Assignment 1

For Zahra Shakeri

BenKun Chen————Student ID: 30005337

Jian Li————————— Student ID: 30023214

Quang Minh Tran——Student ID: 30017773

1 Deliverable 1

At first, we discuss the advantages, the disadvantages and the appropriate cases to use for each software process model.

- Opportunistic model
  - Advantages
    * Give programmer almost complete control over the development
    * Not too procedural to follow
  - Disadvantages
    * No plan; no control of cost or schedule; hard to maintain; the cost of deploying and maintaining is high
    * Code first version of system based on their understanding of the product
    * Since it keep on modifying until satisfying so it is hard with high level of user's satisfaction
    * You might not understand all of user requirement
    * No formal process of testing or QA
    * Developer might be only given a set of minimal requirements
    * They are approaching the project deadline
    * Communication channels are fail and there is no or little business stakeholders
  - When to use
    * For simple project and quality is not important
- Waterfall
  - Advantages
    * Easy to understand and implement.
    * Widely used and known
    * Being a linear model, it is very simple to implement
    * Works well on mature products and provides structure to inexperienced teams

    - Minimizes planning overhead
    - Phases are processed and completed one at a
  - Disadvantages
    - All requirements must be known upfront
    - Inflexible
    - Backing up to solve mistakes is difficult, once an application is in the testing stage, it is very difficult to go back and change something that was not well-thought out in the concept stage.
    - A non-documentation deliverable only produced at the final phase.
    - Client may not be clear about what they want and what is needed
    - Customers may have little opportunity to preview the system until it may be too late
    - It is not a preferred model for complex and object-oriented projects
    - High amounts of risk and uncertainty, thus, small changes or errors that arise in the completed software may cause a lot of problems
  - When to use
    - When quality is more important
    - When requirements are very well known, clear, and fixed
    - New version of existing product is needed

- Spiral
  - Advantages
    - High amount of risk analysis
    - Software is produced early in the software life cycle
    - Strong approval and documentation control
    - Additional functionality can be added at a later date
    - Project monitoring is very easy and effective
    - Concerned people of a project can early review each phase and each loop as well because of rapid prototyping tools
    - Early and frequent feedback from users
    - Suitable to develop a highly customized product
    - Provides early indication of insurmountable risks.

  - Disadvantages
    * Cost involved in this model is usually high
    * Risk assessment expertise is required
    * Amount documentation required in intermediate stages makes management of a project very complex
    * Time spent for evaluating risks for small or low-risk projects may be too large
    * Time spent for planning, resetting objectives, doing risk analysis, and prototyping may be excessive
    * Projects success is highly dependent on the risk analysis phase.
  - When to use
    * For medium to highrisk projects
    * When risk evaluation and costs are important
    * When significant changes are expected
    * When users are not exactly sure what their needs

- Concurrent
  - Advantages
    * Divide and conquer
    * Team works on different projects
  - Disadvantages
    * Components don't integrate properly
    * Component might break each other
    * Component might become obsolete if communication is bad
  - When to use
    * For large or open source projects

- Scrum
  - Advantages
    * Decision-making is entirely in the hands of the teams

    - This methodology enables projects where the business requirements documentation is not considered very significant for the successful development
    - It is a lightly controlled method which totally empathizes on frequent updating of the progress, therefore, project development steps is visible in this method
    - A daily meeting easily helps the developer to make it possible to measure individual productivity. This leads to the improvement in the productivity of each of the team members

  - Disadvantages
    - This kind of development model is suffered if the estimating project costs and time will not be accurate
    - It is good for small, fast moving projects but not suitable for large size projects
    - This methodology needs experienced team members only. If the team consists of people who are novices, the project cannot be completed within exact time frame

  - When to use
    - Scrum process focuses on delivering the highest quality output, in the shortest period of time. This process is suited for development projects that are rapidly changing or highly emergent requirements
    - Mainly for project with comlex technology
    - Scrum is applicable to both small and large-scale projects

From our team situation, all of us are new to software development process and the implemented project is not so complicated. Moreover, the requirements are clear since we check carefully with customers. Thus we choose **Waterfal** model for guiding us the process step by step and we all have clear requirements.

2 Deliverable 2

Main goal and functionality from customer: **This app enables the automation of anything on your phone, example: when you connect to your house Wi-Fi, run a sync app or script. It should be able to launch any app on the phone, with URI support to pass-through arguments. The rules for launch apps/scripts should include basic Boolean logic (AND/OR/NOT/XOR/NOR) with unlimited entries. These rules should also be exportable and importable, and preferably be editable by any text editor. The product should be open source, as other versions lock out features and eventually die. This app is useful in everyday life as you can program it to atomize tasks/actions that you would do repeatedly.**

- Functional requirement of the project
  - User should be able to create condition as wi-fi connection, sync , timer or low battery-level.
  - User should be able to create logic-condition to be logic term (AND/OR/NOT/XOR/NOR) of some condition.
  - User should be able to create operation as launching any app/script or URI.
  - User should be able to create a relation between specified logic-condition to specified operation.
  - User should be able to deactivated a relation.
  - User can export rules (set of relations, logic-conditions and operations) to a text file.
  - User can import text file that contains rules into app.
- Non Functional requirement of the project
  - System should be able to detect if condition is satisfied.
  - System should be able to detect if logic-condition is satisfied.
  - System should activated a relation once created.
  - System should be able to execute some operation when logic-condition related is satisfied.
  - Number of condition for one logic-condition is unlimited.
  - App is uploaded to share-platform

3 Deliverable 3

3.1 Choosing use cases and use case diagram

We choose use cases and a use case diagram to represent our system's requirements since they clearly describe the steps in an activity in natural language, they are easy to understand and provide an excellent way for communicating with our customers. By using use cases, we can easily define the business rules and the options that the actor will have available when executing the activity. Also, use cases distinguish the extensions and inclusions in order to make functional and nonfunctional interactions explicit. Using use cases not only help us analyzing the activity and producing the details of generalization, but it also help ensure that the correct system is developed by capturing the requirements from the user point of view.

In contrast, to represent our system requirements by using user stories and a story map is not as productive as use cases since its lack of visual ability. The user stories often leave out a lot of details, they are deliberately abstract early on in a project and time consuming when we spend most of our time to solve confusions between us and the customer.

Our group is looking for the method that is both timesaving and comfortably intelligible to represent our systems requirements, the use cases is the better method over the story map after a thoughtful comparison.

3.2 Use cases

3.2.1 Use case for creating condition

Use case name:	Create condition
Goal/description:	Create condition include wi-fi, time etc.
Related use cases:	Generalization of - Create condition as wi-fi condition - Create condition as time condition
Actor:	App user
Precondition:	At the main activity
Steps/flow:	
Actor actions:	System responses:
1. Tap “create condition” button	2. Choices dialog appears
3. Tap one of choices	4. Attribute text fields show
5. Fill in the text field	
6a. Tap “confirm” to create	
6b. Tap “cancel” to cancel create	7. Back to main activity
Postcondition:	Condition is created

3.2.2 Use case for creating condition with wifi condition

Use case name:	Create condition as wi-fi condition
Goal/description:	Set condition as wi-fi
Related use cases:	Specialization of Create condition
Actor:	App user
Precondition:	At the create condition dialog
Steps/flow:	
Actor actions:	System responses:
1. Tap “create condition” button	2. Choices dialog appears
3. Tap “wi-fi condition”	4. Attribute text fields show
5. Fill in the text field	
6a. Tap “confirm” to create	
6b. Tap “cancel” to cancel create	7. Back to main activity
Postcondition:	Condition is created

3.2.3 Use case for creating condition with time condition

Use case name:	Create condition as time condition
Goal/description:	Set condition as time
Related use cases:	Specialization of Create condition
Actor:	App user
Precondition:	At the create condition dialog
Steps/flow:	
Actor actions:	System responses:
1. Tap “create condition” button	2. Choices dialog appears
3. Tap “time condition”	4. Attribute text fields show
5. Fill in the text field	
6a. Tap “confirm” to create	
6b. Tap “cancel” to cancel create	7. Back to main activity
Postcondition:	Condition is created

3.2.4 Use case for creating logic condition

Use case name:	Create logic-condition
Goal/description:	Create logic-condition by logic operation on conditions
Related use cases:	
Actor:	App user
Precondition:	At the main activity
Steps/flow:	
Actor actions:	System responses:
1. Tap “create logic-condition” button	2. Dialog appears with attribute text fields show
3. Fill in the text field	
4a. Tap “confirm” to create	
4b. Tap “cancel” to cancel create	5. Back to main activity
Postcondition:	Logic-condition is created

3.2.5 Use case for creating an operation

Use case name:	Create an operation
Goal/description:	Create an operation as launching any app/script or URI
Related use cases:	
Actor:	App user
Precondition:	At the main activity
Steps/flow:	
Actor actions:	System responses:
1. Tap “create operation” button	2. Choices dialog appears with a list of operation
3. Tap one of choices	4. Attribute text fields show
5. Fill in the text field	
6a. Tap “confirm” to create	
6b. Tap “cancel” to cancel create	7. Back to main activity
Postcondition:	Operation is created

3.2.6 Use case for creating a relation

Use case name:	Create a relation
Goal/description:	Create a relation between specified logic-condition to specified operation
Related use cases:	
Actor:	App user
Precondition:	At the main activity
Steps/flow:	
Actor actions:	System responses:
1. Tap “relate” button	2. Choices dialog appears with a list of logic-condition
3. Tap one of choices	4. List is replaced by list of operation
5a. Tap “confirm” to create	
5b. Tap “cancel” to cancel create	6. Back to main activity
Postcondition:	Relation is created

3.2.7 Use case for deactivating a relation

Use case name:	Deactivate a relation
Goal/description:	Deactivate a relation that is activated
Related use cases:	
Actor:	App user
Precondition:	At the main activity and has at least one relation
Steps/flow:	
Actor actions:	System responses:
1. Tap “deactivated” button	2. Item associated with button in list turn grey
Postcondition:	A relation is deactivated

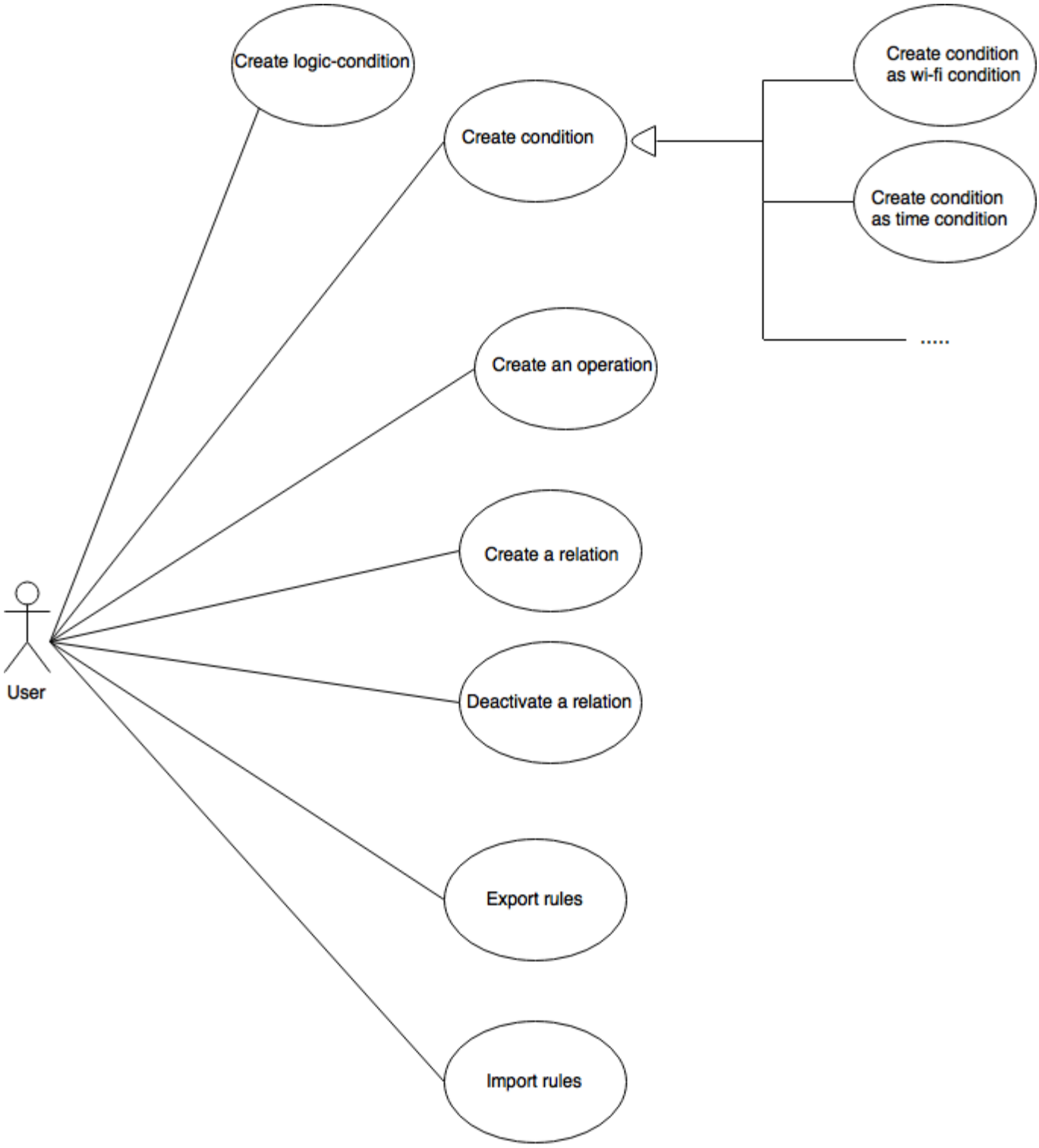
3.2.8 Use case for exporting rules

Use case name:	Export rules
Goal/description:	Export rules (relations, logic-conditions and operations) to a text file
Related use cases:	
Actor:	App user
Precondition:	At the main activity
Steps/flow:	
Actor actions:	System responses:
1. Tap “export” button	2. Save all rules to a text file in default directory
	3a. Show successful message 3b. Show unsuccessful message
Postcondition:	File contains rules exported

3.2.9 Use case for importing rules

Use case name:	Import rules
Goal/description:	Import rules (relations, logic-conditions and operations) from a text file
Related use cases:	
Actor:	App user
Precondition:	At the main activity
Steps/flow:	
Actor actions:	System responses:
1. Tap “import” button	2. Import rules from text file in default directory
	3a. Show successful message 3b. Show unsuccessful message
Postcondition:	Rules imported

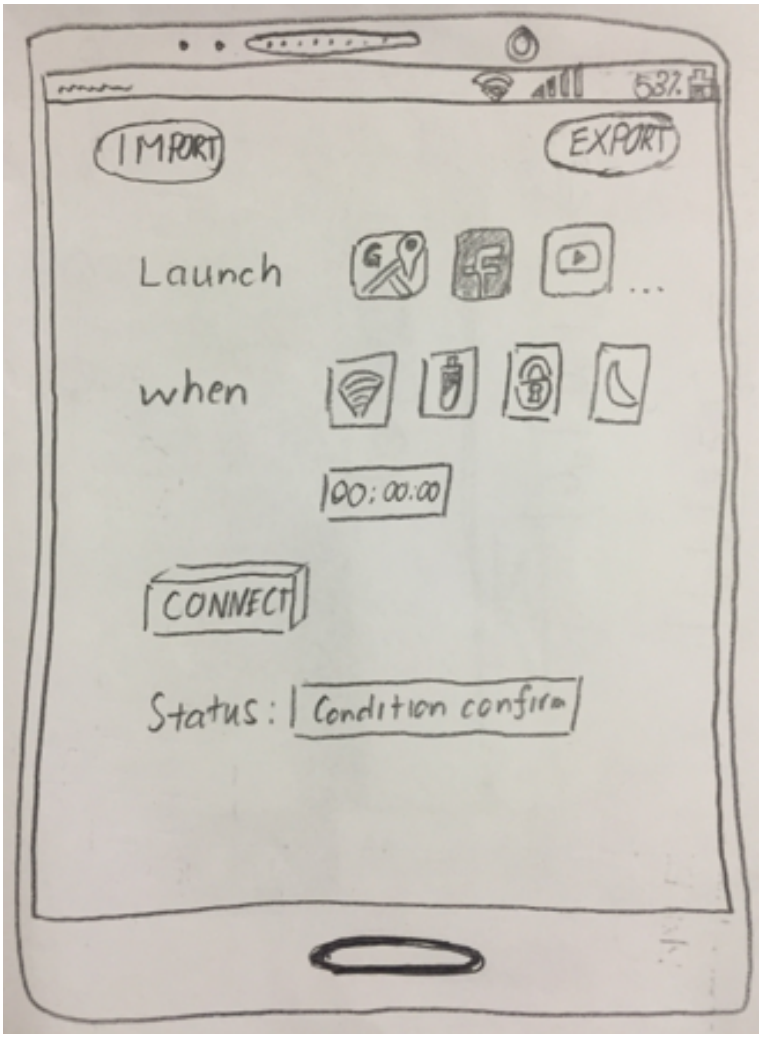
3.3 The diagram for the usecase is:



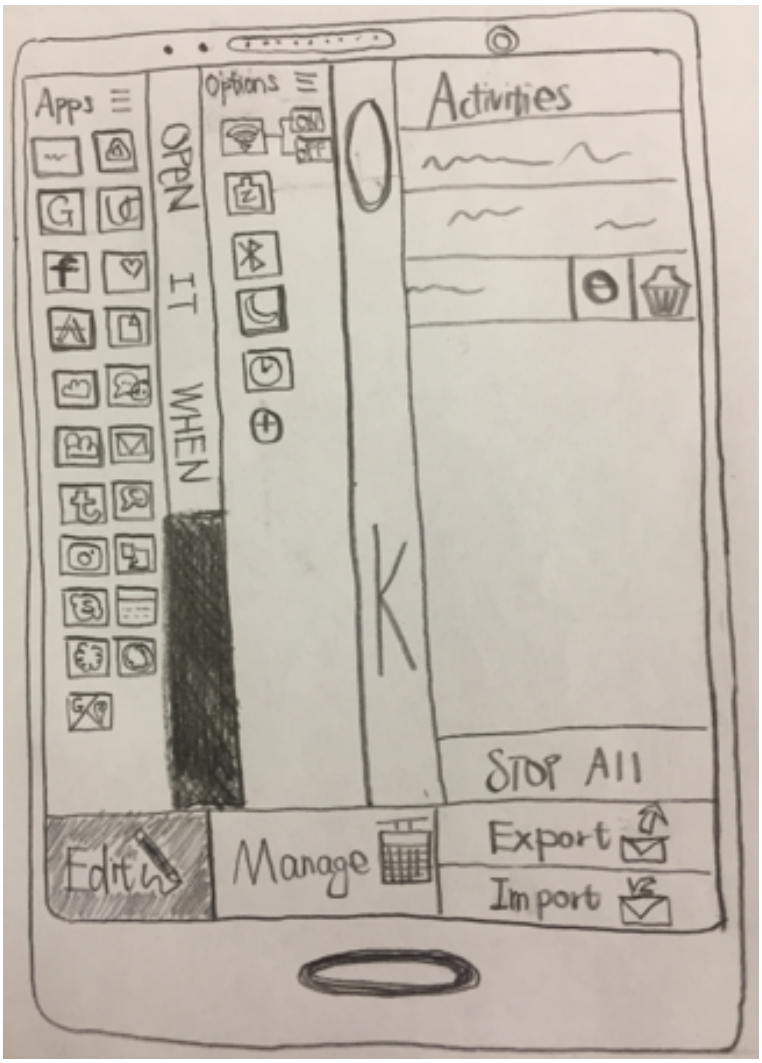
4 Deliverable 4

4.1 Sketches

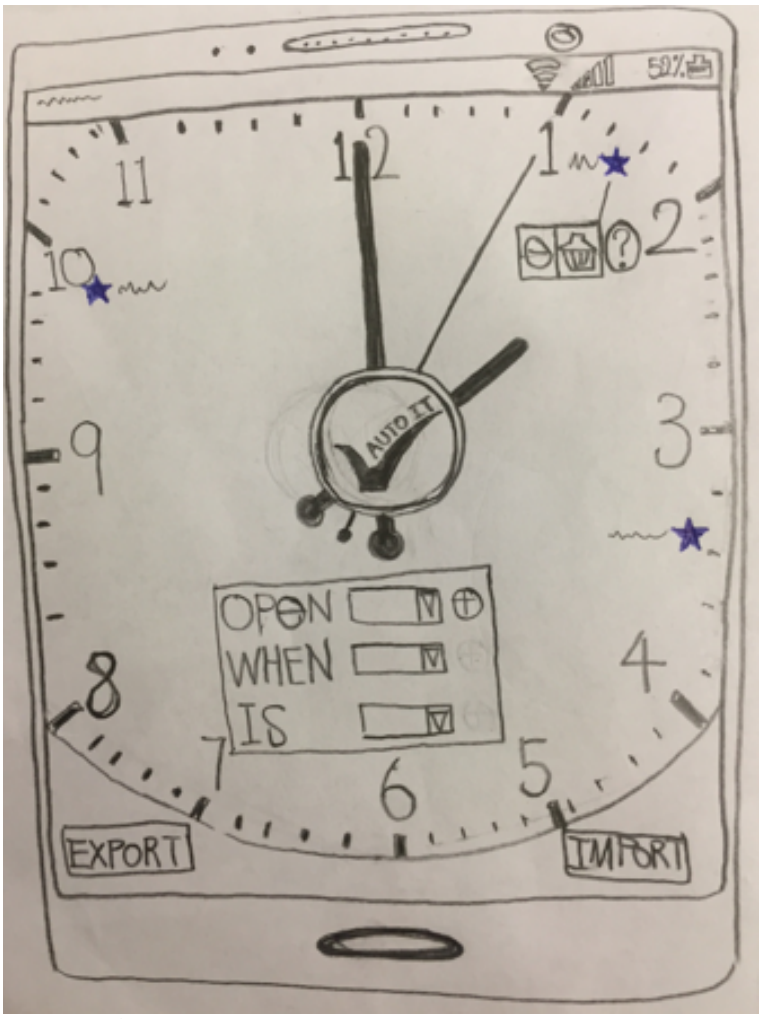
4.1.1 5 different sketched interfaces for choosing



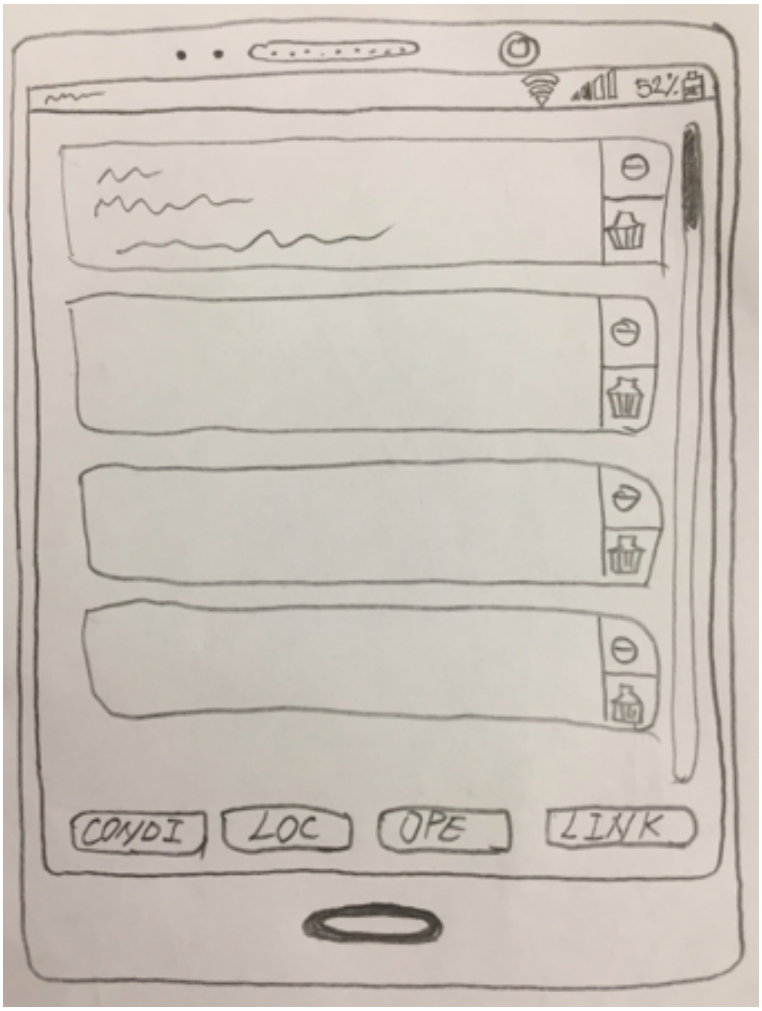
- Advantages
  - Simple and neat user interface such that user will be easy to access
- Disadvantages
  - It does not have the software title which can confuse customer
  - There is no logic gates



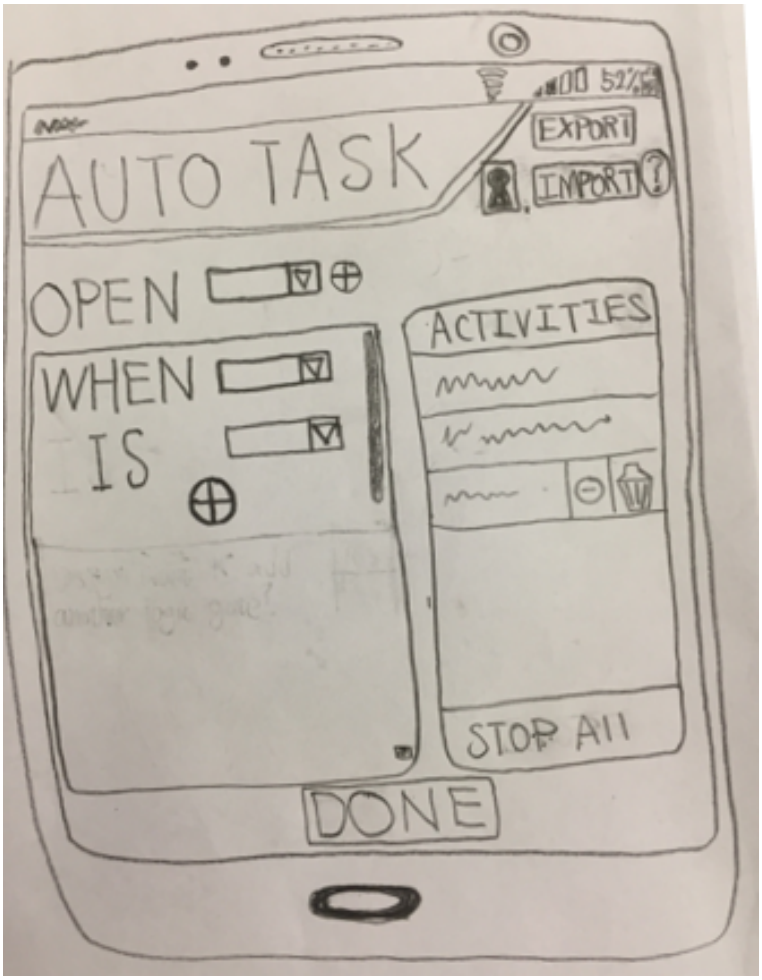
- Advantages
  - Good at visual presentation
  - Split the screen into Edit and Manage page
- Disadvantages
  - Hard to implement in Android system
  - Too constraint because there are too many icons



- Advantages
  - User can clearly see the time for a certain operation is going to run
- Disadvantages
  - Limited condition setting (only timer)
  - Waste of memory since we need to run animation

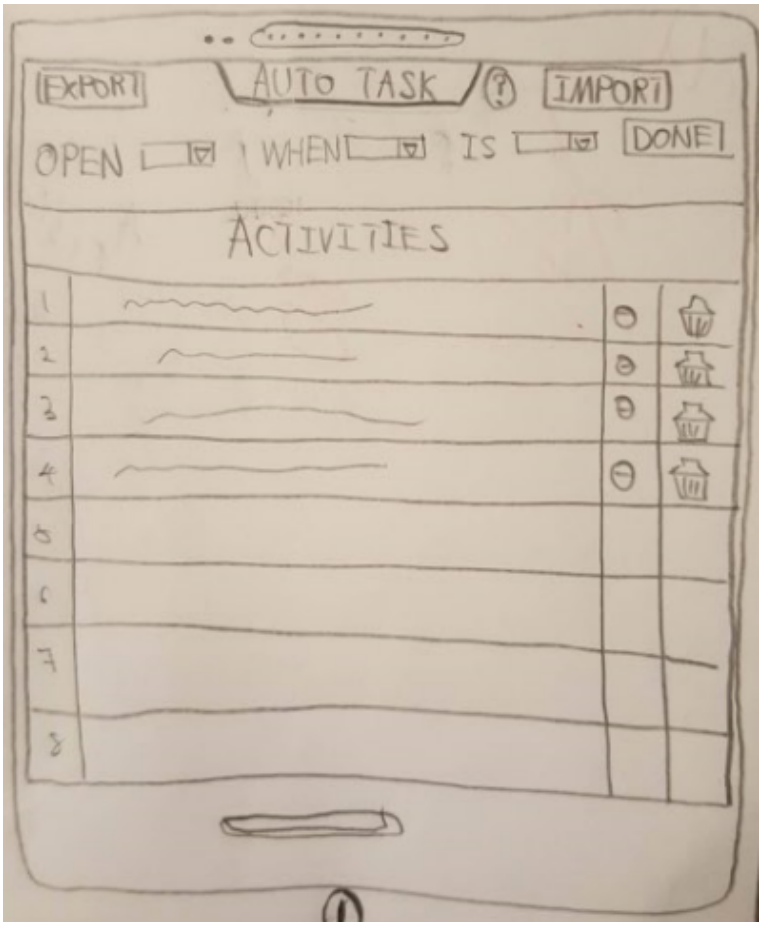


- Advantages
  - Fully functional, good for professional user
- Disadvantages
  - Lack of user-friendly property

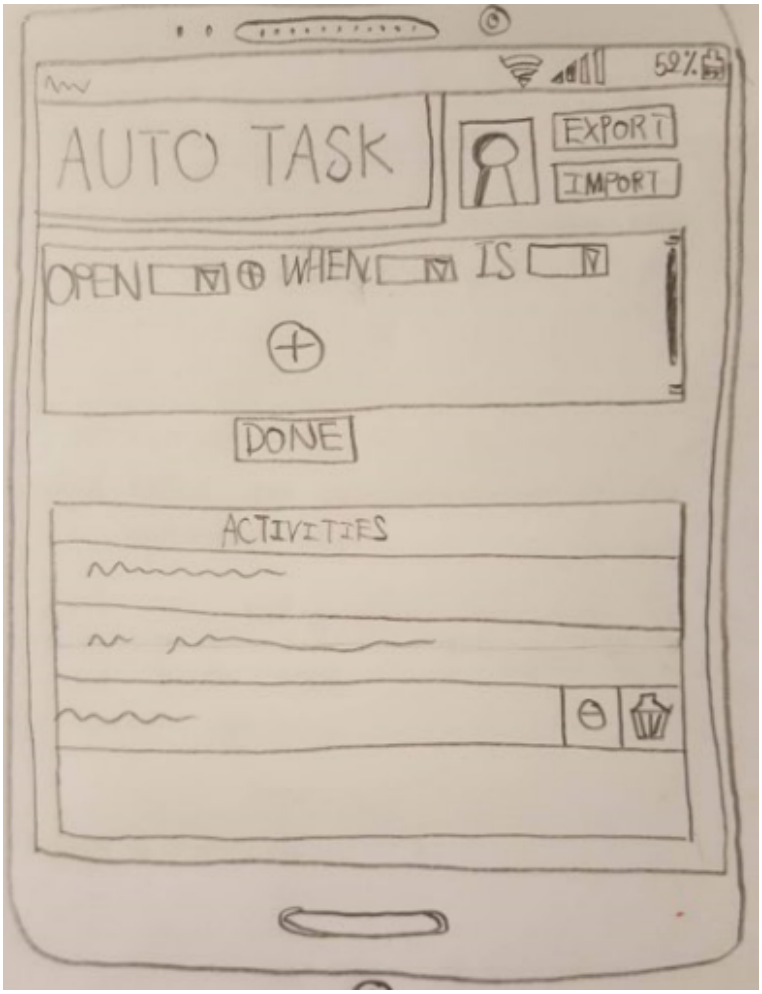


- Advantages
  - User friendly
  - User can achieve all requirements by simply using just three steps
  - User can manage and modify any requirements which they have been implemented
- Disadvantages
  - Intermidiate programing level requirement for our group

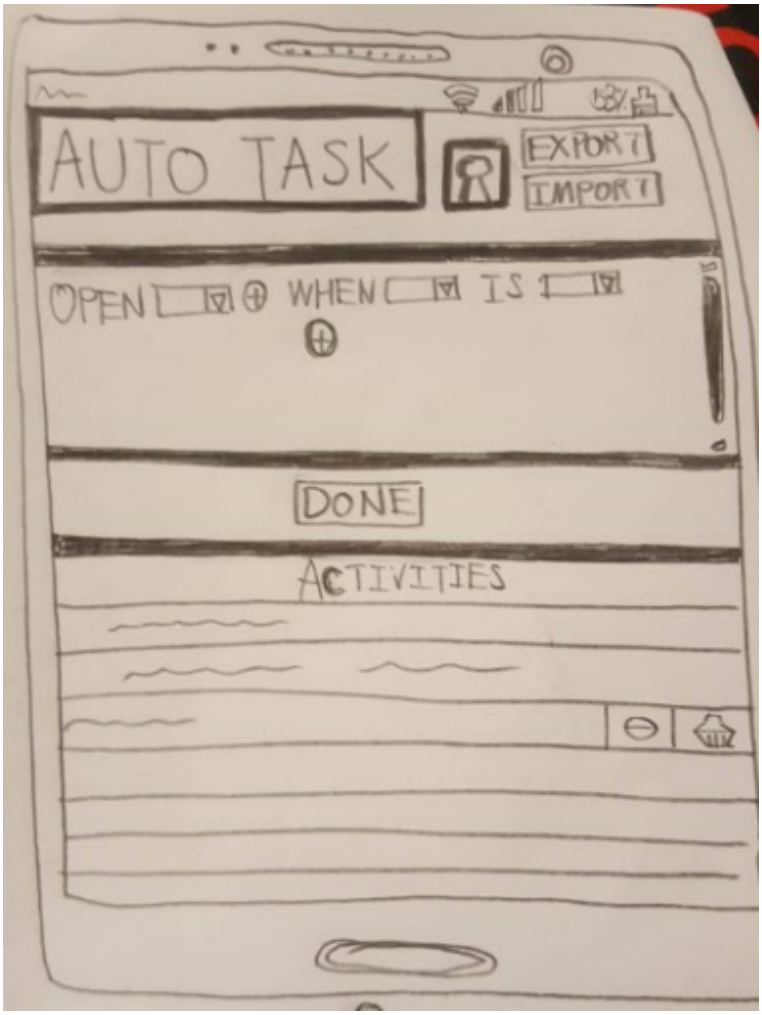
4.1.2 Detail sketches to elaborate



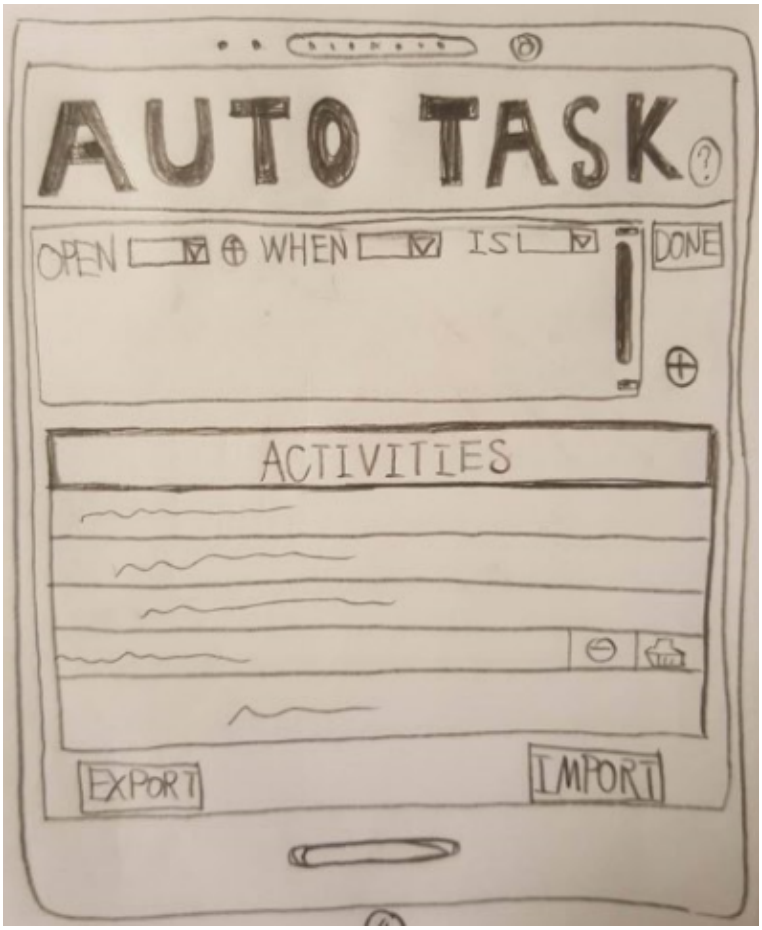
- This sketch has a huge space showing activity, which is good for a tool app.
- But the space for generate activity is not attractive enough.
- And deactivate and delete button might occupy too much space for each activity.



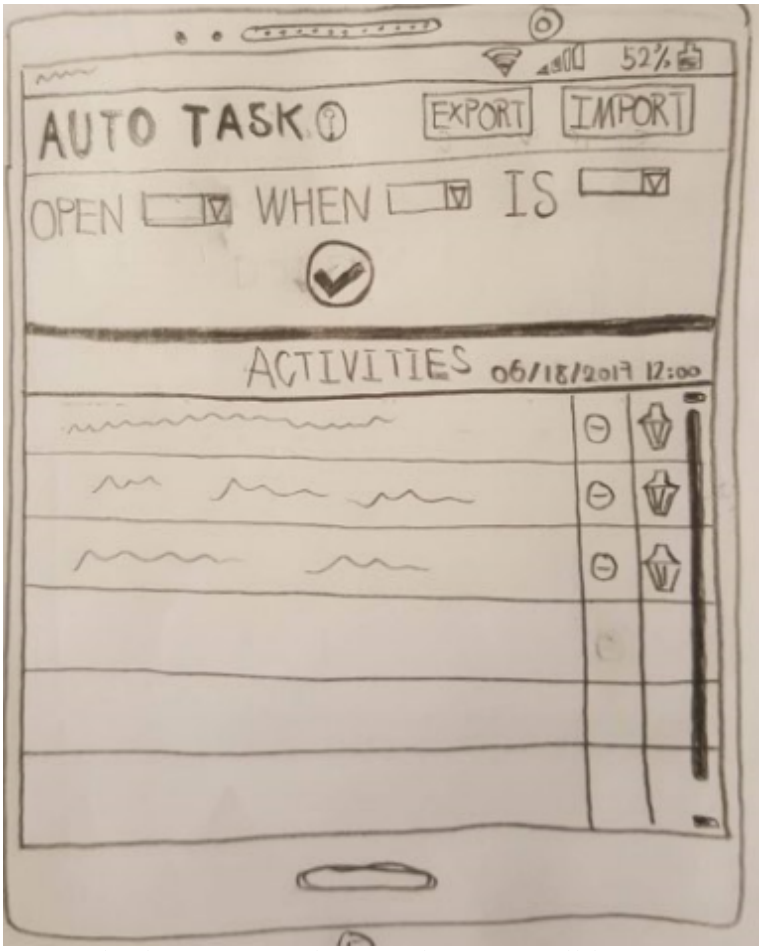
- Compare to 1st one, ”auto task” is enhanced and a profile image is added just beside logo.
- The add-function button is added inside of generate box.



- The third one has enhancement for every section.



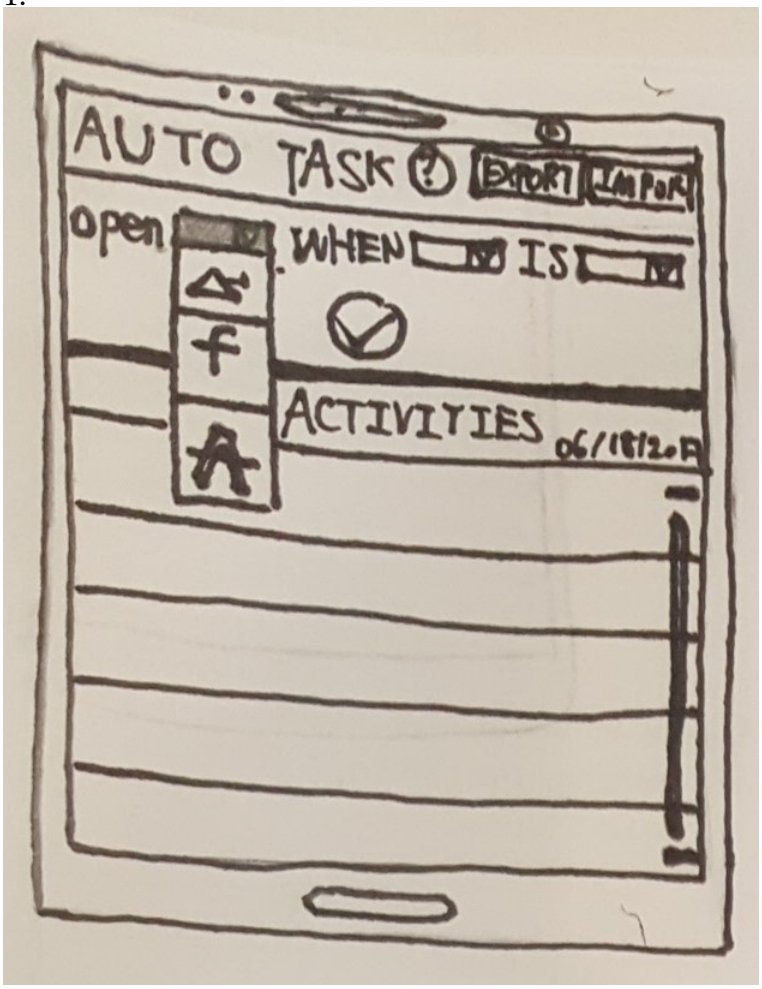
- The profile image is removed in 4th sketch.
- Export and import button are rearranged to bottom of screen.



- The last version has a relative small logo
- And export and import button are rearranged to right of logo since these button are not heavily used should be placed to corner.

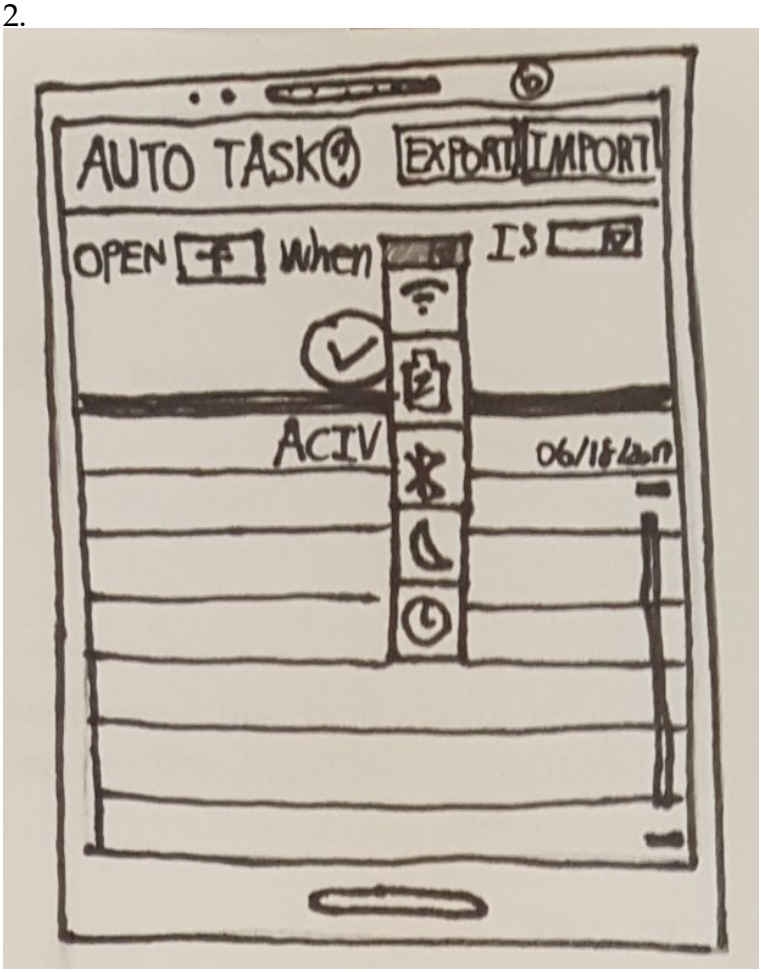
4.2 Storyboard

1.



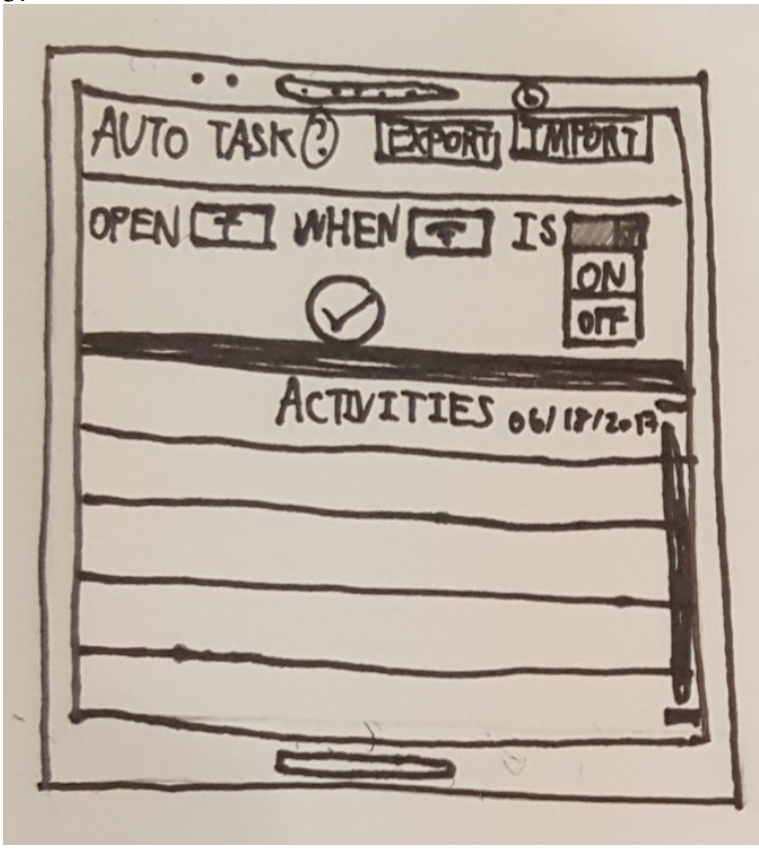
- User first start constructing operation by clicking the ”drop bar” besides the open. The ”drop bar” display all available applications on the cell phone

2.



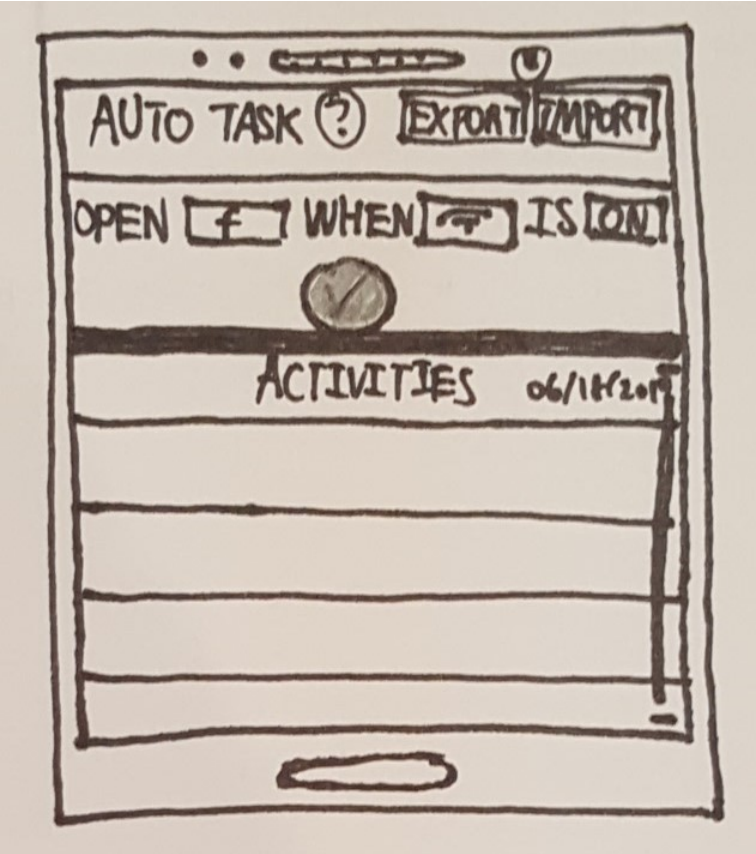
- After user have chosen the applications they want for the operation, they will then choose the conditions by clicking the ”drop bar” besides ”WHEN”.

3.



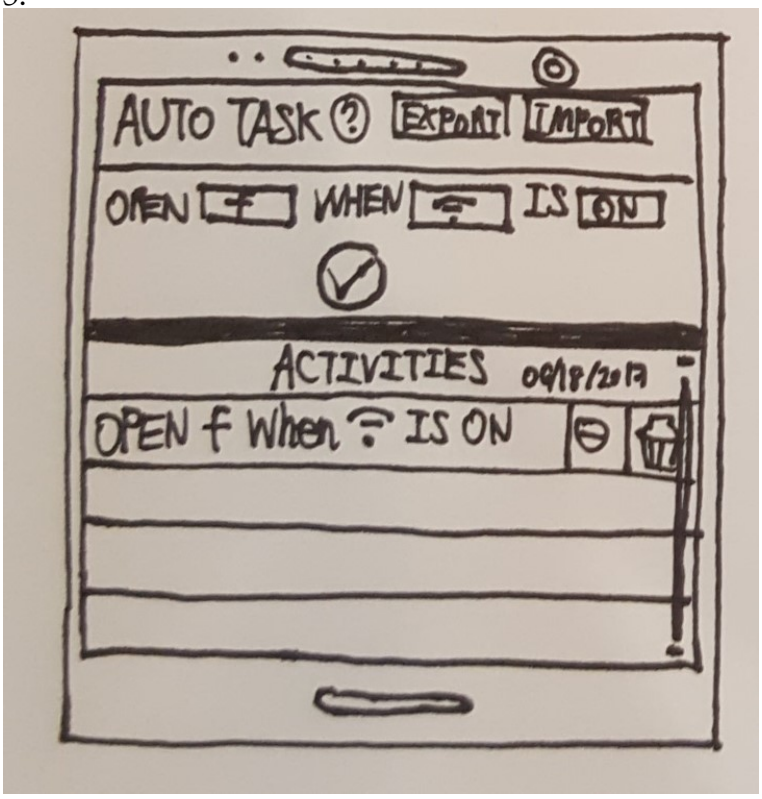
- After user have chosen the conditions they want under running applications for the operation, they will then choose the secondary conditions by clicking the ”drop bar” besides ”IS”. Options under the drop bar will depend on the first condition(s) that user chose.

4.



- After user have finish the operation, simply click the check mark shown in the diagram.

5.



- After user generate the activity, it shows in activity list and activated automatically. And also, user can deactivate, activate and delete each activity.

4.3 Wizard of Oz

`https://youtu.be/-CDmjB_qh34`

4.4 Requirements change

With those mentioned methods such as the sketches, the storyboard and the wizard of OZ, these prototypings is an integral part of iterative user-centered design because it enables us to try out the ideas with customers and to gather feedback. After discussing with customer, we put more detail in the use cases for example adding in the successful or unsuccessful message when the user can import or export the rules. Moreover, by having

these prototypes, we understand more about the requirements and are more certain about the requirements which is the most important part of Waterfall model.

5 Detail of collaboration

5.1 Meeting on Thursday 25th May, 2017 from 4pm to 4:30pm

Attendance: P10-Supplier(Jian, Ben, Quang) and P15-Customer(Scott, Justine, Payal)

Goal of meeting: Clarify the functional and non-functional requirement for the project as the first step in Waterfall model

Meeting minute: -Clarification for functional requirement and non-functional requirement which are:

- Creating an app to run another app (one of method might be using URI) by the condition setting for example when there is wifi signal or bluetooth transaction.
- Create the logic connection of the condition with the app for example run the app when there is wifi connection and battery is more than half
- Non-functional requirement is to put the soure code on the git hub

5.2 Meeting minute on Monday 29th May, 2017 from 2pm to 2:20pm

Attendance: P10-Supplier(Jian, Ben, Quang) and P15-Customer(Justine, Payal)

Goal of meeting: Show to customer the use case models (9 cases) for approval

Meeting minute:
-Cutomers approve the proposed use cases.

5.3 Meeting minute on Thursday 1st June, 2017 from 10:40am to 10:50am

Attendance: P10-Supplier(Jian, Ben, Quang) and P15-Customer(Justine, Payal)

Goal of meeting: Show to customer the user interfaces for approval

Meeting minute:
-Cutomers approve the userinterface.