Course name: Software Quality Assurance and Testing

Member Name and ID:

1. **Fahad Molla -18-37548-1**
2. **Nusrat Songita Khan – 18-37518-1**

Section: B

Contribution:

1. **Fahad Molla-50%**
2. **Nusrat Songita Khan-50%**

Date of submission: 22-09-20

Test Plan for ABC University Administer System

Introduction:

Our software firm is going to develop a system to administer ABC University. Now we create a document which is called Test Plan to give a complete planning of systematic strategy for the testing to develop the system. We create this document to describes the objectives, scope, approach, resources, schedule and focus of software testing activities. In the test plan we also describe the features that should be tested and the risks associated with the test strategy. People of quality assurance department means testers are the primary users of this document. Programmers as well as the management people are also in the target group of the document.

Test Items:

A test item is a software item that is the application under test. In this section we will provide a list of all those component that has been identified as test items.

The following components are the test item:

1. Source code: In the source code the following modules will be tested.

- Student Database
- Employees DB
- Registration Module
- Accountancy Module
- Academic Departments Module
- Library Module
- Exam Control Module

2. SRS: Software requirements specification is the basis for our entire project. It lays the framework that every team involved in development will follow. The SRS will help us to ensure requirements are fulfilled. And it can also help us to make decisions about our product's lifecycle — for instance, when to retire a feature. However, through inspection we have to find out problem from SRS.

3. Design document

Features to be tested:

Library Module:

1. Add books: New books can be added
2. Remove books: Books can be removed

3. Update books information: Can update books information
4. Add journals: : New journals can be added
5. Remove journals: Journals can be removed
6. Update journals information: : Can update journals information
7. Search books: Can search book by book id, book name, or by book author name
8. Search journals: : Can search journal by journal id, journal name, or by journal author name
9. Issue books: Required books can be issued
10. Return books: Return the book without fine before the last date, or with a late fine after the last date
11. Issue journals: Required Journals can be issued
12. Return journals: Return the journal without fine before the last date, or with a late fine after the last
13. Add library members: New library members can be added
14. Remove library members: Library members can be removed
15. Auto generating id number: New books, journals, library members can have a auto generating id number
16. Sorting books: Books can be sorted by order.
17. Sorting journals: journals can be sorted by order.
18. Placing book order: This is where a student or any registered person can place an order for the books they're looking for
19. Reminder: one will get a reminder of when to return a particular book
20. Fine calculator: If the person fails to return the book even after the gentle reminder, can check the issue and the last day of return and can calculate the fine.

Strategies for White Box Testing:

Unit Testing:

Definition	In unit testing, individual program units, such as a procedures, functions, methods, or classes are tested in isolation. It is the first level of testing. Unit testing is used to develop each module. In static unit testing the code will be reviewed by us. Formal inspection comparing with written SRS documentation will be done by us to find any program unit defect. Then the programmer of test team will test the dynamic unit testing.
Participants	The programmer who will writes the program unit will perform the Unit testing as he or she will be intimately familiar with the internal details of the unit. Programmer do unit testing to be satisfied that the unit works as expected. All programmers are accountable for the quality of their own work. This work may include both new code and modifications to the existing code.

Methodology	We will perform the unit testing prior to integration testing. We will conduct it in two complementary phases. **Static unit testing**: in static unit testing formal code review means inspection will be done in the following way: Readiness: we will ensure the code unit is ready for inspection. Preparation: the reviewer will read the code and understand the code, its operation and organization before the review meeting. Examination: Firstly we will make a presentation on the procedural logic of the code, major computational paths and the dependency of units on other units. Then code will be read line by line and reviewer will make general suggestion on fixing defects. After that, the final decision will be taken how defects can be fixed. Rework**:** A summary of the meeting will be created by the record keeper at the end of the meeting. Validation: then a validation process will be occurred which will involves Checking the modified code and ensuring that the suggested improvement have been implement correctly. Then the revised final version will be distributed to all of the group members. Exit: the review process will be completed when the following action will be done: • Every line of code in the unit has been inspected. • If too many defects are found in a module, the module is once again reviewed after corrections are applied by the author. As a rule of thumb, if more than 5% of the total lines of code are thought to be contentious, then a second review is scheduled. • The author and the reviewers reach a consensus that when corrections have been applied the code will be potentially free of defects. • All the CRs are documented and validated. The follow-up actions are documented. • A summary report of the meeting including the CRs is distributed to all the members of the review group. **Dynamic unit testing**: First the programmer of the test team will write test driver and stubs then they will perform the dynamic testing following way: • Execute every line of code. This is desirable because the programmer needs to know what happens when a line of code is executed. In the

	absence of such basic observations, surprises at a later stage can be expensive. • Execute every predicate in the unit to evaluate them to true and false separately. • Observe that the unit performs its intended function and ensure that it contains no known errors.

Integration testing:

Definition	Integration Testing is a software testing where Two or more individual units are combined and tested to verify if they are working as expected. The unit-tested modules which work individually may arise problems when they are combined because of rampant interface errors. So to ensure that unit-tested modules operate correctly when they are combined together as dictated by the design we will do Integration testing. We also do integration testing to construct a reasonably stable system that can withstand the rigor of system-level testing. Some common approaches to performing system integration are : • Incremental Approach • Top Down Approach • Bottom Up Approach • Sandwich and Big Bang Approach
Participants	Integration test engineers and software developers will jointly perform Integration testing.
Methodology	To starting integration, testing we will not need to wait until all the modules are completed and unit tested. We can start it as soon as the relevant modules are available. We will use Bottom up approach to perform Integration testing. In bottom up approach no time is wasted waiting for all modules to be developed unlike Big-bang approach and it easy to find fault. Moreover, we need not to give a early proto type. So as we have only 10 members in our test team we choose bottom up approach. As we choose bottom up approach we will start system integration with the integration o lowest level modules. In this approach, to integrate a set of lower level modules at first we will construct a test driver module which will invokes the modules to be integrated. When the integration

	of a desired group of lower level modules will be satisfied the driver is replaced with the actual module and one more test driver is used to integrate more modules with the set of modules already integrated. Until all the modules will be integrated the process of bottom-up integration will continue.

Regression testing:

Definition	Regression testing is a software testing which confirm that a new code or program change has not affected the existing functionalities. So we will do regression testing whenever a component of the system is modified.
Participants	Regression testing will be performed traditionally by the software quality assurance team after the development team has completed work.
Methodology	It is mainly selection of already executed test cases which are re-executed to ensure existing functionalities work fine. When our test team will find any defect on program after any other testing, they will send that to the developer team. Then the developer will find the problem and will fix that. After fixing the bug the unit or system need to be retested that means the test team will retest that system or unit. That retest will be done by regression testing after every testing.

Strategies for Black Box Testing:

System testing:

Definition	System testing is the testing which tests a complete and fully integrated software product. It presumes that all the component of the software have been previously successfully integrated. It test the overall system as a whole from the customer's perspective. The main priority is that the software system work as a whole system in the customer environment. The main focus is that to verify and fulfill the customer requirement.
Participants	In our system, this testing will be done by independent professional testers.

Methodology	System testing is done after integration testing is completed. During the system testing we will monitor the processes of test execution and defect fixing as it is important to do. We identify two key categories of metrics, to be able to monitor those test processes.They are (i) system test execution status and (ii) defects status. We will provide a detailed defect schema, which will includes a general FSM model of defects for ease of collecting those metrics. Analyzing defects is a crucial activity which will be performed by the software development team while fixing defects.To do system testing in our system we will apply Pareto methodology, defect analysis technique. It's Principle is"Concentrate on the vital few and not the trivial many". An alternative expression of the principle is to state that 80% of the problems can be fixed with 20% of the effort. AKA "80-20 rule" This principle will guide us in efficiently utilizing the effort & resources. We will provide three metrics, namely, defect removal efficiency, spoilage, and fault seeding, to measure test effectiveness. The objective is to evaluate the effectiveness of our system testing effort in the development of a product in terms of the number of defects escaped from the system testing effort.We will do a beta testing at the customer site during the system testing.We will provide a framework for beta testing and will discuss how beta testing will be conducted at the customer's site. Moreover, we will provide a detailed structure of the system test execution report, which will be generated before a product's general availability is declared.

Acceptance testing:

Definition	Acceptance testing is a technique which determine if the project met the customer requirement specifications. The main purpose of this testing is to check and evaluate the system and to verify the system met the end user requirements. The main concept of acceptance testing is not finding defect of a system. It is related to system's quality.
Participants	Acceptance test execution is an important activity. We will perform our system's acceptance testing by the customer with much support from the developers.
Methodology	We will divide our acceptance test cases into two subgroups. The first subgroup will consists of basic test cases, and the second will consists of test cases that are more complex to execute. The acceptance tests will be executed in two phases. In the first phase, the test cases from the basic test group will be executed. If the test results will be satisfactory, then the second phase, in which the complex test cases will be executed, will be taken up. In addition to the basic test cases, a subset of the system-level test cases will

	be executed by the acceptance test engineers to independently confirm the test results .We will randomly select 5–10 test cases from each test category. If a very large fraction ,will say more than 0.95, of the basic test cases pass, then the second phase can proceed. It may be counterproductive to carry out the execution of the more complex tests if a significant fraction of the basic tests fails. System-level test personnel from the development organization will travel to the customer location where the acceptance tests will be conducted. They will assist the customer in preparing a test site and will train the acceptance test engineers on how to use the system. They will provide the earlier system-level test results to the customer's test engineers in order to make informal decisions about the direction and focus of the acceptance testing effort. In addition, the on-site system test engineers will answer the customer's questions about the system and will assist the acceptance test engineers in executing the acceptance tests. Any defect will be encountered during acceptance testing will be reported to the software development organization through the on-site system test engineers. The defects will be submitted through the defect tracking system. The software build will be retested by the supplier and a satisfactory software image will be made available to the customer for continuation of acceptance testing when the defects will be fixed. The failed tests will be repeated after the system will be upgraded with a new software image. An agreement must be reached between the on-site system test engineers and the acceptance test engineers when to accept a new software image for acceptance testing. The number of times the system can be upgraded to a new software image during acceptance testing will be negotiated between the customer and the supplier. Multiple failures of a system during acceptance testing will be an indication of poor system quality. It will be possible that an acceptance test engineer may encounter issues related to acceptance criteria during the execution of acceptance test cases. The system may not provide services to the users as described in the acceptance criteria. Any deviation from the acceptance criteria will discover at this stage may not be fixed immediately. The acceptance test engineer may create an acceptance criteria change (ACC) document to communicate the deficiency in the acceptance criteria to the supplier. A representative format of an ACC document. An ACC report is generally given to the supplier's marketing department through the on-site system test engineers.

Possible Risks:

No.	Risk	Mitigation
1.	Programmer could not finish a module for testing unit for testing in time	Start the testing of other modules
2.	Team member lack of knowledge of particular testing	Plan a course to build that skill or knowledge.
3.	Test manager lack of management skill	Plan management training for manager.
4.	Tight project schedule	Give the core modules first priority.
5.	Cost of the testing exceed	Test the core modules first priority.
6.	Lack of teamwork	Inspire team member and good behavior.
7.	Team member drop from the middle of the project	Requite new member

Team organization and distribution of the work:

Member Role	**Works**
1 Test Manager	• Responsible to manage the whole project. • Give project technical direction. • Create management report.
2 Tester or System Tester	• Execute the tests, Log result. • Recover from error. • Document defects
2 Test Designer	• Create test plan and test suite, • Evaluate effectiveness of test effort
2 Developer in test team	• Implement the test cases • test program, test suite etc.
1 Test System Administrator	• Builds up and ensures Test Environment and assets are managed and maintained, • Install / manage worker access to test systems

	• Support Tester to use the test environment for test execution
1 Database Administrator	• Ensures test data (database) environment and assets are managed and maintained. • Administer test data (database)
1 SQA member	• Take in charge of quality assurance • Check to confirm whether the testing process is meeting specified requirements

Schedule:

NO.	Task	Duration	Start Date	End date
1	Requirement understanding and analysis	14 days	24th-September-2020	7th-October-2020
2	Review meeting	1 day	8th-October-2020	8th-October-2020
3	Create test Cases	15 days	9th-October-2020	23th-October-2020
4	Create test Scenarios	15 days	23th-October-2020	6th-November-2020
5	Creating test Suites	15 days	7th-November-2020	21th-November-2020
6	Review meeting	1 day	22th-November-2020	22th-November-2020
7	Create Test driver and stubs	16 days	23th-November-2020	8th-December-2020
8	Dynamic unit testing	20 days	8th-December-2020	28th-December-2020
9	Regression testing	4 days	29th-December-2020	1st-January-2021
10	Integration testing	16 days	2nd-January-2021	17th-January-2021
11	Regression testing	4 days	18th-January-2021	21th-January-2021
12	System testing	16 days	22th-January-2021	6th-February-2021
13	Regression testing	4 days	7th-February-2021	10th-February-2021
14	Acceptance testing	19 days	11th-February-2021	1st-March-2021
15	Final review meeting	1 days	2nd-March-2021	2nd-March-2021

Conclusion:

We have create our test plan by analyzing our system's features and functionalities. In our test plan we have identified the scope of our testing and the features and functionalities to be tested. Then we have developed a test strategy which is composed of several testing like unit testing, integration testing, regression testing, system testing and acceptance testing techniques. After that we have anticipated risks and suggested risk responses. Finally we have identified the testing tasks and break them down into activities and then assigned to our team members. This test plan is the guide to our testing process throughout the development. It directs our testing approach and describes the testing practices to followed. It help us to confront the challenges which are waiting for us and make us focus on the important areas. It also answer all the questions of us which may arise during our testing process. Besides all these benefits, there are few negative points as well. To make the test plan we have to spend a lot of time and effort. However these test plan serves as a roadmap to our testing process that has all the necessary details related to our process.