Group Members

Software quality, change management and testing

Case Study Assignment

1. **Introduction**:

This document provides detailed quality assurance and test plan of the software investment Management System (IMS) which is aimed toward a person who has considerable number of investments in stock market and banks, and so needs software assistance for book keeping and computations regarding the investments. The Test Plan is designed to present the scope, approach, strategies, time estimation, and required resources of all testing activities of the software. Based on the real-time use cases of the software, this document provides all necessary testing tools and environment setup description and guideline.

* 1. **Objective**:

With detailed testing guideline, this document should provide the quality assurance standards and plan. Which ensures the software’s performance in real life scenario as well as in user circumstances. Also to provide the testing environment with time and resource estimation, risk management testing tools etc. in order to make sure that the system can run smoothly in different platforms like windows, UNIX etc. Also to evaluate its user friendliness, accessibility, reliability, capabilities and performance under high pressure.

* 1. **Scope**:

This document is to provide quality assurance guideline of the entire scope of the system in terms of every functionalities, features and use cases as well as its usage in different real life scenarios from user perspectives. The main scope of the software is given below:

a. Managing investment of a single user, which would include maintaining bookkeeping information about entities like Portfolio, Security, and Transaction.

b. Computation of Net-Worth and Rate of Investment (ROI) of the Investor.

c. Giving alerts to the user, if he requests for one.

d. Downloading the current prices of shares from the web. e. User authentication.

The testing plan is designed with the whole scenario in real life use. So that the whole software life cycle is covered. Such as testing a user’s activity like maintaining, storing, viewing personal investment information (e.g. Portfolio, Security, and Transactions etc.), testing Computations with different numbers and combinations, testing user alerts or notifications for different circumstances, downloading information, user authentication, and so on.

**2 Test Plan**:

**2.1 Overview** This section will cover all the testing steps for the given scope. The subsections will describe the testing strategies, Resource requirement for the testing, testing environment, time estimation, and a few sample test cases accordingly.

**2.2 Strategies**:

*2.2.1 Manual Testing*: To test the whole system as a user. Every features and functionalities should be tried and tested from a user’s perspective. Need to check the whole functionality flow and test different possible outcomes as well. A sample manual testing cycle would be as follow.

|  |  |  |
| --- | --- | --- |
| Step | Success Case | Sample Failed Case |
| 1. Download the software and installation. | Successfully downloaded and installed in any platform. | Failed to install due to some internal issue such as dependency not found |
| 1. Register/login | User authorization or authenticated successfully with username, email and password. | User authorization or authentication failed with username, email and password |
| 1. Create a portfolio | Portfolio successfully created and stored. | Error in creating portfolio |
| 1. Edit a portfolio | Portfolio successfully edited and stored. | Error in editing and saving a portfolio. |
| 1. View Portfolio list | All portfolio under the logged in user shown in a list. | Error in showing the full list. |
| 1. And all other features should be tested like this. | Successfully done as expected with proper user alert. | Any kind of response that is not expected or not understandable. |

Table I: Test steps for manual testing

Each failed test case should be noted down and report as a system bug. Also need to check the bug is responsible for any further bugs in the system.

*2.2.2 Unit Testing*: Testing each functionality in separated testing environment to determine its working properly when every other functionality is not integrated. If failed cases found, it is easier to fix the bug as it is modular based testing. Unit testing is very important to assure each features are working properly. For an example, if we want to unit test the transaction module, we need to test for every possible input and output for every operations like creating, editing, updating and deleting transactions individually such and try to observe the modules response. If anything happens other that the user’s expectation, should be reported as a bug.

*2.2.3 System and Integration Testing*: After the unit testing is done, an integrated testing should be performed with multiple modules and features. Usually this testing is done with correlated modules. In our system, user authentication, portfolio, and transaction module are related and dependent on each other. So, we can test them in integration testing. Such as a user should not be able to see other user’s portfolio. In that case, user role and authentication is working in the back end and portfolio viewing functionality is also related. A portfolio can contain a few number of transactions as well. So all three functionalities are tested in one go.

*2.2.4 Automation Testing*: Automation testing is required to test every corner cases that is involved and may occur in the rarest moment. There are a few automation tools available such as selenium and many more. These tools automatically create test cases with different combinations of inputs and generates test report.

*2.2.5 Performance and load Testing*: After feature and module testing is done, it is needed to assure how much pressure or load can the system can handle. This testing will prove that how well is the database and backend functionalities are designed and handled. The performance of incoming requests and outgoing responses will reflect the overall performance. The database should be able to take the load of huge amount of data. The algorithm and queries to fetch and show data should be optimized that they don’t require a lot of computational resources.

*2.2.6 User Acceptance Testing*: The purpose of the acceptance test is to test the operational use of the system. Comparing the system to its initial requirements will provide the user’s acceptance. If all features are aligned with software requirement specifications, it should be good. Otherwise, differences should be noted and fixed.

*2.2.7 Beta Testing*: After all the functionalities are tested and fixed, a beta build version should be released for limited amount of users so that the real-life operational use can be tested with real users.

**2.3 Resource Requirement**:

*2.3.1 Hardware Requirement:* The software should be stand-alone and not dependent on any other software or system. So, the system is required to be installed and run smoothly on different platforms like UNIX and Windows where no other software can intervene the performance of the system. So at two different machines are required to perform the testing on UNIX and Windows platform.

*2.*3.2: Tester: At least two software testing engineer required to test and report the bugs found in the system.

**2.4 Time Estimation:**

The system has so many features and modules, also the testing plan contains different strategies and steps. So the time required depends and varies in each segments of the testing. Time estimation of testing the whole system is as follows in Table 2. This timeline is for one Software Test Engineer who will be testing on one platform. That’s why two Test Engineers were selected for the task.

|  |  |  |  |
| --- | --- | --- | --- |
| Strategy | Feature/Module/Functionality | Estimated Work Days | Report to |
| Manual Testing | Whole system (Full Cycle testing) | 5 | Project Manager |
| Unit Testing | Single (individually) | 8 | Module developer |
| Integrated Testing | Paired modules (2/3 at once) | 5 | Module developer |
| Automation Testing | Single, Paired | 3 | Module developer |
| Performance Testing | Whole system (on different volume and scale) | 4 | Project Manager |
| User acceptance Testing | Whole system (Requirement check with initial documents) | 2 | Project Manager |
| Beta Testing | Whole system (as build file) | 5 | Project Manager |
|  | **Total Estimated word days for testing** | **32** |  |

Table 2: Time work days estimation for test plan

**2.5 Test Cases**:

In this section, a few test cases are demonstrated along with expected behavior of the system and possible failed cases.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case No. | Action (s) | Description | Module | Expected Behavior | Possible Failed Scenario |
| 1. | a. Run the installation file.  b. Agree with the terms and conditions.  c. Wait until installation wizard shows the installation result | User install the software in their machine. | Installation | a. Run the installation wizard, shows all terms and conditions  b. System wizard shows installation progress through a progress bar.  c. If the installation is completed, system shows a success message | a. System fails to show the wizard.  b. Progress bar got stuck due to any internal error.  c. No success message after completing installation. |
| 2. | a. Run the software.  b. Enter Email/ Username.  c. Enter Password.  d. Click Login. | User logs into the system. | Authentication | a. Run the software without any error  b. Email/ Username field inputs correctly  c. Password field inputs password without showing the characters.  d. It the credentials are correct, System logs in the user, otherwise shows wrong credentials error message. | a. couldn’t run the software due to any internal reasons.  b. Don’t take username/email, password input properly.  c. Login button doesn’t work.  d. System can’t log in the user successfully even after credentials are correct.  e. No error message in case of wrong credentials |
| 3. | a. Click add button.  b. Fill up all input fields.  c. Click save button.  d. Click edit button.  e. Update input fields.  f. Click save button.  g. Click delete button.  h. Click yes button. | User creates, updates and deletes  Their own portfolio | Portfolio | a. System shows add new form  b. All fields takes inputs without error  c. Successfully store the new entry.  d. Shows edit form when edit button is clicked  e. All fields takes inputs without error  f. Successfully store the entry.  g. Shows a popup saying “Are you sure?”  h. If user clicks yes, deletes the entry, if clicks no, pop up should be gone. | a. Add button doesn’t work.  b. Input fields don’t take user inputs.  c. Failed to store new entry.  d. Edit button doesn’t work. Form doesn’t show up.  e. Confirmation popup not coming when user clicks delete button. |
| 4. | a. Click add button.  b. Fill up all input fields.  c. Click save button.  d. Click edit button.  e. Update input fields.  f. Click save button.  g. Click delete button.  h. Click yes button. | User creates, updates and deletes their own transaction | Transaction | a. System shows add new form  b. All fields takes inputs without error  c. Successfully store the new entry.  d. Shows edit form when edit button is clicked  e. All fields takes inputs without error  f. Successfully store the entry.  g. Shows a popup saying “Are you sure?”  h. If user clicks yes, deletes the entry, if clicks no, pop up should be gone. | a. Add button doesn’t work.  b. Input fields don’t take user inputs.  c. Failed to store new entry.  d. Edit button doesn’t work. Form doesn’t show up.  e. Confirmation popup not coming when user clicks delete button. |
| 5. | a. Click add button.  b. Fill up all input fields.  c. Click save button.  d. Click edit button.  e. Update input fields.  f. Click save button.  g. Click delete button.  h. Click yes button. | User creates, updates and deletes their own security | Security | a. System shows add new form  b. All fields takes inputs without error  c. Successfully store the new entry.  d. Shows edit form when edit button is clicked  e. All fields takes inputs without error  f. Successfully store the entry.  g. Shows a popup saying “Are you sure?”  h. If user clicks yes, deletes the entry, if clicks no, pop up should be gone. | a. Add button doesn’t work.  b. Input fields don’t take user inputs.  c. Failed to store new entry.  d. Edit button doesn’t work. Form doesn’t show up.  e. Confirmation popup not coming when user clicks delete button. |
| 6. | a. Click portfolio list  b. Click traction list | System shows portfolio and transaction data according to the security | Display Data | a. System shows all data in a list based on the security settings and user authentication. |  |
| 7. | a. Click compute ROI  b. Click net-worth | System computes ROI and net-worth | Computation | a. Computes and shows the result without any mathematical errors. | a. Mathematical error in computation. |
| 8. | a. Click view share list  b. Click edit on specific row from the list  c. Fill up input fields  d. Click saves | System shows current share price, User can edit share | Share | a. System shows the list of share by fetching data from outer source.  b. Clicking edit button should redirect to update page with edit form of that specific share item.  c. System takes all input correctly  d. Stores the information correctly and shows success message. | a. System fails to get share information.  b. List is not showing.  c. Redirects to different share edit page.  d. Failed to store updated inputs. |
| 9. | a. Click set notification  b. Set preferences and save.  c. Click delete button.  d. Click yes button. | User can set and delete alert for themselves, System should show the alerts | Alerts | a. System shows notification settings and preference form.  b. System stores notification information.  c. Shows a popup saying “Are you sure?”  d. If user clicks yes, deletes the entry, if clicks no, pop up should be gone.  e. System shows notification when it is set. | a. Notification form doesn’t show up.  b. Notification can’t be set.  c. Notification can’t be deleted.  d. Confirmation popup not coming when user clicks delete button.  e. System doesn’t show notification as saved preferences. |

**3 Project completion Checklist**

1. Fix bugs and give feedback to the testing report.
2. Revisit project scope, requirements and implemented feature list to ensure everything is done and completed.
3. Sign off the user acceptance agreements with the stakeholders
4. Finalize the software production reports with final product features, functionalities, use case and test results.
5. Prepare release notes, terms and conditions and software required privileges.
6. Conduct Project closure meeting with team members and stakeholders.