Code Testing Plan and Result

Climate Awareness

ENSE 400/477

Anuprus Burokas - 200389250

Mahamed Bashir - 200371994

Timothy Pasion - 200253968

Software testing Definitions

Unit Testing: Tests behavior one piece of code. This could be testing classes and widgets. The purpose of unit testing to see the expected behavior of a method matches with actual output. E,g testing login method.

Integration Testing: integration testing allows us to combine different units, modules , etc and tests the functionality of these as a group .

System Testing: System tests generally allow us to test the overall behavior of our app. For example testing to see the behavior of our app when a user tries to login without the internet .

Unit Testing and Integration Plan

Most of our class's methods/ functions have unit tests which test each input with expected output. We define failure behavior as what the function returns or does in the case it cannot perform what it has been asked to do. For example, a function that checks whether a user entered correct credentials or not.

System Test Plan

System testing
Test plan for system testing for this application would entail the following tests for functional and nonfunctional requirements.
System test will entail the following to test the system code:

1. Testing the fully integrated system and external peripherals to check how all the components of the system interact with one another and the overall system. This will be end to end scenario testing.
2. Testing inputs to see if the components and the system returns the desired results.
    a. This would include any text inputs and gestures that the user does in the application.
3. Testing the user interface and user experience.

In order to test the system code we can perform the following tests.
Regression Testing
Usability Testing

Functionality Testing
Stress Testing

Regression Testing
Regression testing is done after each implementation of a new feature has been added to the system or any refactoring of the code. This is to test that the new feature or refactored code that has been added or changed does not affect the existing features or any refactoring of the code has been done. To set up regression testing for our application we would need to identify any bugs by debugging first, this will allow us to build suitable test cases to use on the modified and affected parts of the code.

Usability Testing
To evaluate how user friendly our application is we can perform usability testing on a group of people that fit our target audience(North star customer) of the application. Qualitative measurements metrics that we can test for are: Usefulness, discoverability, accessibility,desirable and usable. The feedback from the test users will help with modify the application to satisfy the end users when the application has been completed.

Functional Testing
To perform functional testing we can perform the following in order.

1. Identify a set of test data.
2. Compute expected results with the selected test data.
3. Execute test cases.
4. Compare results between the expected and actual results.

To do this we can either do manual testing, in which we manually input data to see if the expected results are comparable to the actual results or we can use automation tools to help with efficiency.
In the case for our application it will look at the main functionality of the system and the performance of those core functions, can the user navigate through the application without any trouble, are there suitable error messages that can guide the user to correct their actions and persist any data the user has already inputted.
An example of testing if there are suitable error messages is in the forum posts. user's can not submit a forum post with empty fields or any profanity, in the case of these errors the user is guided by highlighted red input boxes to fix before they can submit a post.

Stress Testing

Testing the system's robustness and error-handling capabilities under a large volume of concurrent users using the application. A stress testing plan that we can implement in our application is to gather system data, analyze the system and define test goals.
For this application it would be suitable to look at how many concurrent users can use the application at once, and look if there are any data locking and blocking, network issues and any performance bottlenecks in the application. Neo Load can be used to evaluate the application performance under a large volume of users, and analyze response times.
Firebase is our backend as a service which has metrics on user engagement which can be broken down into retention span on the application, and new users. Firebase also provides metrics of the number of reads and writes per day.
The metrics to look for in stress testing would be:

- Application response
    - Page Time: How much time has elapsed to retrieve all the information in a page
    - Hit Time: Average time to retrieve images or a page
- Scalability & Performance
    - Pages per Second: How many pages have been requested per second
    - Throughput: Response data size per second
- Failures
    - Failed Connections: Number of connections that have failed
    - Failed Hits: Number of failed attempts for broken links or unloaded images

With these metrics in mind we run a script to simulate a scenario of a large volume of users and inputs into the system and get results back on the performance of the system and compare expected results to actual and then tweak the code to meet the test goals.

This is a system test plan that we can use to test our application,with each minimum viable product being released on a bi-weekly basis we can test each new feature being added to the system using regression testing. User testing can be done after each new feature has been added and after all the features have been added to get feedback on the user experience and refactor the feature or go back and fix any undesirable features in the application. Functionality testing can be done before stress testing, this will ensure that all the user inputs and or actions are tested for, stress testing can be done when all the features have been added to ensure that a large volume of users and their actions can be processed without failure of the system.

Github Action Workflow

We are able to run all of our tests naturally utilizing GitHub Actions. The CI/CD tools that GitHub offered to us allow us to set up files that define the workflow for running the tests. Our GitHub Activities files are under the .GitHub/workflows directory in our repository. The workflow incorporates the checking out of the repository, starting runtime services, and then the Flutter test commands. We have also configured our GitHub action to run integration testing once a week to test overall integration testing.

Test plan result

We ran into issues with flutter mocks and the package we were using thus resulting in our testing to fail whenever the github workflow is triggered. It is something we would have to fix in the future but as of now we were not able to run our tests successfully.

Future work

The issues that we ran into during building and testing each feature implemented is using GitHub actions to automate scheduled unit and integration testing, but the results of these were always failed, thus we opted into manually testing each of the finished features. To test these features manually we had to input different kinds of inputs to check if the resulting output gave the correct results. Testing the functionality of each feature while building also helps with finding out if the correct outputs are being shown through console log. To manually check if the integration of the new features successfully are implemented and works with the rest of the application is done by debugging to see if the feature would break the application on run time.
To increase productivity in this project is to implement automated testing for unit and integration testing, this would save a lot of time rather than manually inputting values to test for any inconsistencies in the project.

Reference

Hamilton, Thomas. “What Is Software Testing? Definition, Basics & Types in Software Engineering.” *Guru99*, 25 Feb. 2022, https://www.guru99.com/software-testing-introduction-importance.html#5.

“Mock Dependencies Using Mockito.” *Flutter*, https://docs.flutter.dev/cookbook/testing/unit/mocking.