ENGG*4450 - Large Scale Software Architecture

ASSIGNMENT 4 - Build and Test a New Feature

Instructor:
Dr. Petros Spachos

Teaching Assistant:
Marc Baucas

Group: 3

Pieter Jurgens Krige - ID: 1012072

Andrei Korcsak - ID: 1008518

Disha Singh Nath - ID: 0995702

Harshal Patel - ID: 1032961

Bilal Ayash - ID: 0988616

Neel Bhandari - ID: 1004853

Submission Date: Thursday, December 3rd, 2020

School of Engineering

University of Guelph

Contents

List of Figures

List of Tables

1 Introduction

The goal of this assignment was to build and test the new feature of SignalMessenger discussed in the previous report or another feature if the previous chosen one was analysed to be infeasible. To successfully implement the feature and provide the user with some additional value, the group tested and debugged the functionality. To demonstrate the validity of the steps followed for this assignment, verification processes were followed and documented. Finally, a user guide was built to explain the use of the new feature to the users. For efficiency of the implementation of the new feature, a development process, in particular the SCRUM process, was closely followed.

2 Development Process

The team continued with scrum model approach as it has done an incredible job in making each team member more efficient and streamlined. SCRUM has many advantages such as high productivity, great flexibility, adaptability, increased team accountability and high transparency among other benefits. The group initially started with Sprint - 15 day iteration compared to the traditional 30 day sprint, due to which the process was scaled down accordingly. This started with allocating 1/4 day to group meetings to maintain a balance between the other courses. The next thing was to decide on Prioritizing Product Backlog which is shown in Table 1. In addition to this, the group also followed the Daily Scrum process which included 15 minutes team meeting each day. Each team member would answer; What have they done since the last meeting; What they will be doing between now and the next meeting; What obstacles they came across in approaching the task. The visualization of SCRUM process can be seen in Figure 1[1]. This assignment had obligatory seven steps to go through in completing the assignment productively. The group obeyed these steps over the course of 15 days as follows:

1. The first step is to review the analysis performed for case insensitive contact sorting in previous assignment. Checking on the requirements and is it up to date. Is the effort estimation still accurate ? Finally, selecting on appropriate set of functions to implement case insensitive contact sorting. Here, the group had difficulties with the implementation of the contact sorting feature because of the reasons mentioned in section 3. As a result the group re-scrutinized the new feature request list and decided to go with 'Better White Theme'.

2. This step is to decide on development process and how to keep track on implementation effort. SCRUM model is the one group resumed on. Group followed the product backlog items with the estimated time as shown in Table 1. Pair programming was also used.

3. The third step is to build and verify the functions as in section 3 and 4. Verification strategies such as informal and formal reviews, static and dynamic analysis were used. This also includes keeping track of the number and type of defects discovered through each technique.

4. Fourth step is to build an automated test suite for better white theme as shown in section 5. This was done after implementing the functions based on our product backlog items.

5. This subsequent step is to do the additional white box testing on one of the most complex classes implemented.

6. To further provide user with a short user guide on how to use the better white theme feature as in section 6.

7. The final step is report writing to describe the steps followed to implement and verify the activities for better white theme feature.

The group used pair programming where Bilal, Pieter and Andrei joined on a video call and Pieter started on programming while the other two were navigating and reviewing each line of the code as written. They were switching the roles frequently to finish the work productively. This was a great way to do the task as the observers can spot the errors, bugs, and potential improvements to the code on the spot and it made the work very time efficient.

Figure 1: Visualization of SCRUM Process

Table 1: Product Backlog Items with the Estimated Time Using SCRUM Model

Backlog Item	Start Date	End Date	Duration	Member
Review of analysis performed for the requirement analysis	Nov 19	Nov 19	1	All
Development Process, Survey and Reflections	Nov 20	Nov 21	2	Harshal
Build the functions (Implementation Analysis). Techniques used and their effectiveness. White box testing for the most complex class by using suitable testing strategy	Nov 22	Nov 26	5	Bilal, Pieter, Andrei
Verify the functions using informal and formal reviews, static and dynamic analysis etc.	Nov 27	Nov 28	2	Disha
Automated test suite for better white theme feature	Nov 29	Nov 30	2	Neel
User guide on how to use better white theme feature	Dec 01	Dec 01	1	Harshal
Report writing to describe the steps followed to implement and verify the activities for better white theme feature	Dec 02	Dec 03	2	All

3 Feature

3.1 Feature Change

In Assignment 3, the group chose to implement a feature request which sorts the contacts in a case insensitive manner, different from Signal App's current case sensitive sorting. Research on Signal App's GitHub page was also conducted where we found information regarding the feature, however, it was outdated and no longer relevant for the current version of the code we are working on. Getting idea from the GitHub page, to implement the change, efforts were made to identify a Java class that contains elements like contact_sort, contact_list, name_sort or name_list but none were found for contact sorting purpose. After more speculation on where the information regarding the contact sorting was located, it was found that we don't have access to the database used so we were unable to detect how it was indexed, which as a result, didn't allow us to truly understand the data sorting mechanism. Given the lack of access, knowledge of group members and the time constraint, the group concluded that the implementation of the new feature to arrange contacts in case insensitive manner was infeasible.

To make changes to the project plan according to the above findings and make sensible choices, the group held a meeting to decide the next steps. After holding a voting session, majority of the group voted to change the new feature that needs to be implemented to one that has files accessible to the public, fits our timeline and risk analysis, and improves the user experience. With keeping these criteria in mind, the group went on the Signal App's feature request page and chose to implement a new feature which provides better background and text contrast than those currently in place.

3.2 Feature Choice

From the Signal Messenger App's web page, the request to get new colour theme for light theme setting was selected. Currently, the light theme supports light text on coloured (dark colored) background but generally the aim of a light theme is to have dark/black text on light background. Figure 2 shows the current view of a chat where it is noticeably harder to read the white colour text on the dark background. This request was chosen as it was seen as an valuable improvement by the group and given it had more than 174 views on the thread, it was also seen to be popular among other users [2]. Additionally, by adding this feature, Singal App becomes more user friendly and has the same light theme standard followed by many leading applications such as WhatsApp as seen in 3. Other major criteria that were part of the decision making process are discussed in detail in the next section.

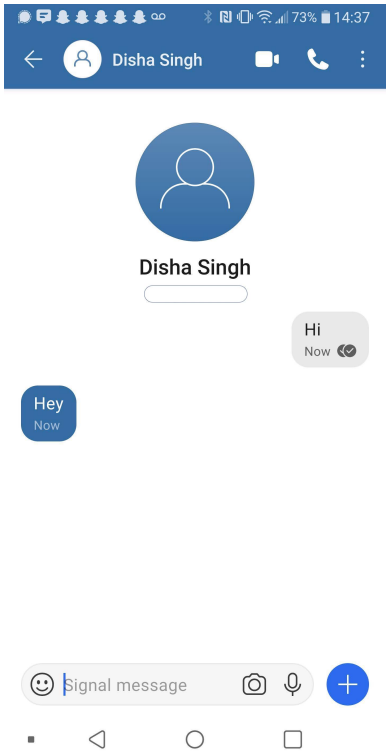

Figure 2: Signal App's Light Theme

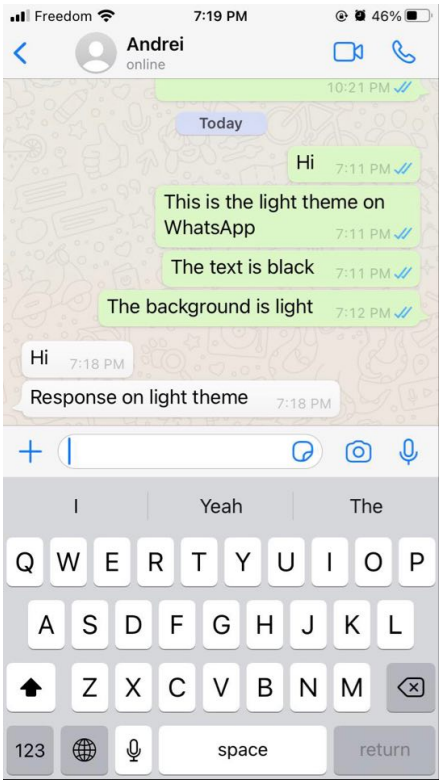

Figure 3: WhatsApp's Light Theme

4 The Criteria

To decide which feature the group was going to implement, using tools like customer priority, effort estimation and risk analysis, the criteria for the feature were set up. Customer priority was seen as an important criteria while choosing the new feature as it is important to take into consideration the user's wants and likes to improve the application since they utilize the final product. With a team of 6 individual along with a deadline to be met, effort estimation was also considered as a criteria to allow the team to complete the implementation successfully. Further, risk analysis was important while undertaking the new feature to ensure the feature chosen does not pose a catastrophic failure to the application in the event of an existing bug or defect in the code. The team also looked at the existing libraries and frameworks that could assist with the implementation of the feature.

4.1 Customer Perspective

The feature for this assignment was chosen keeping in mind the customer feedback and requirements.In addition to the initial research, to make the decision stand alone and bias free, the group conducted a survey where 5 questions were asked to possible Signal App users. Question 1 to 4 were chosen to get the general user perspective and their view on the current light theme to understand the user perspective better. Question 5 was added to quantify the user's desire of having the new feature implemented. The responses were recorded and upon scrutinizing the feedback, a conclusion was made that roughly 82% (4.1/5 stars in question 5) of users wanted a new light theme to be introduced, which would make the application more aesthetically pleasing and easy on the eyes. The statistics from the survey can be seen in Figure 4 and 5.

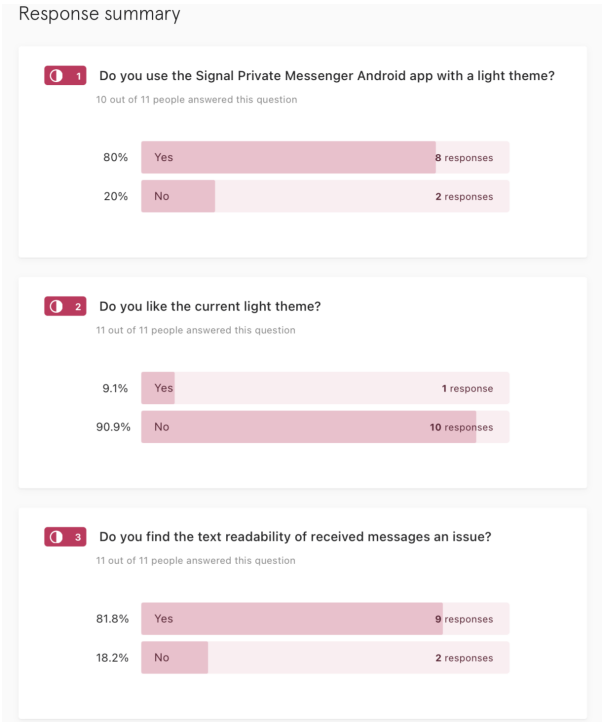

Figure 4: Survey Result 1 of 2

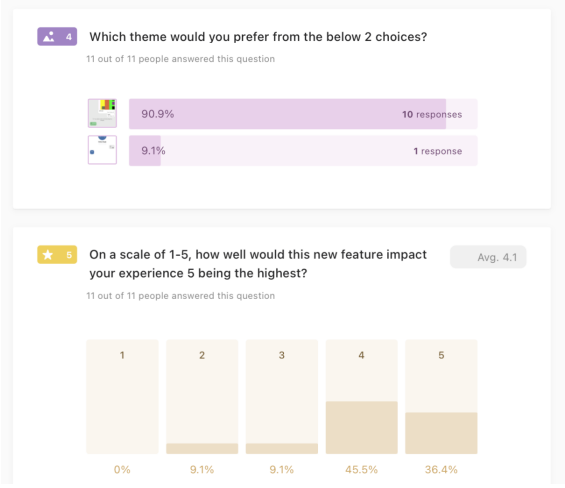

Figure 5: Survey Result 2 of 2

4.2 Effort Estimation and Risk Analysis

In order to determine whether a feature should be implemented a series of factors needed to be considered to assess how much 'effort' would be required. For the purposes of this assignment the coding time for implementation was chosen as our primary metric to determine the effort needed. It was thought that although the time estimate was susceptible to change as the project unfolded, it would still be a good method as we could then determine where time would need to be allocated for timely completion.
The three-point estimation method was used to get the best estimate of effort required.

$$E = \frac{w + 4m + b}{6}$$

This equation took into account the best, average, and worst case scenarios in order to make certain that the decision made took into account all aspects of the effort required. This was quantified by estimating the time required for all scenarios and then the equation was used to obtain an average time. The estimates made were based on a review of the code where it was determined that in a best case scenario for the developer, b, this feature would take about two hours to complete. In an average case, m, it may take closer to to 6 hours to complete, with the worst case, w, for the developer, taking a full work day to implement, at 10 hours. Thus, based on the estimates made, the total 'effort' required would be around 6 hours of development time.

The estimate of effort only took into account the time required, however the risk associated with the implementation of the feature was also very important. Prior to the implementation of the feature, a risk breakdown Table 2 was made.

Table 2: Risk Analysis

	Likelihood of Occurrence		
	Very Likely	Possible	Unlikely
Loss of app functionality	Catastrophic	Catastrophic	High
No contacts displayed	Catastrophic	High	Medium
Sorting out of order	Medium	Medium	Low
Inconvenience	Medium	Low	Low

5 Implementation Analysis

The feature which was ultimately implemented was an improvement to the light theme of the Signal App for Android. The main concern of the current light theme is the white text on a dark background when receiving a message, this is not ideal for users who have poor eye sight causing difficulty properly seeing the smaller text. For this reason, the theme was adjusted to avoid the use of white text in conversations. This change was implemented in the themes.xml file, this file stores all the data needed on which elements need to be altered when the theme was changed from light to dark. In this file the light theme was updated to reflect our need for dark text: sent_text_primary_color, sent_text_secondary_color, received_text_primary_color, and received_text_secondary_color were all changed to be black. As a result of changing the text colour, the message received background color also needed to be modified to get a well suited contrast. To achieve this, the conversation_colors.xml file was updated so that each color was a lighter alternative of the color it was replacing. Thus, when these changes were made, the black text was now more readable when a message was received. It was also asked in the feature request to change the very white bright background colour, this was done and the background was made slightly darker with a light grey. Figure 6 and 7 show the difference before and after the implementation.

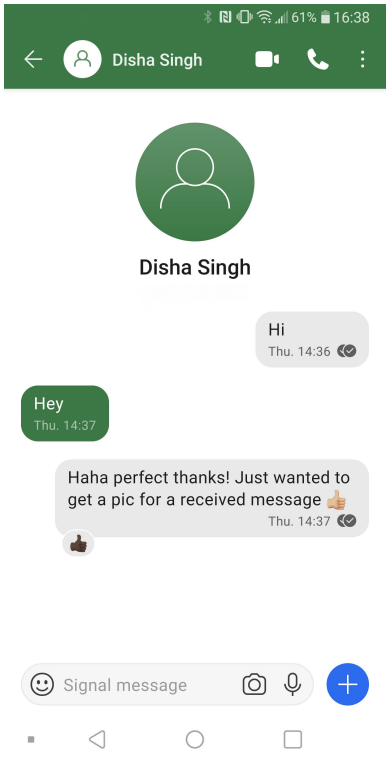

Figure 6: Before Change Implemented

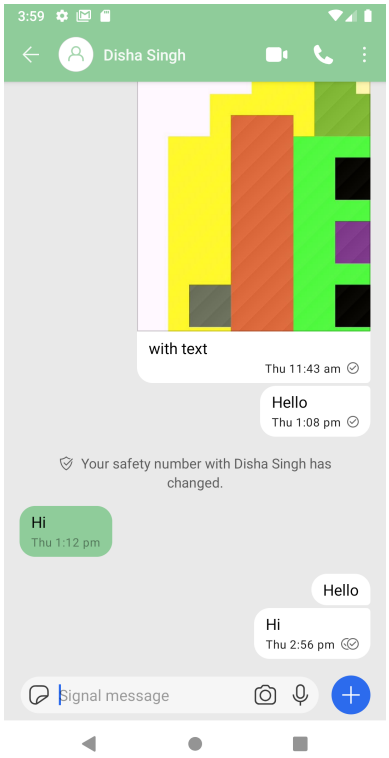

Figure 7: After Change Implemented

6 Verification Strategy

Different techniques and people come forth with different defect analysis. Removing defects and issues during the implementation stage of the project is 100 times cheaper than to remove after implementation [3]. In order to keep identify issues discovered during the process and achieve a higher defect-removal percentage, a combination of techniques is favourable. After some discussion, the group concluded that it would be beneficial to test the code, so testing was done on the application to ensure proper implementation is being carried out. Further to cover other aspects of the project other than the code check, the group also decided to conduct an informal design review to have a quick scan of prominent issues that maybe be present in the application.

6.1 Informal Design Review

Table 3 shows the progress of the project up to this section. It was created keeping the software project under consideration. It documents the review objectives and their status on topics such as system requirements and development of the project. This strategy helped keep track of the issues and delays present during the process. Though the team started working on the new feature, they forgot to distribute the document which caused a delay,creating a backlog. It also reminded the group to periodically manage and control the new changes made to the file. By implementing this strategy we learned a lesson to always look back at the steps at every stage of the project to ensure better productivity.

Table 3: Informal Design Review

Topics	Date Approved	Status
The team periodically holds meetings	Nov 17	In-process
The design methodology was chosen based on user's new feature requests	Nov 17	Completed
Necessary resources have been identified to perform software design activities on the project	Nov 17	Completed
A new feature design is approved	Nov 20	Completed
The software structure has been chosen based on design methodology	Nov 20	Completed
A System Design Document is created	Nov 20	Completed
A Functional Design Document is distributed to team members	Nov 20	Overdue
A Functional Design Review is performed	Nov 21	In-process
At least one in-progress assessment has been performed	Nov 23	Completed
Changes to the system design baseline are managed and controlled	Nov 25	In-process
Software quality assurance periodically reviews the design	Nov 25	In-process

6.2 Testing

Testing is crucial when implementing new code, as it allows developers to verify that the code behaves exactly to their specifications. Testing while implementing the code allows errors to be fixed more easily, as there are fewer things that haven't been corrected at the end. This allows developers to look for errors in a small chunk of code rather than the entire file. As a result, testing of the new feature implemented in this project was done using various techniques and resources. These include XML validation, White-box testing, and Black-box testing.

XML validation is a web-based XML error checker. This platform allows the user to input a XML file to check for syntax errors. Figure 8 illustrates the output from xmlvalidation.com. Through the figure, it is evident that no syntax errors occurred while checking the code.

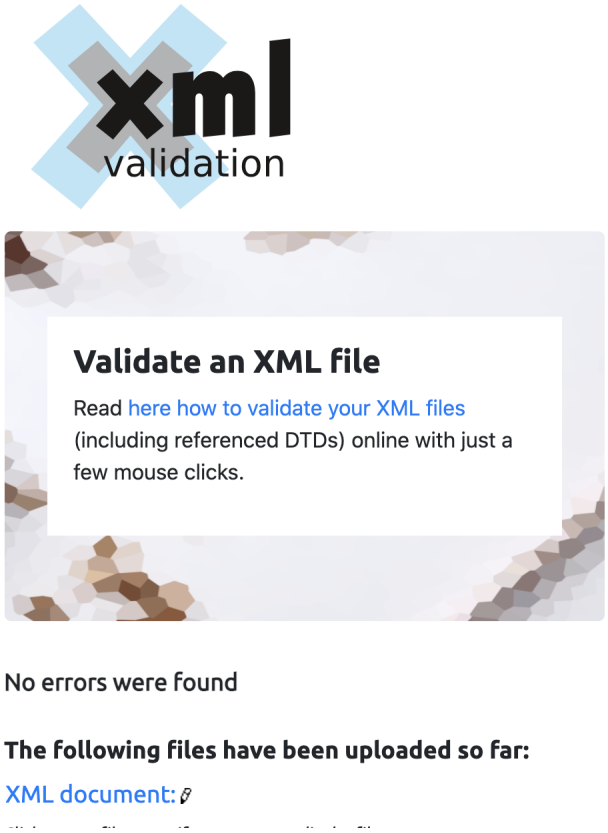

Figure 8: XML Validation Check

6.2.1 Black-box Testing

Figure 9 presents a use case diagram for the feature added to the Signal App. Since the only change made to the android applications is the UI theme alteration, the black box test consists of testing the user interaction with the application through the use of the touch screen on the phone. Outside of the feature, code was also checked to ensure that the remainder of the software performed as expected before introduction of the feature. This involves going on the signal-app and testing other features such as search for contact or send a video in messages. The user case diagram for the new implementation is presented below:

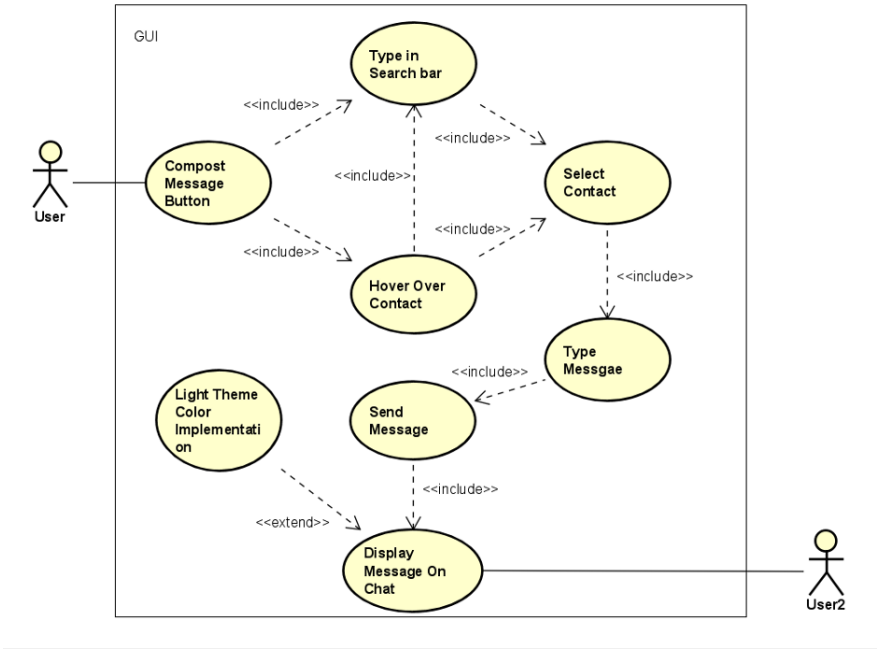

Figure 9: Use Case Diagram For Testing

The first user clicks on the compost message button and types in the name of user2 in the search bar. User1 will hover over user2 name and selects the contact then types a message and presses send. After the message has been sent, the message is displayed on the chat. This is where the implementation is extended. The graphical user interface is edited, and the colors of the messages received and sent are changed to make it easier for the user to read. The main issue was the fact that messages text where in light colors over dark backgrounds, the new implementation changes the messages background to a light color making it easier to read.

6.2.2 White-box Testing

To ensure that the code behaved as expected, it was tested throughout the development process. As all changes were done to the user interface using the themes.xml file, a Junit program was created for testing purposes. Throughout the development process the code was tested manually, by doing so we were able to verify the appropriate changes were made to the colour scheme to allow better text visibility for users. Unfortunately, the automated testing does not currently fit the objective. It was implemented using Junit and would have been used to verify all the required fields that were to be changed to their correct colours.

Part of our testing process required us to make changes in the themes.xml files and constantly check on the Signal App to observe any changes when accessing the contact messages. Due to the large number of colour variables in the theme section, testing all the variables separately would not prove efficient, therefore we acquired a different approach to the problem. Our strategy involved searching the variables that relates to our feature such as “conversation” and “text” in order to identify which variables we should work on.

The JUnit testing for our code is located in the TestingTheme.java file inside the directory app/src/test/java/org.thoughtcrime.sercuresms/contacts. This file was used to test our updated colour variables from the themes.xml file. We used the assertFalse() function to determine if the variables do contain data, and then assertEquals() to compare the colour variables with the correct colour we were interested in. Figure 10 shows how the testing of the variables from the xml file were achieved.

```
package org.thoughtcrime.securesms.contacts;

import org.junit.Test;

import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertFalse;

public final class TestingThemes {

    @Test
    public void testColour() {
        assertFalse(themes.getNodesAtXpath("//TextSecure.BaseLightTheme[conversation_background=@color/core_grey_05]").isEmpty());
        assertEquals("core_grey_05", themes.getNodesAtXpath("//TextSecure.BaseLightTheme[conversation_background=@color/core_grey_05]"));

        assertFalse(themes.getNodesAtXpath("//TextSecure.BaseLightTheme[conversation_item_bubble_background=@color/core_white]").isEmpty());
        assertEquals("core_white", themes.getNodesAtXpath("//TextSecure.BaseLightTheme[conversation_item_bubble_background=@color/core_white]"));

        assertFalse(themes.getNodesAtXpath("//TextSecure.BaseLightTheme[conversation_item_sent_text_primary_color=@color/core_black]").isEmpty());
        assertEquals("core_grey_05", themes.getNodesAtXpath("//TextSecure.BaseLightTheme[conversation_item_sent_text_primary_color=@color/core_black]"));

        assertFalse(themes.getNodesAtXpath("//TextSecure.BaseLightTheme[conversation_item_sent_text_secondary_color=@color/core_grey_90]").isEmpty());
        assertEquals("core_grey_05", themes.getNodesAtXpath("//TextSecure.BaseLightTheme[conversation_item_sent_text_secondary_color=@color/core_grey_90]"));

        assertFalse(themes.getNodesAtXpath("//TextSecure.BaseLightTheme[conversation_item_received_text_primary_color=@color/core_grey_60]").isEmpty());
        assertEquals("core_grey_05", themes.getNodesAtXpath("//TextSecure.BaseLightTheme[conversation_item_received_text_primary_color=@color/core_grey_60]"));
    }
}
```

Figure 10: Test Cases

7 User Guide

This section has a short and compact user guide to direct user on how to use the better white theme feature in an effectual manner. Moreover, it shows why the feature is imperative for the user and their eyes. It also improves the readability while keeping user engaged in the conversion.

1. Download the Signal Private Messenger App from the device Appstore or https://signal.org/en/download/
2. Open the Signal Private Messenger on an Android device.
3. Ask a Signal App user to send you a message.
4. Open their conversation.
5. The received message's background should be lighter in color and text should be darker which makes it better on eyes and put less stress on them.
6. Note that the wallpaper color should be less brighter than it previously was, which ultimately makes it more aesthetically pleasing and better for eyes as well as readability.
7. Here are the pictures from before and after the implementation which shows how the user can benefit from the improved light theme.

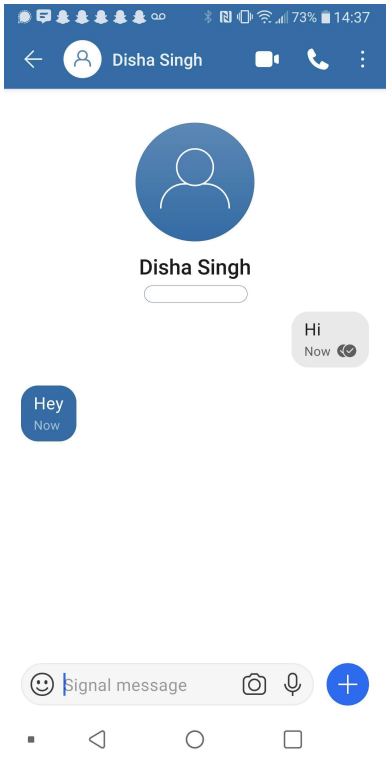

Figure 11: Before implementation

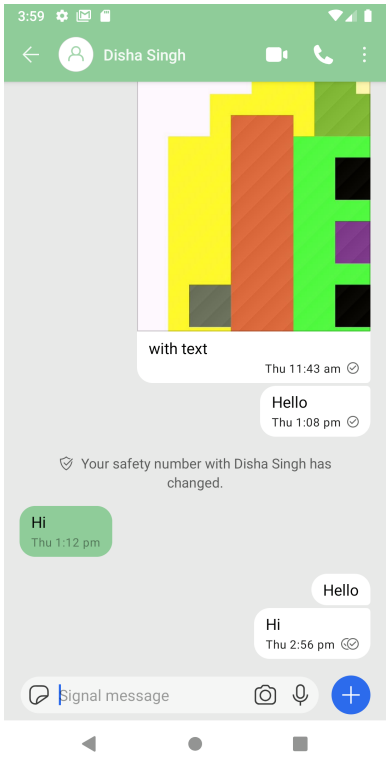

Figure 12: After implementation

8 Reflection

In the span of the previous 3 assignments along with this assignment, the group used variety of tools and techniques which were proved to be amazing, advantageous and made the group's work time efficient.

Assignment 1 required reverse engineering where, the group used Unified Modeling Language (UML) diagram tool. Here, the group did not make use of any software process models such as SCRUM, XP, RUP, etc. The reason behind this could be the very nature of the assignment which is skewed towards individuals rather than group work. Each member were to choose a folder inside the Signal Private Messenger repository in github and make a UML diagram. The only group work here was to make a higher-level UML diagram to show the overall architecture of the system. The group experience developed in this assignment turned out to be helpful for the successive assignments.

The second assignment was group intensive and the basic tool group used was Use Case diagram to get started. Software process model SCRUM was used to guide the group throughout the assignment. SCRUM has many advantages as described in section 2 above. In addition, the group also made use of techniques learned from engineering design courses such as decision matrix, to choose 2 issue fixes out of 7 shortlisted choices. The decision matrix worked out to be wonderful. Though the SCRUM process model did help in majority of the tasks, the group failed to take full advantage of it as majority of the team was new to large scale projects and GitHub. The group did not recognize the full potential of the SCRUM and mismanaged the timing of the tasks. This led group to rush the customer acceptance testing and that is where group lost marks.

Assignment 3 used number of tools, some continued from the previous assignments and some were new. The group resumed using Use Cases and SCRUM model. Special care of duration and timing of tasks was taken to complete each task in time. To settle on the 'Case Insensitive Contact Sorting' feature, a survey was conducted where 4 questions were asked to possible Signal App users. Robustness analysis was used for use cases which assessed existing classes in the Signal Private Messenger application, plus any new classes needed in order to implement the new feature. The team faced problems with task distribution in this assignment as one task is dependent on another. For example, robustness analysis could not be completed without completing use case analysis, risk and effort estimation without decision making and so on. Due to this, the distribution of work became very complicated which resulted in more time and effort loss. In order to avoid this any further, group used an intuitive method called 'Finish-to-Start' dependency where a successor activity cannot start until a predecessor activity has finished [4]. Finally, the group used techniques such as goal modeling, domain modeling, risk estimation and effort estimation to be assertive on the selected feature.

In the current assignment, the group encountered number of difficulties and some were outrageous. The biggest issue was when the group took a deeper dive on how to implement case-insensitive contact sorting feature and concluded that we do not have enough information available to do so and hence decided to change the feature. This is shocking and depressing but at the same time a big learning opportunity. We do not always succeed and failure is a part of life. The group at some point underestimated the importance of the details and effort required. The learning lesson here is how the group quickly recovered from the loss and implemented the better white theme feature. This recovery certainly has to do with the teamwork of each member and how everyone stood up in this difficult situation.

Overall, each tool and technique has its own importance and place to be used at. The members of the group definitely want to make use of tools such as SCRUM, UML, User Case diagrams, robustness analysis, White-box and Black-box testing, Junit testing, Decision Matrix, survey analysis and customer perspective for future software development projects.

9 Conclusion

This assignment proved to be a good learning experience for the group. It emphasized on the implementation of the new feature for Signal App along with different tools to validate the process. By conducting a design review along with different tests for the code, the group was successfully able to detect and remove the issues that could cause bigger and expensive problems in the future. It also provided with a great learning opportunity on how to handle stressful situations when the project does not go as planned. Following the SCRUM process throughout the assignment assisted the group to stay organized and updated with everyone's goals and progress. It also helped motivate the group members. Overall, this assignment gave a good insight on how large scale software architectures are carried out in the real world.

10 References

[1]"Scrum Process: why what you're doing might be not Scrum at all", Magora Systems, 2020. [Online]. Available: https://magora-systems.com/scrum-vs-scrumbut/. [Accessed: 10-Nov-2020].

[2] Vishwa, "Can we have a better white theme?," Signal Community, 21-Aug-2020. [Online]. Available: https://community.signa [Accessed: 01-Dec-2020].

[3] P. Spachos, Class Lecture , Topic: "Lecture 16 - Introduction to Testing", College of Engineering and Physical Sciences, University of Guelph, Guelph,ON.

[4] E. Harrin, "How You Manage Task Dependencies," ProjectManager.com, 03-Jul-2020. [Online]. Available: https://www.projectmanager.com/blog/manage-task-dependencies. [Accessed: 16-Nov-2020].