ENGG*4450 - Large Scale Software Architecture

ASSIGNMENT 3 - Requirement Analysis

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: Tuesday, November 17th, 2020

School of Engineering

University of Guelph

Contents

List of Figures

List of Tables

1 Introduction

On SignalApp's community page, users have the ability to report bugs and suggest feature requests that can help improve the application. In this assignment, the group was tasked to chose one feature request for the SignalApp and analyze it using different models such as goal model to identify user's requirements, domain model to map out domain entities and use cases to define how users can take advantage of the new feature. A robustness analysis was also carried out to identify which classes were to be added/modified to in-cooperate the new feature with the given code. The SCRUM model was followed to ensure productivity and team work during the process.

2 Development Process

The team once again decided to go with the SCRUM model approach as it worked out splendidly in the previous assignment. 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 7 appendix A [1]. This assignment had obligatory six 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 was to go through the list of new feature requests at the Signal Messenger App's web page.
2. The next step included making a decision to select one of the listed features using techniques such as effort estimation, risk assessment and understanding customer priority as described in section 3.1 and 3.2. Moreover, a survey was done to get the user's thoughts on what they contemplate about the new feature selected by the team .
3. The third step was to complete a requirements analysis for the chosen feature as done in section 3.3. This involved identifying why the user wants the requested feature along with relevant assumptions about the problem domain. It also demands detailed list of specific functions and quality requirements along with a rationale for how the functions will allow the users to meet their goals. The group used skills developed through lectures such as goal and domain modeling which can be seen in section 3.4 and 3.5.
4. The subsequent step was to generate a detailed set of use cases for the selected feature. This included a use case diagram to give an overview of the set of use cases, and create individual descriptions for each use case, including pre and post-conditions, exceptional behaviour, and alternative paths for them.
5. To further verify the use cases above, a robustness analysis needed to be done in this step. The analysis accounted for the existing classes plus any new classes needed to implement the new feature.

6. The final step was to write a report describing the steps group went through to select and analyze the new feature request.

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 [3]. This becomes apparent after looking at Table 1 where each descendant task starts only after the precursor task is completed. This enormously helped the group members including any blunders as the chances of doing any task wrong are trifling.

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

Backlog Item	Start Date	End Date	Duration	Member
Reading list of new feature requests	Nov 1	Nov 2	2	All
Development Process and Survey	Nov 3	Nov 5	3	Harshal
Selecting one of the listed feature and applying techniques such as effort and risk estimation along with understanding customer priority	Nov 3	Nov 5	3	Disha
Completing requirements analysis using goal and domain modeling	Nov 6	Nov 8	3	Pieter, Andrei
Use cases for the selected feature	Nov 9	Nov 11	3	Neel
Robustness analysis for use cases	Nov 12	Nov 14	3	Bilal
Report writing to describes the steps followed to select and analyze the new feature request for Signal Private Messenger	Nov 15	Nov 17	3	All

3 Requirement Analysis

3.1 Customer Perspective

The feature for this assignment was chosen keeping in mind the customer feedback and requirements. Contacts in the SignalApp are currently sorted in order where names with uppercase come first and lowercase after. If you have 2 contacts ”dad” and ”Dave”, ”Dave” would show before ”dad” due to it’s first letter being uppercase. Figure 10 shows a visual representation of this sorting. The new Android feature request was to display contacts in a case insensitive order, where ”dad” would be before ”Dave”. This request was chosen as it was seen as an important addition by the group ,and given it had more than 130 views on the thread, it was also seen to be popular among other users[2].

In addition, to make the decision stand alone and bias free, the group conducted a survey where 4 questions were asked to possible SignalApp users. The responses were recorded and upon scrutinizing the feedback, a conclusion was made that 14 of the 17 surveyors (roughly 82%) wanted the case insensitive contact sorting feature to be introduced, which would ultimately makes the process faster and more convenient. The statistics from the survey can be seen in Figure 8 and 9.

By adding this feature, SingalApp would become more user friendly and have the same sorting standard followed by many applications these days. Primarily, users access contacts to invite other people to the app or start a new conversation. Scrolling through all the contacts to find the person whose name starts with ”a” can be time consuming, inefficient and may frustrate the user, ultimately discouraging them to invite their friends. Addition of this feature would benefit the SignalApp by giving it’s users a trouble free experience, thus encouraging participation. It also helps the users by making the process simple and systematic.

3.2 Effort Estimation

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 [4].

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

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

3.3 Requirement

Identifying proper set of requirements is a fundamental element in comprehending a project efficaciously. The group used techniques such as One-on-One asking and Surveys to gather the requirements. Below are the requisites that the group must complete to accomplish the contact sorting in alphabet case-insensitive order.

1. When the invite option is selected the contacts are sorted by alphabet case-insensitive order.
2. When the new message option is selected the contacts should be sorted by alphabet case-insensitive manner as well.
3. When the user search for any contacts, it should appear in an alphabetically case-insensitive order (Ex: dad will appear before Dave opposite to that in figure 10).

Above constraints covers all the corner cases where the user would encounter contacts listing.

3.4 Goal Modelling

It is important to have defined goals to ensure the design of the project is inline with the customer needs. They can be classified as two types, soft goals, that are goals which do not have a clear-cut satisfaction criteria, and hard goals, which describe the functions that need to be carried out. A goal model was used to represent different soft and hard goals for this project. Figure 1 shows the goal model followed to successfully implement the new case-insensitive sorting feature. The soft goals are represented as clouds while the hard goals as eclipses. The relationships between the goals are also highlighted to show their inter-dependencies.

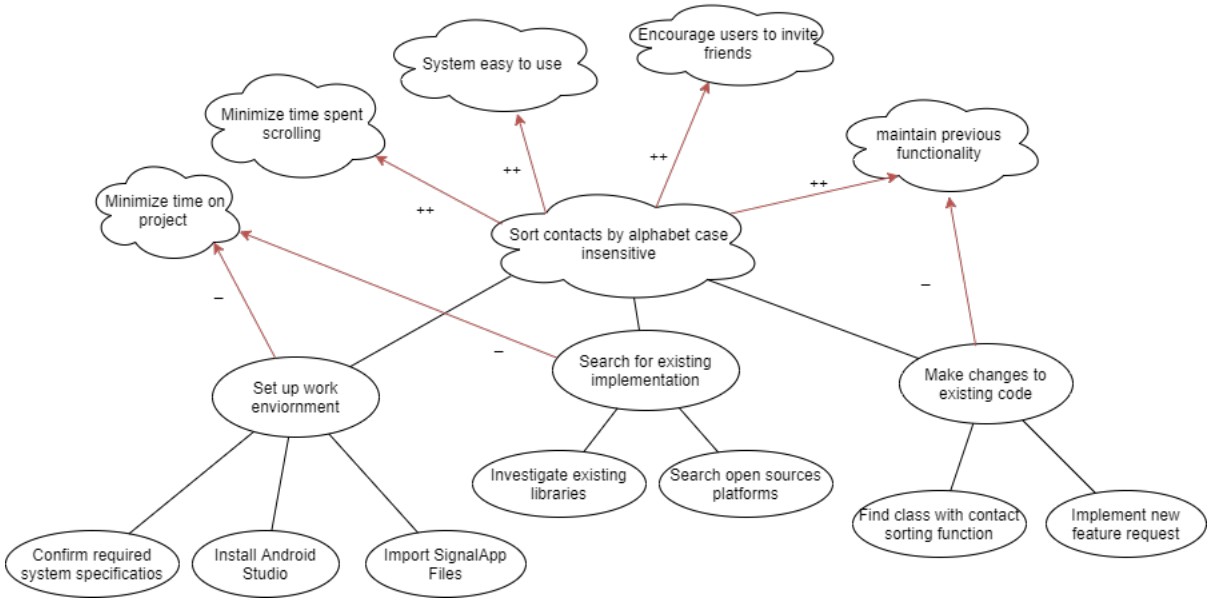

Figure 1: Goal Model for Implementing Sorting by Alphabet Case-insensitive

3.5 Domain Modelling

Before developing a solution for our feature, it must be presented into the form of a diagram so that everyone involved with the project can understand the requirements. Such diagram to be used is the domain model which shows the functionality in a non-technical method, so that it can be understood by both the developers and the client. A domain model provides the actors involved and the relationship between the different elements in order to replicate the problem. By defining the elements required for our feature, it will help us avoid any confusion that could lead to the wrong design.

The diagram in Figure 2 shows a visual representation of an user accessing the contacts list and scrolling down for a specific contact name. This domain model represents the functionality prior to the implementation of the feature. While searching for the contact, the user realizes that they are having a difficult time finding the contact name they are looking for due to a name sorting issue. If some of the contacts on this list happen to be in lower case, they will be sorted into a group separated from the names starting with upper case. Upper case names will be sorted first followed by lower case names. This will cause difficulties in finding a particular name when the alphabetical order is not proper.

The second diagram in Figure 3 shows the functionality after the alterations to the code. When the user accesses the contact list, the contact names are now sorted in a case-insensitive manner. Names starting with an upper case letter and lower case letter are no longer treated separately as two different groups, and are correctly sorted alphabetically.

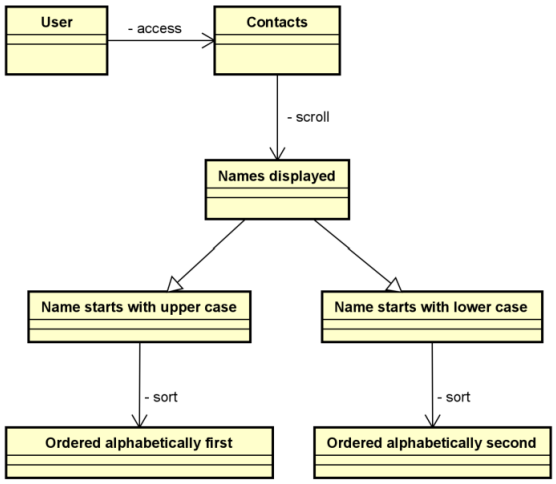

Figure 2: Domain Model for Contact Name Sorting Alphabetically before Implementation

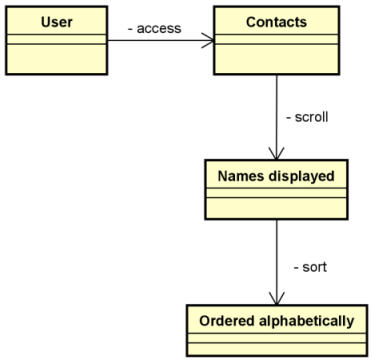

Figure 3: Domain Model for Contact Name Sorting Alphabetically after Implementation

4 Use Case Analysis

When looking at the contacts on the Signals app, sorting is performed alphabetically with case sensitivity. This means that two contacts, both starting with the letter A can be stored under two different categories; “A” and “a”. Figure 4, illustrates the current state of the Signals app. Rather than seeing the contacts appear in the correct alphabetical order, the contacts beginning with an upper-case letter appear first, followed by the contacts starting with a lower case. By sorting contacts with case sensitivity, the contacts book becomes too complicated to navigate. It also requires more space to be stored. The use case can be seen in Figure 4 and is described below:

Main Success scenario:

1. Go to system contacts
2. Create new contact A, must start with a lowercase A (”a”)
3. Create new contact B, must start with an uppercase B (”B”)
4. Login to Signals app
5. Access contacts to invite friends
6. Scroll and search for contact A
7. Contact B appears before contact A

To make the contacts easily accessible it is vital to change the sorting method to alphabetically case-insensitive. To have sorting done in a case-insensitive manner, we no longer have two categories in contacts for names starting with the letter A (for ”A” and ”a”). This allows for users to browse their contacts in a more intuitive and efficient manner. The use case after changing to case insensitive is described in Figure 5:

Main Success scenario:

1. Go to system contacts
2. Create new contact A, must start with a lowercase A (”a”)
3. Create new contact B, must start with an uppercase B(”B”)
4. Login to Signals app
5. Access contacts to invite friends
6. Scroll and search for contact A
7. Contact A appears before contact B

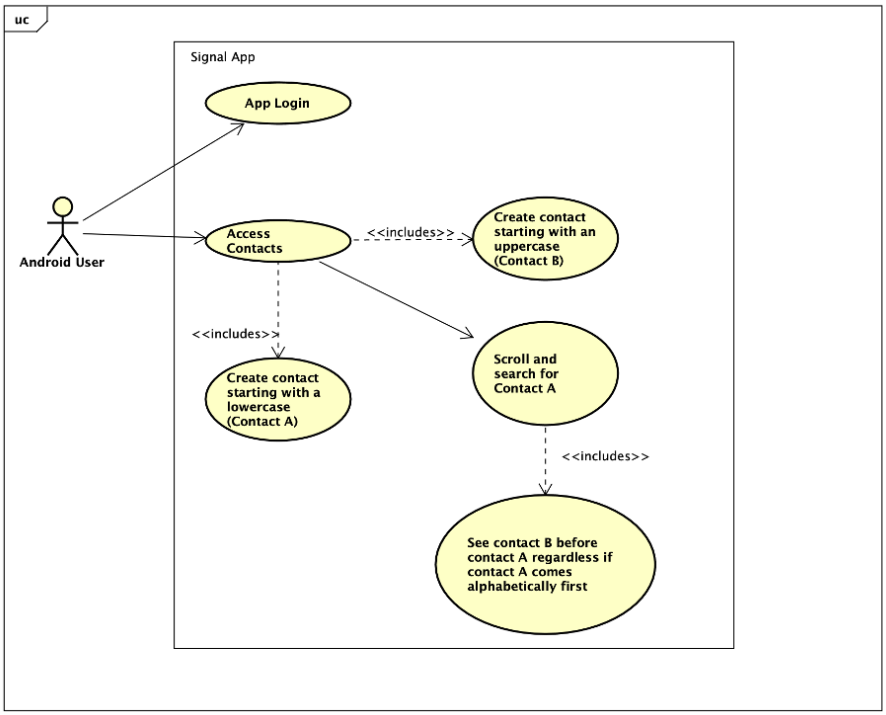

Figure 4: Use Case for Sorting by Alphabet Case-sensitive

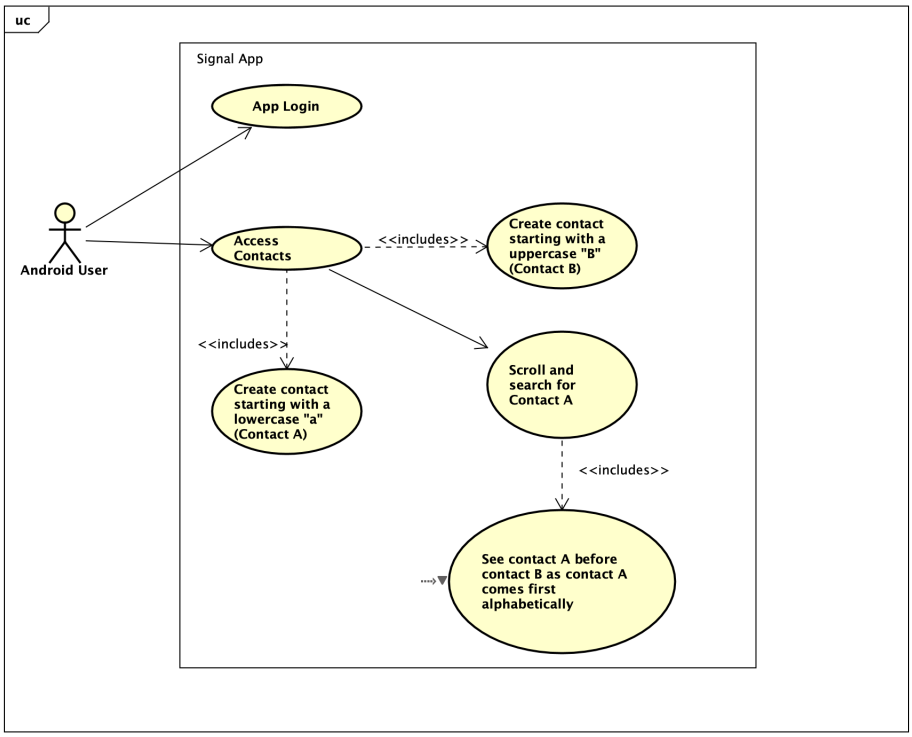

Figure 5: Use Case for Sorting by Alphabet Case-insensitive

5 Robustness Analysis

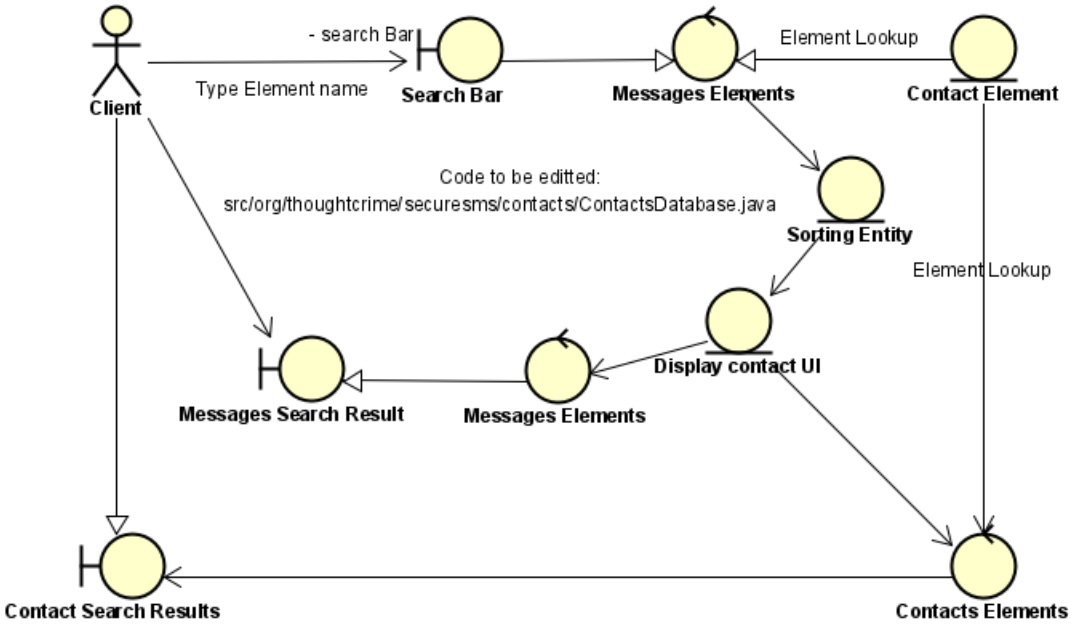

Figure 6: Robustness Analysis Diagram

Figure 6 highlights the major use cases for the Signal app search contact. No classes will be newly added to the application, but the sorting algorithm will be altered. The sorting algorithm that sorts the database can be found in the securesms folder under the name of ContactsDatabase.java. The client types the element name in the search bar (boundary class), using the messages element (control class) and the contact element (entity class), the data is saved into a database. The sorting entity is used to remove unmatched results and displayed on the UI. The main goal is to fix the sorting algorithm that exists to take lower case letters in consideration when comparing with upper case letters. Refer to figure 10 to see the current sorting algorithm in action.

To successfully implement this improvement the following class should be improved:

```
  public Cursor getCursorForRecipientFilter(CharSequence constraint,
   ContentResolver mContentResolver)
{
  final String SORT_ORDER = Contacts.TIMES_CONTACTED + " DESC," +
                            Contacts.DISPLAY_NAME + "," +
                            Contacts.Data.IS_SUPER_PRIMARY + " DESC," +
                            Phone.TYPE;
```

```
final String[] PROJECTION_PHONE = {
    Phone._ID,                    // 0
    Phone.CONTACT_ID,             // 1
    Phone.TYPE,                   // 2
    Phone.NUMBER,                 // 3
    Phone.LABEL,                  // 4
    Phone.DISPLAY_NAME,           // 5
};

String phone = "";
String cons  = null;

if (constraint != null) {
  cons = constraint.toString();

  if (RecipientsAdapter.usefulAsDigits(cons)) {
    phone = PhoneNumberUtils.convertKeypadLettersToDigits(cons);
    if (phone.equals(cons) && !PhoneNumberUtils.isWellFormedSmsAddress(phone)) {
      phone = "";
    } else {
      phone = phone.trim();
    }
  }
}
Uri uri = Uri.withAppendedPath(Phone.CONTENT_FILTER_URI, Uri.encode(cons));
String selection = String.format("%s=%s OR %s=%s OR %s=%s",
                                 Phone.TYPE,
                                 Phone.TYPE_MOBILE,
                                 Phone.TYPE,
                                 Phone.TYPE_WORK_MOBILE,
                                 Phone.TYPE,
                                 Phone.TYPE_MMS);

Cursor phoneCursor = mContentResolver.query(uri,
                                            PROJECTION_PHONE,
                                            null,
                                            null,
                                            SORT_ORDER);

if (phone.length() > 0) {
```

```
      ArrayList result = new ArrayList();
      result.add(Integer.valueOf(-1));                    // ID
      result.add(Long.valueOf(-1));                       // CONTACT_ID
      result.add(Integer.valueOf(Phone.TYPE_CUSTOM));     // TYPE
      result.add(phone);                                  // NUMBER
      result.add("\u00A0");                               // LABEL
      result.add(cons);                                   // NAME

      ArrayList<ArrayList> wrap = new ArrayList<ArrayList>();
      wrap.add(result);

      ArrayListCursor translated = new ArrayListCursor(PROJECTION_PHONE, wrap);

      return new MergeCursor(new Cursor[] { translated, phoneCursor });
    } else {
      return phoneCursor;
    }
  }
```

The sort order string should be updated to handle lower and upper case names. A new method should be implemented which resorts the contacts names. This method can be added in the following code to keep the implementation simple and avoid time consuming debugging.

6 Conclusion

For this assignment, the SCRUM model approach worked very well. The group has learned a lot from its past inaccuracies and aimed to improve them this time. In the last assignment, the team was unable to keep up with the product backlog items and as a result, it took longer to finish the implementation than estimated. So, we focused more on following the timeline as shown in Table 1. This dramatically improved our work with minimum inaccuracies and helped each team member to efficiently approach the given tasks.

While identifying the problem, goal model and domain model provided a great way to keep track of necessary steps which gave the group a defined path to follow during the project. By estimating the effort and analyzing the risk associated with the project, it was easier to set realistic deadlines and results. Further, having built use cases helped the team to clearly understand and explain the users perspective while using the feature, and visually describe different circumstances. Finally, by carrying out a robustness analysis, it became easier to track and note how the functionality of the project was to be met. In conclusion, the above mentioned models and analysis assisted the group to set up the project and seamlessly construct a design from the given requirements.

7 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] Person, "Sort contacts by alphabet case insensitive," Signal Community, 10-Jan-2020. [Online]. Available: https://community.signalusers.org/t/sort-contacts-by-alphabet-case-insensitive/11139. [Accessed: 14-Nov-2020].

[3] 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].

[4] P. Spachos, Class Lecture , Topic: "Lecture 9 - Estimation and Prioritization", College of Engineering and Physical Sciences, University of Guelph, Guelph,ON.

A Appendix

A.1 Development Process

Figure 7: Visualization of SCRUM Process

A.2 Survey Results

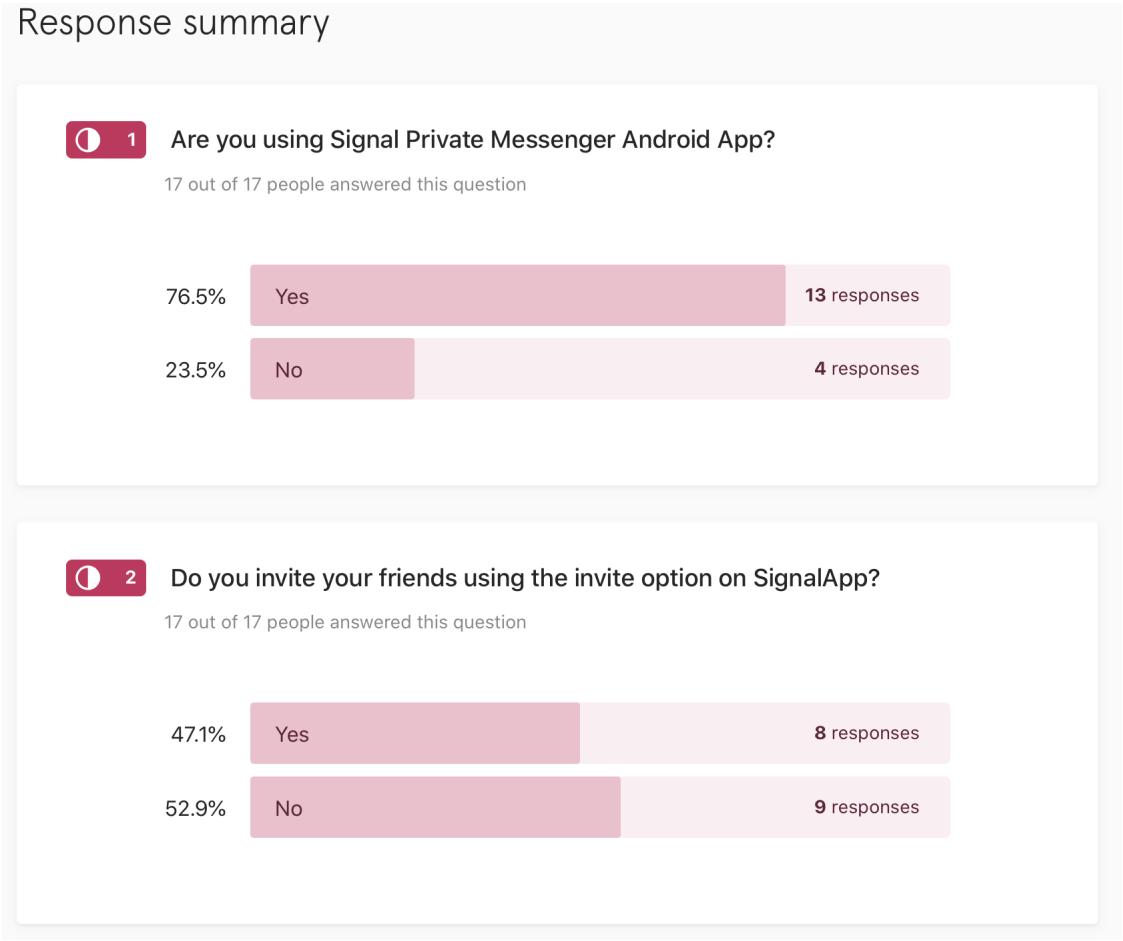

Figure 8: Survey Result 1 of 2

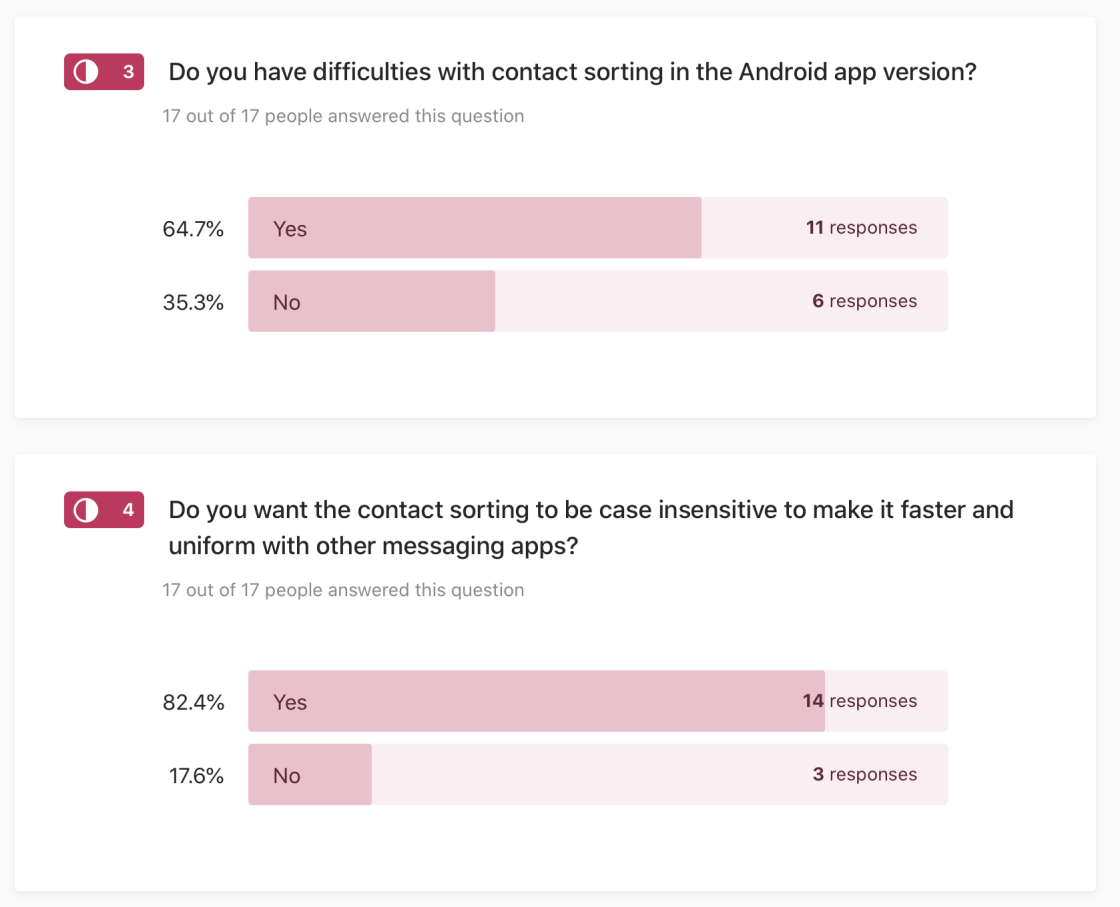

Figure 9: Survey Result 2 of 2

A.3 Current Sorting by Alphabet Case-sensitive

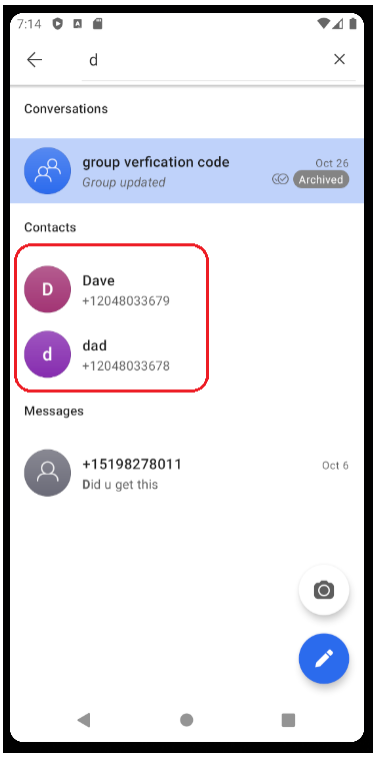

Figure 10: Current Contacts Sorting