Project Test Plan
MotoMoto

The New Panelists
10/27/2021

Blake Del Rey
Isabel Guzman
Jacob Sunia
James Austin Jr.
Naeun Yu

Team Leader: James Austin Jr.

Table of Contents

1. Introduction

1.1 Purpose of this Document?

The purpose of the Test Plan document will provide the procedure and expectations for evaluation, performance, security, and validation for the MotoMoto application developed by the New Panelists. This document will be used to explain the plot of how testing will be done, scope of testing, testing objectives, approach, and how performance would be tested.

1.2 Testing Objectives

The main objective of the testing that will be done for this application is to ensure that our application functions as intended for users, engineers, and any other entity that interacts with MotoMoto in any way. In order to accomplish this objective, testing measures will be put in place to ensure that our features are all completed exactly as promised to the client, the application can support a large number of users at the same time and does not entirely break when that number is surpassed, is fast and responsive enough to not taint the user experience, and secure enough for users to trust that their data will not be extorted by any entity when given to us.

1.3 Scope

The current scope of the test plan will be to ensure that the GUI testing and component testing of the software is completed as per requirements provided by our client. Testing will be done to ensure that all new functions, existing functions, UI/UX, security, and core components are all tested to ensure that the system is operational.

The main goal of ensuring that all functionalities are mainly user input errors with our objective of achieving this goal is through mostly automated testing on our software and manual testing for usability. With the help of automated programs, this will make it easier to detect and resolve application vulnerabilities in a faster amount of time.

1.4 Features to Be Tested

The following list of features will be the subject of testing for the application. Some features are dependent on each other and will be used in separate testing

1.4.1 System UI

- Login and Registration
- User Accounts
- Car Builder

- Community Board and Community Board: Main Feed
- Part Flagging
- Event List
- Part Price Analysis
- Notification System
- Email Client

1.4.2 System Speed

- Because the application is being developed as a single-page application the page speed only needs to be tested for a single page and this page is the application home page.

1.4.3 System Security/Software Vulnerabilities

- The whole application will be tested as user security will be a primary goal of maintaining a report with our users. Since we plan on using a secured server all core components and dependent features will be tested.

1.5 Features Not Tested

2. Testing Methodology

MotoMoto will be conducting both functional and non-functional testing of the application. The goal of this section is to explain how our team manages testing and explains in more detail of the features tested. All testing besides security testing will take place on a local host to ensure that released material will not be used. Our main goal will be to ensure usability, availability, and reliability tests are conducted with 99.999% passing marks.

2.1 Unit Testing

Unit testing will be used to test each unit, most likely each function, within the MotoMoto application. This automated testing will ensure that a function in a class satisfies its intention and performs the designed behaviors. This testing helps to find any bugs and flaws within the application in the earlier phase of development, which will ease the process of integration testing. Unit testing will be performed on major features and any necessary features that the end-users interact with.

Sub-Features that will be Tested

- Login/Register page:
  - Validate that all user inputs have valid usernames and passwords that are stored within our RDBMS.

    - We will test password lengths where each password must be at least 8 ASCII characters and will have a maximum of 15 ASCII characters.
    - Testing that usernames entered have no other characters other than letters or underscores.
        - Spaces and dashes will not be allowed in usernames/passwords.
    - System will validate if a entered
- User Accounts:
    - User profile pictures will be tested so that images will not exceed 1 MB during .jpeg conversion.
    - Profile descriptions will be tested to ensure that the user can have an empty description or can type up to 500 characters.
- Community Board and Community Board: Main Feed
    - Testing will be done on the number of community board post characters before allowing user to post to the board (limit to 1-1500 characters)
    - Picture uploads will be tested to match our conversion criteria (up to 5 images per post and the size of each image doesn't exceed 1 MB during .jpeg conversion)
- Car Builder
    - VIN testing will be conducted as all entries must be 17 characters.
        - We will test to ensure that an error is occurred if a user does enter an invalid VIN
- Part Flagging
    - Validate the full compatibility of a part, if the part is compatible with the car.
        - Since all OEM parts are recognized by a specific vehicle, parts will be tested to evaluate recognizable parts to a vehicle that they were manufactured for.
        - Some aftermarket manufactured parts will also be recognized by a specific vehicle so, these parts will be tested to ensure virtually that a specific part is compatible.
- Event List
    - The zip code field will be tested as each entered number must be 5 digits long and will be tested to assure that no invalid zip codes are entered.
    - Checks the number of event post characters before allowing user to post to the board (limit to 1-2500 characters)

2.1.1 Testing Metrics

- Unit testing will be measured based on a pass or fail criteria on whether an independent feature of MotoMoto can work on its own.

- Since we will be developing the software to run these tests we will ensure that all potential errors will be recorded in external notes to ensure that each test that fails will be a task to work on for the following sprint

2.1.2 Risks of Unit Testing

- The risks of doing unit testing will be refactorization of the software architecture
    - This can become a problem if Software Design is not up to par and if the system needs an overhaul based on lack of proper structure within the application.
- Not writing correct unit tests can be poor testing which we should be testing based on the behavior of the application instead of how the code is written.

2.2 Integration Testing

Integration testing will be used to ensure that all individual components of the application are able to work in unison. Testing all subsystems within the application will be a key to ensuring both functionalities of the application and communication between microservices. Using an incremental testing strategy during integration testing will allow for a more iterative process to detect any errors and dependency defects.

- Incremental Integration Testing
    - Integrating modules one at a time, this method is more beneficial for the development team to detect any errors within the integration process for the immediate fix.
    - It tests as a group of modules to ensure smooth integration and data flow between modules. The testing will be used after the completion of unit testing.
    - Testing will be done on each dependent module in our application
        - Each dependent module will include the software in total as many of the features included in 1.4.1 will be included in integration testing functionalities
        - Each one of these tests will go through a functionality test to determine pass or fail states to help during code-review stages
- The testing will be done using internally generated source code.

2.2.1 Integration Testing Metrics

- Integration testing will be judged based on a pass/fail criteria of if a component/module is having any technical issues or vulnerabilities attached to it.
- To Enter Integration testing, unit testing must be completed first where information about the design of the integration will be noted
- Multiple of tests will be completed before a task is created to gain full report of what issues are present during the testing phase

- If all tests are passed, then the next iterative phase of implementation can take place.

2.2.2 Risks of Integration Testing

- Since the goal of integration testing is to validate components that can work together, there exists no real risks other than not writing complete tests. The lack of tests that are created can become a problem before deployment if this phase is not taken seriously throughout the SDLC which can be the cause of future implementation issues.

2.3 Performance Testing

The main objective of MotoMoto's performance testing is ensuring that no feature within our application holds back the performance of any other feature within our application or the application as a whole. Satisfactory application performance is imperative to ensuring a quality user experience which is why our performance testing will have multiple tests to ensure the application's performance is acceptable. These tests will be performed every Sunday at 1:00AM PST:

- Load Testing
    - Load testing is used to determine how many users MotoMoto can concurrently support on our application.
    - The following scenarios will be load tested. The format for each test will be in the format (duration in minutes, number of users)
        - (2, 100)
        - (2, 500)
        - (2, 1000)
        - (2, 10000)
        - (2, 100000)
    - If there is reason to suspect exceptionally high traffic within the following week we will also test the scenario (2, 1000000) to ensure our application does not crash under extremely high demand.
    - In order for these scenarios to be considered passed there must be 0 interrupted requests.
- Stress Testing
    - After figuring out MotoMoto's maximum user threshold through load testing, we will then use stress testing to determine our application's behavior when more users are concurrently using our application than supported.
    - From the user capacity determined from load testing we will use 5 incremental test cases to stress test with. If our capacity is represented by N then the following test cases will be used:
        - (2, N + 1000)

            - (2, N + 5000)
            - (2, N + 10000)
            - (2, N + 100000)
- General Performance Testing
    - Acting as a speed test, general performance testing is meant to provide an overall picture of how fast the webpage performs. The scores from performance testing is based on the following factors:
        - First Contentful Paint: The amount of time it takes from the beginning of the page loading to when the first piece of the page's content is rendered in the window.
            - Acceptable metric is between [0,1800ms]
        - First Input Delay: The amount of time when a user interacts with an element on the DOM, to when the page actually processes that event.
            - Acceptable metric is between [0,100ms]
        - Largest Contentful Paint: The amount of time it takes to render the biggest image or text block visible to the user on page load.
            - Acceptable metric is between [0, 2500ms]
        - Cumulative Layout Shift: The amount of time the largest set of element shifts take to occur.
            - Acceptable metric is between [0, 100ms]

2.4 Security Testing

Security testing is one of MotoMoto's main priorities as our plans are to ensure that our automated system is able to identify all weaknesses and loopholes that can cause information leaks. With the use of automated security testing, our main goal will be to ensure that threats, vulnerabilities, and risks are all uncovered to prevent attacks from potential malicious intruders.

Our goal will be to assign both penetration testing and vulnerability testing to multiple software engineers within our team and will be tested during new feature implementations which can happen during sprints) and on every Wednesday of each week. Having frequent security testing will ensure that we are taking an iterative approach to understanding what vulnerabilities are within our system.

2.4.1 Vulnerability Scans

During scans, we want to identify all internal and external problems within our software to understand if a vulnerability will be a security liability. We ultimately plan on using an open source automated tool to help verify the applications network and internal system vulnerabilities. These vulnerabilities include binary analysis, cross site scripting, SQL injection attacks, and platform vulnerabilities.

Using vulnerability scanning externally within our software will help determine whether MotoMoto is exposed/prone to attacks on our web server, relational database management system and application. This will be used to target external IP addresses and will be tested to verify the strength of the UI. Internally, our goal will be to ensure all vulnerabilities within our application such as communication within our application is secured. Ensuring that all potential risks are uncovered will help ensure that threats will be deflected and detected.

Vulnerability testing will include the following Automated Tests:

- Binary Analysis: Will sift through files that are composed of binary code and evaluate the applications data flow without the necessary needs of reverse-engineering bits.
- Cross Site Scripting: Evaluates the network of the application to ensure that no injections of malicious scripts or readings of sensitive page content to a victim user.
    - The target will be to identify all the pages in the web application finding all injectable parameters in forms, headers, and URLs while reporting information to the tester.
- SQL injection attacks: Scans will be done to protect each registered user's personal information as queries can become intercepted and modified disrupting the applications content.
    - SQL injection attacks vulnerabilities precautions need to be taken to avoid the compromise of the network server, all back-end infrastructure, or DoS attacks.
- Platform/Feature Vulnerabilities: These scans will be used within our application to ensure that collection, maintenance, and disseminating information are all discoverable within vulnerability testing.
    - This will be used to identify any vulnerabilities within the RDBMS with the use of SQL injection vulnerability protection, we plan on also testing other vulnerabilities that can occur with the database and network server (i.e Initial Deployment Failures, Misconfigured Relational Database, and Inadequate Auditing).

2.4.2 Automated Penetration Testing

Penetration testing will be used to simulate a hacker within MotoMoto to ensure that all vulnerabilities that are scanned as a potential issue are ethically exploited. With the implementation of third party software we will prioritize all flagged issues that can cause a data/network breach and attack the system to fully understand the vulnerability.

Using Vulnerability scans and to find all potential risks in the source code will allow for a thorough test of the applications security architecture. In our case we will be using penetration tests on discovered vulnerabilities so that all internal components of the application are protected (such as user data management, application data, and network security).

Features to be Penetration Tested:

- <u>Web Application Tests:</u> Ensuring that all functionalities and interfaces with sets of data in the web application are thoroughly tested by information from Vulnerability Scans
  - Back-end technologies (RDBMS, Dependencies, High level Programming Language)
  - RESTFUL API's
- <u>Network Service Tests:</u> Evaluating the network system and the services provided for probable security issues and will be knowledgeable in penetration of solving security issues with:
  - Network Technologies (firewalls, switches, routers)
  - Network Protocols (HTTPS)

Types of Internal Penetration (White Box Testing) that will be used

- SQL injection
- Proxy Server Attacks
- Identification and Authentication Failures
- Cross Site Scripting
- File Upload Flaws

External Penetration (Black Box Testing)

External penetration will be more focused on attacks on the whole system itself where the IP is given to our automated system to try to simulate possible attacks around the perimeter of our network. The main goal will be to compromise the network's firewalls where an assessment will take place to see what is exposable within MotoMoto.

2.4.3 Testing Metrics

- Vulnerability Scans
  - We plan to scan our vulnerabilities every Wednesday of the week to ensure that all potential risks are uncovered
  - Each scan will vary in time as it will depend on the number of IPs that are accessing the application
  - We will measure the vulnerability scans by the

        - Number of Critical Vulnerabilities Detected
            - Critical security scanning possibilities of SQL injection
            - The application can be compromised by attackers
        - The Scanners distance of Coverage over the internet
- Penetration Testing
    - Penetration testing will be measured based on the success of the attack itself and visibility of what needs to be fixed
    - Each automated penetration test can take from 15 minutes to 2-3 days depending on the systems that are being used to test the application
    - Mostly, penetration testing will be used to attack known and unknown vulnerabilities within the network and application to identify specifics of each test.
    - Penetration testing will be conducted during each inclusion of a new feature which can potentially take a week during each sprint to complete after gathering information from the vulnerability scanner.

2.4.4 Risks of Security Testing

- Vulnerability Scans have very little risks compared to penetration testing as there can be intrusions that take place at the same time of testing.
    - The pros of vulnerability testing significantly outweigh the risks of testing as there could be crashes due to programming errors that the team sets.
- Penetration testing can have many different issues that can occur.
    - Since we will be using penetration to exploit vulnerabilities, system outages could take place due to reckless behaviors during the implementation of white box testing and can lose productive hours if the application is simply not available to be modified.
    - We can also come across false negatives (undetected vulnerabilities) which can be the cause for concern as testing will be only limited to so much time.

2.5 Usability Testing

Usability testing measures the application's ease of use and how the users interact and behave with the application. This testing gauges the user-friendliness of the application through the user experience and interactions. It provides feedback from potential users, and any negative feedback can be corrected before the public release of an application. This testing is an iterative process, and it will be performed repeatedly until the application is not confusing to use and the user behaviors are as expected while the users are interacting with the application. Constant usability testing reduces the risk of creating a confusing application and saves budget and other costly resources.

2.5.1 Testing Metrics

- Usability testing will be used at least once every sprint to evaluate the application's user-friendliness among the clients and other potential users.
- Testing feedback will be validated and measured based on what the feedback of the client has noted on what should be modified for their liking
- All unit testing and integration testing should be completed to avoid all bugs that are viewable by the user.

2.5.2 Usability Testing Risks

- With the conduction of usability testing we can face problems with out of scope implementation issues that will ultimately change the shape of our software architecture.
  - Software architecture changes are going to happen but, our goal will be to keep a consistent shape to avoid overhauls.

2.6 User Acceptance Testing

Acceptance testing will be used during the final stages of the applications release to the client. The main goal here will be to ensure that each feature within MotoMoto will be working based on the clients requirements and is acceptable during deployment. This mostly will represent our Alpha and Beta testing that will most likely be the client and outside users to help the developers use the application for themselves before the applications due date (Scheduled May 2022).

With the integration of end-user testing, this process will ensure that all levels of testing have been completed and the application is ready to be deployed. There should be very little to no vulnerabilities and the only errors that are acceptable is if the application should have cosmetics errors within the style sheets.

All of this testing will be manually contributed by outside users as it will be to ensure that the application will be working properly. This testing is done at the end of the SDLC where the client or any interested users will be able to actually run the application itself. All testing will be monitored by reviewal email or verbal messages to the developers to ensure that everyone is on the same page of what is liked or not liked about a specific view of the application. This testing will only happen once in the project's actual lifecycle and will happen potentially a week or two before release date.

2.6.1 User Acceptance Testing Metrics

- Our testing will mostly be based on what our users experience outside of our clientele. Our goal will be to receive feedback from outside users to ensure the likeness of the application by car enthusiast. User acceptance testing will only happen once during the SDLC.

- Testing will be conducted through email reviews or in person feedback on what was liked or disliked about the application
- Using this feedback, we will be able to shape the application better for future thought processes on the release of other versions past initial release.

3.0 Environmental Requirements

3.1 Softwares

- Visual Studio Code - Minimum Version 16.11.4
- Browser:
    - Google Chrome - Minimum Version 94.0.4606.71
- MariaDB - Minimum Version 10.7.0
- Wireshark - Minimum Version 3.4.9

3.2 Testing Tools

- Visual Studio Code - Minimum Version 16.11.4
- Wireshark - Minimum Version 3.4.9
- Google Pagespeed Insights - Minimum Version 8.4.0