live version.
- We use Chrome in incognito mode to avoid the cache preserving our files between changes.
- We rely on the JavaScript console in Chrome to display syntax and semantic errors.
- We sometimes use the Chrome JavaScript debugger to step through portions of particularly confusing code.

Backend testing and working environment

- Backend testing is conducted using a local WAMP server before applying to webhost.
- We are using the object-oriented method of coding the PHP files. This makes our codes modular and easier to troubleshoot.
- Connection testing was tested on the webhost server.
- PHP files have results, error messages, and success messages verify execution.
    - var_dump() is used with SQL variable to verify proper SQL commands using variables sent from the user. Var_dump() is commented out or removed after tests pass.
- We use PHPmyadmin to verify changes have been made to the database through our php codes.
- After local testing, codes are applied to the webhost to be tested with frontend, by the front-end team.

TEST SCHEDULE

Since our system is still under development, we will commence development testing first. This will be completed in phases to make sure we fulfil all requirements provided by the client. Our final system draft will be completed by May 24th

Phase	Date to Start	Date to Complete
1	Jan 30th	Feb 27th
2	Feb 27th	March 27th
3	March 27th	April 17th
4	April 17th	May 11th

May 11th is the due date for the project. We will be expected to submit our final software system. Therefore, system testing will commence on the 25th of April and be completed on the 29th of April. After we receive the results and feedback from the testing, we will fix errors, bugs, and changes needed. Regression testing will be completed after corrections have been made to the system. Regression testing will be completed on the 4th of May. Acceptance testing will be the final stage of testing. This will commence on the 11th of May. This is when the system is to be presented and tested by the client.

ASSUMPTIONS

Assumptions and support of platforms:

- Web browsers
    - Microsoft Windows, Mac OS, Linux
    - Android, iOS
    - Device must have attached keyboard, or popup virtual keyboard upon html input .focus() call.
    - Device display size must be at least 1270x480
    - Aspect ratio (TBD)
    - No performance requirements (TBD)

RISKS AND CONTINGENCIES

- 000webhost.com will often fail to let us access the ftp server; it could potentially go down. This would be disastrous to our data (loss of all data).
    - Client development is backed up on a local drive, Github, and Dropbox.
    - Server development files are backed up on a local drive and Github
    - https://github.com/stephen601/CS-472-Software-Engineering-Project/tree/main/code
- Pixi js has no input text/virtual keyboard feature, date picker, or scrollable region, we will have to create those ourselves, hopefully they are good enough.
- There is a risk that Pixi js updated, and it breaks the client, because right now we embedded in a link to it.
    - We should download it at some point.
- There is a risk JavaScript or Chrome/Webkit/Firefox changes.
    - We are defending against this by writing simple JavaScript using minimal features.