Software Engineering
Assignment - 2

Date: 23-11-21

Name: Abhishek Aditya B.S.
SRN: PESIUG19CS019
Section: A

Problem statement chosen: Scenario #2, XYZ a leading global mobile application servile-provider for utility payment.

A.
Before diving into planning the test strategy, one must ensure the content, project/product is well understood and any discrepancies between the testing teams should be settled. The project understanding involves reviewing the possible use case scenarios of the product, discussions with the upstream and downstream partners, playing around with the product, performing deep down inspection and walk through of the product. Since it will be infeasible to test all the test case scenario combinations, one must prioritize the test cases based on customer requirement, schedule, budget, skills and talent of the test team. There should be a clear understanding of what is "in scope" and "out of scope" of the testing. Once this is done test Adequacy criteria should be planned to determine when to stop testing or consider testing to be completed. In the view of planning effective and efficient test strategy the following must be followed:

1. The testing mindset on the test models that will be followed:

- Demonstration model / mindset:

To make sure the mobile application for payments runs well and doesn't have any issues in the newer domains of banking and financial transactions, travel bookings and loans. Ensuring the application works for both iOS and Android platforms as well as for other various kinds of mobiles and their configurations in the

Indian market by performing regression testing, and making sure the testing will be done to find out any bugs or issues which cannot be easily detected.

- Evaluation model / mindset:
To find faults throughout the application development life cycle and making a log of the error measures and metrics throughout. Since cost of fixing errors goes up as we move down the phases in the lifecycle, any faults in requirements and design phase should be detected early by analysis and review techniques.

- Destruction model / mindset:
Try to make the application fail many times, to find as many faults, since good test cases are those which finds faults in the product and decide when to stop testing based on a measure like number of faults or issues left in the system or fault density

- Preventive model / mindset:
Fixing the issues that happen in early phases of the application lifecycle can drive the cost down, by reviewing and following an effective test driven development.

2. Testing types which will be followed:

- Unit testing: Test for coding / construction error before the application goes for quality assurance test. Tests the smallest individually executable code units done by programmers. The test cases could be for code logic / algorithms, boundary conditions, interfaces, error handling etc. It brings out any issues in the application which might have happened during the implementation.

- Integration testing: Test to verify the interfaces and connectivity between various components in the application, can find problems like latency issues and resource contention problems which are not detectable by unit testing.

can be done either by top down or bottom up approach by integrating the unit tested components until the application works as a system. It brings out any issues which might have happened in the detailed design phase of the lifecycle.

- System Testing: Tests a completely integrated system to verify that its following all the specified requirements. It involves testing the end-to-end flow of an application or the software as a user, for validating if the end features work fine, with the test focused towards the whole application. It brings out any issues which might have happened in the requirements and architecture phase. Some of the tests which could be done are:

» Smoke and Sanity testing - To make sure the most important functionality works and to decide if the application is fit for further testing.

» Regression testing - To make sure all kinds of mobiles and operating systems in Indian market and overseas are compatible with the application. These tests ensure any changes made to the application doesn't have any side effects.

» Functional and Non Functional testing - to test the feature/functionality covering all the scenarios including failure paths and boundary cases. Non-functional testing looks to verify the attributes such as performance or robustness of the application etc.

» usability testing - to determine the ease of use from an end user's perspective. How easily the application can be used by the end users, level of skill required to learn/use the application, time required to get used to in using the application.

» Localisation testing - performed to verify the quality of the application localisation for a particular target culture/locale.

» Load testing - carried out to determine the behaviour of the application or to check the robustness under varying load.

» Platform or cross platform tests: To make sure the application runs on various mobile operating system like Android or iOS and various mobile hardware configurations.

» Security Testing: Testing to uncover vulnerabilities of the application and determine that its data and resources are protected from possible intruders.

» Compliance Testing: to determine the compliance of the application with internal or external standards.

» Scalibility Testing: Carried out to check the performance of the application in terms of its capabilty to scale up or scale down the number of user request load.

- Acceptance Testing: involves running a suite of tests on the whole system. There are high level tests to verify the completenes of the user story and requirements. It includes both business logic tests as well as UI validation elements. It acts as the final quality gateway and brings out any issues which might have happened by not fulfilling a particular requirement and also showcases if any requirements were left out during the feasibility phase.

- Alpha & Beta Testing: Before rolling out the application for the end users the application must be tested amongst a small group of potential users either in-house (alpha) or for external user (Beta).

3. Test Environment that will need to be available:
It mentions the minimum hardware requirements that will be used to test the application. setting up a right test environment ensures success of software testing else it could result in delay, cost escalations and incorrect conclusions. For the given mobile application test bed could include:

- Test cases which include simulations of payments from newer domains like banking and financial transactions, travel bookings and loans.
- Servers having mobile operating systems like Android & iOS simulators and emulators to test the mobile application.
- Servers in use should have decent minimum configurations in terms of hardware and software to run the application and to handle varying stress and load tests.
- Actual physical mobile hardware in case a particular hardware or software mobile operating system could not be emulated or simulated to carry out testing.
- Testing the front end UI to determine the ease of use of the application and to ensure there will be no problems in payments happening from the front end.
- Testing the backend router, API calls to test how secure is the application, so that the payment gateway is secure and not exposed to intruders/hackers.
- Testing the ACID (Atomicity + consistency + Isolation + Durability) properties of the backend database to ensure all the payments are done in a consistent way and making sure there is no discrepancies in the payments made.
- Testing the ROLLBACK feature if the payment has happened but the backend could not verify the payment or some component in the application crashed so the payment made is refunded to the correct user.
- Testing how much network bandwidth is needed for the payments to be completed and test cases for the application under varying network conditions to test coverage of all types of users and situations with high/low speed network connectivity.

4. Automation Strategy:

Since periodic testing and running is to be done, planning a good automation Strategy is of utmost importance to the company. The mobile industry is changing rapidly with new hardware and software configurations coming out to the market frequently, so an automation framework should be set up to perform regression testing and platform/cross-platform testing. Since manual testing can be slow, tedious and hard to get test coverage, the newer test cases should be generated and run without human assistance so that a wide range of functionalities and non functionalities can be tested in a shorter amount of time.

The application should be compatible with the automation tool selected and the testing team should be comfortable to use it or else full effective use of the tool will not happen. Tools like Selenium, Calabash, QF-Test, Robust framework etc. can be used. Since these frameworks are fast and repeatable, the company can make sure they are not behind their competitors by quickly developing new test cases and adapting to newer versions of configurations introduced in the market. Maintaining the automation framework and test suite, keeping it up to date will ensure effective and efficient testing.

5. Risk identification for the Strategy, analysis, contingency planning and trigger for the risk:

Risk is the probability of an unwanted incident during or towards testing. Any changes in logic or plans, evolving technology or competitors direction should be done in a controlled manner. Changes made towards improving the Quality of the product should be taken care of. If some of the test modes or testing could not be carried out, alternate options should be looked and used to mitigate the issues in the application.

Any issues in the automation framework or the test environment should be handled by trained personnel. If any issues are found by the end users or during the testing the first step will be to find out from which component is the defect being generated from, next careful assessment and mitigation planning needs to be done in collaboration with developers and other upstream and downstream partners. Once the risk assessment plan is made, trigger should be identified so that in future if similar type of issues pop up, the testing team will know where in the application is the issue happening. Once all this is done the new mitigation plan developed is executed and monitored.

6. Process Improvement suggested

Creating a detailed test schedule with WBS and estimating the effort of tasks in terms of man hours and other metrics can drive the test team towards building an effective test strategy with a list of deliverables which includes test specifications for each of the modules of the application and test cases for different conditions planned. Proper planning and allocation of resources to ensure that the schedule factors in the characteristics of the planned resources.

Project milestones/checkpoints should be identified considering the deliverables expected, the schedule and commitments if any to the end users. These milestones are also used to identify any risk triggers and to help kick in the mitigation plans. Effective communication between the upstream and down stream partners can also boost the project development and testing. Further testing team should be organized with roles and responsibilities assigned to each member. The roles can be test director, test automation manager, test analyst, test development engineers, and Software test engineers.

B. Test Cases

Test Case #1

Test Case ID : TC1

Title : Successful payments on Android version 8 and above, iOS version 10 and above.

Description : User should be able to use the application and make payments successfully on any mobile with android v8 and above, iOS v10 and above. There should not be any incompatibility issues and all the features should work in the intented way. UI should be the same on all compatible android and iOS version and not tailored and designed for a particular version.

Precondition : The user must be verified in the application by pure criteria's, must have legitimate source of payment to initate the payments, legitimate bank account or other forms of verified money wallets to receive money from the application.

Assumption : User Mobile operating system is running android version 8 & above, iOS version 10 & above, with minimum hardware requirements and good connectivity to internet.

Test Steps :

- Check the current running android version, iOS version in mobile device or simulator under test.
- If android version >= 8, iOS version >= 10, check the connectivity and run the application.
- If any issues found in the above step, log the issue and take actions to mitigate or prevent such issues.
- Make sure the payment sources to send and receive have been verified and are legitimate.
- Next add a source of payment and a source to receive the payments which have been verified.
- Make real or simulated payments either small or big from the portals of newer domains like banking & financial transactions, travel bookings, loans.
- Verify the payments from start to end till the payments are received in the correct source for receiving payments.

Expected Results:
Show "Successful Transaction" message with other details like transaction ID, Sender receiver account numbers or wallet ID with an option to download the successful Transaction page for future refernce.
Comments: Additional test cases can be performed to verify if the payments reach with a paset time interval threshold like 2-5 seconds.

Test Case #2
Test Case ID: TC2
Title: Succesful payments on mobile devices with network bandwidth of 500kbps and above.
Description: Since a user cannot have good network connectivity at all times and situations, test designed to make sure he will be able to make a successful transaction.
Precondition: The verified user must be connected to the internet either by cellular network or any type of local area network.
Assumption: User mobile is running android version >= 8 and iOS version >= 10 with a minimum network connectivity 500 kbps.
Test Steps:
- Check the network connectivity speeds on the mobile device or any simulator, emulator and check if there is any firewall enabled which prevents from connecting to some of the portals needed for payment.
- If network connectivity speed <= 500kbps alert the user saying transaction cannot be made, if connectivity is lost display message cannot connect to Internet, any firewall related issue should be alerted to the user. Any other issues, should be logged & taken care of.
- once source of payment and destination is verified, Simulate the payments or make real payments big or small from newer domains.
- Test to make sure if the ROLLBACK feature of the transactions work without any kinds of issues in case the network connectivity drops below the minimum speed of 500kbps or if the connectivity is lost.

- Simulate network congestion situation, and if transactions are taking longer then Rollback the payment else continue with the transaction.
- Test if the payments are sent and received at correct specified source within a time interval threhold without latency.

Expected Results: show a "succesful Transaction" message with other details like transaction ID, Sender receiver account numbers or wallet ID, with an option to download the transaction page for future reference.

comments: Additional test cases to test how much longer will the transaction take to complete if network speed < 500kbps and can it be in acceptable limits.

c. Measures and Metrics planned for use:

The Measures which can be used are:

- Fault Density - Ratio of no. of faults found to the size of program, (bugs/LOC). Higher density represents lower quality.
- Defect Leakage - Indicates test efficiency. (Total no. of Defects found after testing / Total no. of Defects before testing) x100
- MTBF - Mean time between failure in hours. Based on statistical analysis that indicate the probability of failure.

The Metrics which can be used are:

- Requirement Compliance Factor - using the traceability matrix, the RCF measures the coverage provided by a test case to one set of requirements.
- Defect Discovery Rate - Number of defects found per line of code (LOC)
- Cost of Quality - Total cost of prevention, appraisal, rework, failure, to the total cost of the project.