Text-based conversational interface as an alternative to a crowdsensing mobile application

Master's Thesis

Neha Sree Thuraka

Text-based conversational interface as an alternative to a crowdsensing mobile application

THESIS

submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE
TRACK DATA SCIENCE TECHNOLOGY

by

Neha Sree Thuraka
born in India

Web Information Systems
Department of Software Technology
Faculty EEMCS, Delft University of Technology
Delft, the Netherlands
http://wis.ewi.tudelft.nl

Text-based conversational interface as an alternative to a crowdsensing mobile application

Author: Neha Sree Thuraka
Student id: 4743598
Email: nehasreet@gmail.com

Abstract

Crowdsensing is a powerful tool to easily sense diverse physical environments by collecting data from an undefined network of people. With advancements in smartphone technology, there has been an increase in the use of mobile applications to perform crowdsensing tasks. However, previous work shows that mobile applications have issues with attracting and retaining users, thus limiting the utility of crowdsensing as a data collection technique. To mitigate these issues, we propose the use of conversational agents (chatbots) as an alternative to custom mobile applications for crowdsensing applications. We hypothesize that the use of commonly used text-based applications (e.g., Telegram) enriched with the automated conversational capabilities can increase the attraction and retention of crowdsensing participants.

In this thesis, we designed and implemented a crowdsensing system that supports the execution of mobile and chatbot interface. We propose a design of the text-based conversational interface that provides different elements and features of a traditional mobile application. To compare these two interfaces for performing crowdsensing tasks and to understand the differences in terms of user engagement and usability, we conducted two experiments on the TU Delft campus with students as the participants. Based on the location of the experiment, we designed four task domains and three types of tasks.

In the first experiment, we organized a 'between-subjects' study. We recruited 80 students to analyze user engagement and usability in a quantitative fashion. The experiment shows that chatbot has better user engagement and usability than the mobile application. We conducted a qualitative survey to understand the underlying reasons behind the participation patterns. Analysis of the results of this survey shows that the unavailability of the participants and the assignment of inappropriate tasks are the main reasons behind non-participation of some students.

To deepen our analysis, we organized the second experiment as a 'within-subjects' study with 10 participants in a controlled environment. The

experiment shows that all participants unanimously preferred chatbot over the mobile application to perform crowdsensing tasks.

As a result of both experiments, we conclude that the text-based conversational interface can be used as an alternative to the mobile application to execute crowdsensing tasks and the former is more engaging than a mobile application interface for crowdsensing applications.

Thesis Committee:

Chair:	Prof. dr. ir. Alessandro Bozzon, Faculty EEMCS, TUDelft
University supervisor:	Prof. dr. ir. Alessandro Bozzon, Faculty EEMCS, TUDelft
Committee Member:	Dr. Christoph Lofi, Faculty EEMCS, TUDelft
Committee Member:	Dr. Huijuan Wang, Faculty EEMCS, TUDelft

Preface

Before you lies the dissertation *'Text-based conversational interface as an alternative to a crowdsensing mobile application'*, that concludes my MSc at TU Delft. This document has been written in partial fulfillment of the graduation requirements of the MSc Computer Science programme, Data Science & Technology Track. This journey taught me many things and gave me the experience of conducting scientific research. This thesis would not have been possible without the support and guidance of several people whom I would like to thank.

I would like to thank Prof. dr. ir. Alessandro Bozzon for giving me this opportunity and for his immense support at each step. I would also like to thank Sihang Qiu for always being willing to help me throughout the project and all the members of Sigma team for their comments and suggestions. I am grateful to other thesis committee members, Dr. Christoph Lofi and Dr. Huijuan Wang for the taking the time to attend my thesis defense and Prof. dr. ir. Geert-Jan Houben for the inspiration. Last but not least I would like to thank my friends, fellow students and parents for their moral support.

Neha Sree Thuraka
Delft, the Netherlands
August 19, 2019

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent years, crowdsourcing has shown potential as a technique to ease the access to knowledge and resources of the crowd. Crowdsourcing uses the collective intelligence of a large group of people to help solve a problem. This technique helps in gathering high volumes of data for relatively low cost [13]. This lead to the explosion of its applications in diverse areas such as bioinformatics [18, 20, 5, 32], zoology [31] and astronomy [29] to name a few. In this context, one of the most sought after research areas is mobile crowdsensing.

Crowdsensing refers to the act of collecting data from a possibly undefined network of people to sense diverse environments. These sensor devices are typically equipped with wireless capabilities that allow them to produce data and upload it to the internet. Based on the level of involvement of the user, crowdsensing can be classified into *participatory* and *opportunistic*. While participatory sensing requires the users to explicitly provide information, opportunistic sensing requires minimal involvement of the user and is mostly autonomous. For the success of a crowdsensing application, especially the ones with participatory sensing, it should be able to recruit, engage and retain the users for a longer duration [8].

To increase the accessibility and thus participation of users, personal smartphones have been used to implement crowdsensing as a mobile application. But, current mobile crowdsensing literature that uses mobile applications has mostly ignored the importance of human factor which is one of the main reasons behind the success of an application [26]. Though there is some previous research on motivating users by incentivizing [26] them, there is no research that concentrates on engagement of the crowdsensing application users. User engagement is important for a mobile crowdsensing sensing system to maintain adequate user participation, to collect adequate data and to ensure the desired output quality. But in reality these applications have inadequate user retainment as users may not be willing to participate for a longer duration [9].

Moreover, a smartphone user does not use more than 27 apps per month despite the increase in the number of choices [25]. This number is less considering the availability of more than 2 billion apps there out in the market. So, there is a need to search for an alternative way that retains, engages users for a longer duration. We hypothesize that the use of commonly used text-based applications (e.g., Telegram) enriched with the automated conversational capabilities can increase the attraction and retention of

crowdsensing participants.

Conversational agents (chatbots) have gained immense popularity in recent years, especially because of their availability on popular messaging applications such as Telegram and Facebook Messenger as the user does not have to download an additional application to execute crowdsensing tasks. This helps in attracting more users for using the conversational interfaces as these messaging applications already have millions of users who are very familiar with the interface and other functionalities [12]. It also retains users by engaging them through mimicking human-like conversation [2]. Moreover, a recent research shows that the conversational interface is suitable for microtask crowdsourcing and also chatbot provides better user satisfaction than a web interface [21]. So, in this research, we explore the usage of the conversational interface as an alternative to the mobile application to execute crowdsensing tasks.

1.1 Research questions

The goal of this research is to explore the usage of a text-based conversational interface as an alternative to a mobile application for performing crowdsensing tasks. Therefore, the focal point of this research is to answer the following main research question:

How can a text-based conversational interface be used as an alternative to a mobile application to execute crowdsensing tasks?

To answer this question, we organized the research around the following three sub-questions:

RSQ 1: What is the state-of-art in mobile crowdsensing?
Before we proceed further to explore the possibility of doing a crowdsensing task through the text-based conversational interface, it is necessary to look into the literature related to crowdsensing. We aim to understand how crowdsensing system works and also how a mobile application interface is currently used to perform these tasks. We analyze the limitations of current crowdsensing applications that use mobile application interface and also explore how a text-based conversational agent can overcome them.

RSQ 2: How can a text-based conversational interface be designed to provide different elements and features of a traditional mobile application?
To check if a text-based conversational interface can be used as an alternative to a mobile application for executing crowdsensing tasks, it is important to be able to perform the same tasks under equivalent circumstances through both interfaces. The objective here is to study how the user interaction and features that are typical of mobile crowdsensing interfaces can be mapped to text-based conversational actions.

RSQ 3: How does the conversational interface affect user engagement and usability in a crowdsensing application?
After designing a chatbot by mapping UI elements and features of a traditional mobile, we propose a crowdsensing system that supports both mobile application and

chatbot. We further wish to study the differences that might exist while performing a crowdsensing task in a text-based conversational interface and the mobile application under equivalent circumstances - in terms of user engagement and usability.

1.2 Contributions

Realizing the importance of user engagement in mobile crowdsensing applications, we explore the usage of text-based conversational interfaces. This is one of the most important factors that will govern the future enhancement of crowdsensing. We designed and implemented a text-based conversational interface by mapping UI elements and features of a mobile application. Further, we designed the crowdsensing system that supports both mobile applications and chatbots. Using this text-based conversational interface in the crowdsensing system, we conduct experiments to study the usage of this interface as an alternative to the existing mobile application. We summarize the contributions of this research as follows:

C1: We conducted a literature study to understand the evolution of crowdsensing techniques. This study investigates the state-of-the-art techniques of mobile crowdsensing methods and further explores the shift from usage of add-on sensor devices to the smartphone as a sensor. Thereafter, we point out that current literature on crowdsensing through mobile application interface has mostly ignored user engagement. Furthermore, we analyze the usage of text-based conversational agents in crowdsourcing and correlate this analysis to improve user engagement in crowdsensing application.

C2: We designed and implemented a crowdsensing system that supports the usage of both the mobile application and text-based conversational interface. To support the deployment and comparison of both interfaces, we designed a generic crowdsensing system containing task generation, task assignment, and result aggregator. We developed a mobile application according to the state-of-the-art research. We designed and implemented a text-based conversational interface by mapping various features and elements of the traditional mobile application.

C3: To compare the two interfaces for deploying a crowdsensing task and understand the differences in terms of user engagement and usability, we organized two experiments. The location of these experiments is TU Delft campus and the participants for the experiments are TU Delft students. Based on this location, we designed four task domains and three types of tasks. The first experiment is a between-subjects study and has 80 participants. The second one is a within-subjects study conducted in a more controlled environment and has 10 participants. We performed a quantitative analysis in the first experiment to measure user engagement and usability. Through this analysis, we show that the chatbot has better user engagement and usability than the mobile application. Further, to understand the reasons behind these results, we conducted a qualitative survey. Finally, to validate the results of the first experiment, we conducted the another experiment.

1.3 Thesis outline

The rest of the thesis report is structured as follows:

- In Chapter 2, we describe types of crowdsensing and components of crowdsensing system. We identify the current limitations of a mobile crowdsensing application and also explore the usage of text-based conversational agents to overcome them.
- In Chapter 3, we present the proposed crowdsensing system architecture that supports both the mobile application and chatbot. Further, we study how the elements and features that are typical of mobile application interfaces be mapped to text-based conversational actions. Finally, we discuss the workflow design of the interfaces.
- In Chapter 4, we describe different task domains and task types. We further discuss the implementation of the crowdsensing system, mobile application, and text-based conversational interface.
- In Chapter 5, we detail the experimental setup to explore the usage of a text-based conversational interface as an alternative to the mobile application to execute crowdsensing tasks. We also discuss and analyze the results of these experiments.
- In Chapter 6, we conclude the research and explore the possible future work.

Chapter 2

Literature study

To answer RQ 1, we conducted a literature survey to study state-of-art in mobile crowdsensing. In Section 2.1, we study different types of crowdsensing based on the level of user involvement. Further, to understand the components in a crowdsensing system and the evolution of crowdsensing applications, we study different crowdsensing applications that use add-on sensors and interpret their drawbacks in Section 2.2 and we look into the usage of smartphones as a sensor for mobile crowdsensing applications and analyze their limitations in Section 2.3. Thereafter, to overcome the current limitations of mobile crowdsensing that require explicit user involvement, we investigate previous work to understand the usage of text-based conversational agents in crowdsourcing in Section 2.4. Finally, we summarize our findings in Section 2.5.

2.1 Types of crowdsensing

In this section, we study different types of crowdsensing based on the level of involvement of the user to understand different mobile crowdsensing applications.

Participatory sensing
Participatory sensing requires the explicit involvement of individuals to contribute to sensor data. In these applications, complex problems can be solved efficiently by leveraging the intelligence of a large group of users involved. For example, GarbageWatch [4] recruits people to monitor the content of recycling bins with an objective to improve recycling. For this, it requires the user to go to the location of the bin to click the picture.

Opportunistic sensing
Opportunistic sensing requires minimal involvement of the user and is mostly autonomous. Therefore, they lower the burden placed on the user. For example, CenceMe [22] is a smartphone application built to infer peopleś presence (e.g., dancing at a party with friends). For this, it automatically collects data such as current location, audio clips and random pictures using the user's smartphone.

In this research, we are interested in the applications that require the explicit involvement of the user. Therefore, in the following sections, we discuss the

evolution of crowdsensing in the context of these kinds of applications.

2.2 Crowdsensing Applications

Crowdsensing has a broad spectrum of applications including but not limited to infrastructural monitoring, environmental monitoring, and investigation of the spread of disease. In this section, we discuss some participatory crowdsensing applications in the following areas,

Environmental applications
In environmental crowdsensing applications, the natural environment such as the level of pollution at a place or even water levels in creeks is monitored. These applications monitor various elements of the environment by getting data from a large group of individuals. Ikarus [33] is one such environment sensing application that measures thermal atmospheric conditions. To sense this, the paraglider pilots were asked to use a navigation device and record GPS locations, barometric altitude, and timestamps during their flight. After the flight, all the track logs are uploaded by the pilots to a common database of the community website.

Common Sense is a prototype that has been deployed to track pollution [7]. It consists of specialized handheld air quality sensing devices that communicate with mobile phones Bluetooth to collect data related to various air pollutants such as carbon dioxide and nitrogen oxides and then the data is saved to a remote database.

Infrastructural applications
These applications monitor elements related to public infrastructure. This includes surveillance of traffic and road conditions, availability of parking slots, etc., ParkNet detects available parking spots through ultrasonic sensor devices [19]. The location of the detected slots is communicated with the server using GPS receivers. The data is then aggregated at the server which provides a real-time map of parking availability.

Health and wellness applications
These applications collect data to monitor the health and wellness of people. For example, an individual self monitors their diet routine and physical activity.

Heartphone[28] assesses the cardiovascular function through the photoplethysmographic sensors that were integrated into wearable earphones. For further processing, the sensed data is sent to an iPhone via serial communication.

2.2.1 Crowdsensing Components

Analysis of these applications shows that there are mainly three components in a crowdsensing application: Sensor device, Medium, Server. As depicted in Figure 2.1, a sensor device is the one that generates data. The server remotely stores data and also does the processing of the data. Medium transmits the data from the sensor device to the server. Table 2.1 shows components of some crowdsensing applications.

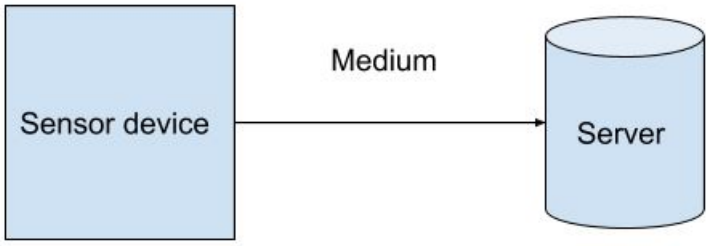


Figure 2.1: Crowdsensing application components

Table 2.1: Crowdsensing applications

Publication	**Sensor device**	**Medium**	**Server**
Ikarus[33]	GPS-based flight navigation device	Manually uploaded to website by pilots	Database of community website
Common Sense[7]	Handheld air quality sensing devices	Bluetooth	Remote database
ParkNet[19]	Ultrasonic sensor device and GPS receiver	Cellular uplink	Centralized server
Heartphone[28]	Photoplethysmo-graphic sensors	Serial communication	iPhone

However, installing extra hardware or another physical sensor may not be an option for everyone [33, 7, 11, 19, 6]. While some users would be reluctant to use extra hardware, some may not even find it. Moreover, for some devices, users might have to learn how to use them [33]. This reduces the accessibility of the application to a large number of audiences. On the other hand, with advancements in smartphone technology, there has been an increase in the usage of smartphones as sensors for crowdsensing. In the following section, we discuss the usage of the smartphone as a sensor for crowdsensing applications.

2.3 Smartphone as a sensor

Nowadays, smartphones are not limited just as a medium for communication, instead it has evolved also as a sensor device with high computing capabilities and connectivity. Their multi-purpose functions made them extremely popular. Moreover, according to Statista, there are more than 2.7 billion smartphone users in 2019. Therefore, the smartphone also becomes an important tool for mobile crowdsensing applications. Most smartphones have embedded sensors like GPS, camera, and microphone. Smartphones can provide continuous sensing since people carry them all the time. Moreover, they can also act as a medium to transmit the data and can also

provide coverage in remote places where static sensors are hard to deploy and maintain [15].

2.3.1 Applications

To understand the implementation of crowdsensing using mobile application interface, we discuss applications in the following areas,

Environmental applications
CreekWatch developed by IBM Almaden Research Center monitors water levels and quality in creeks [14]. It gathers data from the users through pictures or text messages. To collect the data, they have developed an iPhone application. The information generated from these aggregated reports can be used to track water pollution or even the location of trash along the creek.

Infrastructural applications
Bikestatic records the condition of the road i.e., presence of bumps and potholes using the built-in sensors like accelerometer gyroscope through a mobile application. This data is used to provide a visualized road map that can help in improving the experience of a cyclist [4]. Similarly, Microsoft Researchâs Nericell utilizes individualsâ phones to not only determine road conditions but also measure average speed or traffic delays and honking levels [23].

Personal Environment Impact Report(PEIR) is a crowdsensing system that uses GPS collected data from users to classify their activities and tracks their transportation modes [24]. This provides data to assess elements such as their carbon footprint and fast food exposure.

Health and wellness applications
Another application that uses pictures but in a different domain is DietSense [30]. In this, the users take pictures of what they eat and share it to compare their eating habits. This can help in watching and understanding what people with similar health conditions eat which can motivate them to control their diet or even ask suggestions from others.

2.3.2 Limitations

Most of the current research in crowdsensing using smartphones focus on applications and system. However, for a mobile crowdsensing application to succeed, it should be able to recruit, engage and retain the users for a longer duration [8]. But not much work has been done to increase participant engagement which ensures their participation in the long run. So, it is also important to analyze the user engagement and usability of the applications that deploy crowdsensing tasks.

Moreover, because of the availability of an enormous number of mobile applications, there has been an increase in app fatigue [25]. So, there is a need to search for an alternative way to reach the users and retain, engage them for a longer duration.

2.4 Crowdsourced conversational agent

Recently, text-based conversational agents have been extensively used and they are known for engaging users by mimicking the conversation as a real human [2]. A text-based conversational agent is an example of the cognitive computing system that emulates human conversations to provide informational, transactional, and conversational services. In this section, we discuss Human Aided Chatbots, i.e. chatbots that rely on humans in the loop to operate. Human Aided Chatbots exploit human intelligence, brought for instance by crowd workers or full-time employees, to fill the gaps caused by limitations of fully automated solutions.

Guardian is a crowd-powered web-API-based spoken dialog system that leverages the wealth of information in web APIs to enlarge the scope of the information that can be automatically found [10]. The crowd is then employed to bridge the dialogue system with the web APIs and a user with the dialogue system.

Similarly, Chorus is a crowd-powered conversational assistant that allowed the end-users to have a conversation with a seemingly single conversational partner, but in reality a group of crowd workers was involved [16]. To generate the response, they used the voting process and the highest voted one would be sent to the end-user.

Crowd-Intelligence chatBot(CI-Bot) is another hybrid intelligent chatbot [17]. When a task is requested, CI-Bot first tries to solve it automatically. If the task is beyond its knowledge, it would be distributed among selected workers. The responses from the workers would be aggregated and the response generated would be added to the corpus.

Facebook M is a text-based virtual assistant that involves humans in the loop to train an artificial intelligence system. When a user asks a question or suggestion, M would decode the natural language and ask more follow up questions to gain a deeper understanding of what the user is looking for. If the answer is already available in the corpus, it replies, otherwise, it would request a group of workers to help.

To conclude this, the above applications exemplify that chatbots can be used in combination with crowdsourcing. But, there is no research available that explores the usage of chatbots for executing crowdsensing tasks. Crowdsensing can be seen as a type of crowdsourcing as both of them are dependent on the participation of the human crowd to get the job done. But, depending upon the type of task, crowdsensing may involve more active involvement of users. Further, while crowdsourcing can be performed using any digital device, crowdsensing expects the device to be mobile.

2.5 Summary

The concept of using smartphones as a sensor instead of tagged physical sensors has only arisen recently. With many studies being performed on how to incorporate crowdsensing using the mobile application into real-world applications, other important factors such as user engagement and user satisfaction are overlooked. However, for a crowdsensing application to succeed, it is important to pertain users for a longer duration.

On the other hand, chatbots have gained popularity recently for their ability to engage users by mimicking human-like conversation. Though chatbots have been used for crowdsourcing which is a similar technique, crowdsensing requires more active involvement of the users.

With more research being done in developing applications in a crowdsensing framework, we recognize that to our knowledge the idea of using the text-based conversational interface as an alternative to a mobile application for the deploying a crowdsensing task remains to be explored.

Chapter 3

System design

Before moving into the System implementation in Chapter 4, in this chapter, we elaborate and justify all the decisions impacting the design of the text-based conversational interface. In Section 3.1, we discuss the requirements of the system and in Section 3.2, we describe the crowdsensing framework used in this research. Thereafter, in Section 3.3 we elaborate the design of a text-based conversational agent that can be used to deploy crowdsensing tasks. And to answer RQ 2, we study UI elements and features of a traditional mobile application interface that would be mapped to the actions in implemented text-based conversational interface in Section 3.4. Further, in Section 3.5, we discuss the workflow of both the interfaces implemented in Chapter 4.

3.1 System requirements

Before designing the crowdsensing system in the subsequent section, we describe the requirements of the system in this section.

- As we are exploring the usage of a text-based conversational interface as an alternative to mobile application for performing crowdsensing tasks, the system should have both the mobile application and text-based conversational interface
- The system should support both interfaces simultaneously
- The system should have a database that stores all the data related to participants and tasks. Moreover, both interfaces should be connected with the same database as participants of both the interfaces should be provided with equivalent tasks
- It should have a Task generation module that generates the designed tasks
- It should also have a Task assignment module that assigns the generated tasks to appropriate participants
- Finally, it should have a Result aggregator module that aggregates all the responses from the participants and calculates the final result

3.2 Crowdsensing system framework

In this section, we discuss the framework that has been designed based on the requirements described in the previous section. It has four major components: User interfaces, Task Compiler, Database and Crowd modules which are briefly explained in the sub-sections 3.2.1 to 3.2.4

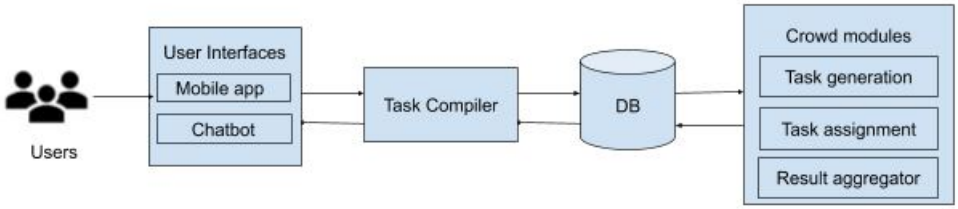


Figure 3.1: Crowdsensing system framework

3.2.1 User Interfaces

As the intent of this research is to check if the chatbot can be used as an alternative to a mobile application for executing crowdsensing tasks, the system has two interfaces: mobile application and chatbot. And, the users are grouped into two categories: mobile application users and chatbot users. Both groups would be presented with the equivalent environment which means that they would be performing the same tasks but with a different interface.

3.2.2 Task Compiler

The Task Compiler gets information from the database and compiles the task. This module ensures that the task is presented in an appropriate format for the corresponding application. For example, while a microtask is presented as a message to a chatbot user, it is displayed in a form for the mobile application user.

3.2.3 Database

Both applications are connected to a common DB which stores all the information related to the tasks and responses. As soon as a task is generated and assigned to the users, the details are stored in the database. After completion of a task, the answers submitted by the user are also saved here.

3.2.4 Crowd modules

Crowd modules contain Task generation, Task assignment, and Result aggregation. These Crowd modules work together to aggregate the responses from the users, generate and assign the tasks, which would be stored in the Database.

Task generation
Task generation module is responsible for scheduling and generating tasks. This is the initial module of that activates the other components i.e., the tasks need to be generated first before task assignment and result aggregation. Task dependency should also be taken into account while generating new tasks. For example, âFind the most popular restaurant in a neighborhoodâ, can be divided into two tasks: Finding restaurants in that neighborhood, and Voting for the most popular one. But, before generating the second task, we need to have at least a specified number of responses from the first one. To schedule a task, expiration should also be taken into account. For instance, a task like finding the cleanliness of a place may need to be scheduled very often. We describe the Task generation strategy used in the experiments of this research in the next chapter.

Task assignment
The number of responses required can be mentioned within the task itself, which can be used to assign the generated task accordingly to the specific number of users. This module is responsible for selecting suitable users and assigning them a particular task. The strategy can be as simple as assigning it to all users or randomly selecting some. While this can work for some tasks, it can also jeopardize the quality of results in some cases as inappropriate workers may either not complete tasks or introduce false answers. For example, crowdsensing tasks may require the users to be at the location, so it is better to select the users around the task location. We discuss the task assignment strategy used in this research in Chapter 4.

Result aggregator
As same task is assigned to multiple users, all their responses must be aggregated into a single value to get the final result. If a task contains n microtasks each assigned to k users, the input to the Result aggregator would be,

$$Response = \begin{bmatrix} r_{11} & r_{12} & \dots & r_{1k} \\ r_{21} & r_{22} & \dots & r_{2k} \\ . & & & \\ . & & & \\ r_{n1} & r_{n2} & \dots & r_{nk} \end{bmatrix} \tag{3.1}$$

where r_{ij} is the response of the user j for the microtask i. After applying aggregation technique, the output is a set R= R_1,R_2,...,R_n where R_a represents the final result of the microtask a. It is important to choose an appropriate strategy as the final result depends not only on the user's response but also on how it is integrated. The aggregation strategy can be customized based on the need. We describe the Result aggregation strategy used in this research in Chapter 4.

3.3 Design of Text-based conversational interface

Crowdsensing system framework discussed in the previous section has two interfaces. This research explores the usage of a chatbot which is a text-based conversational interface for executing crowdsensing tasks.

To get a deeper understanding of chatbot design, in this section, we detail the main elements and components in the chatbot system that is implemented in Chapter 4. As depicted in Figure 3.2, after the system receives an input from the user, it is sequentially processed by four core components namely: Action interpreter, Action executor, Database and Response generator. These components work coherently in generating the response which enables the chatbot system to interact with the user. Other chatbot elements like User input, State, Context, Response and functionality of all the components are succinctly discussed below.

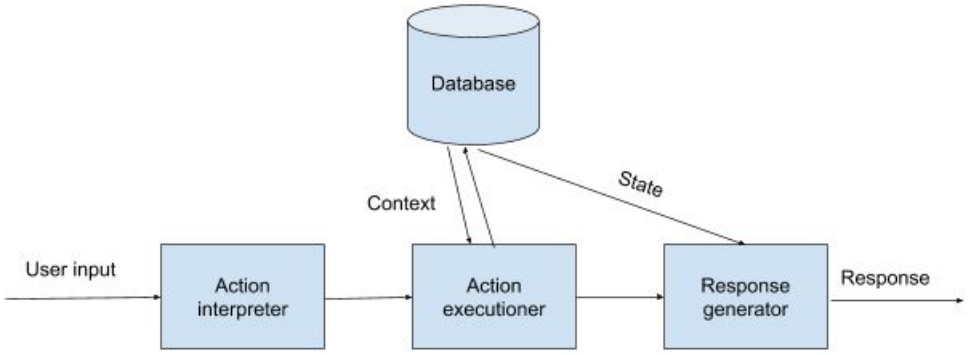


Figure 3.2: Core elements and components of chatbot

User input

The primary way of communication between the user and the chatbot is through messages. User input which can be a query or an answer is a message from the user to a chatbot. While some messages may simply consist of plain text, others may contain richer content such as attachments and buttons. Free text, Attachments, Commands, and Choice are different message types which are briefly explained below.

- *Free text:* Text is the most flexible and unrestricted type of message for communication with a chatbot. To give an outlook, if the chatbot asks, "How are you?", the user may answer with the word âFineâ, or with a sentence, "I am fine"
- *Attachment:* Attachments represent images, videos, audios, and files exchanged between the user and chatbot. The user can upload a photo to support a query or a document to store for future use
- *Choice:* To simplify the interaction for users special keyboards can be used predominantly for close-ended questions or a formal input. When chatbot intends to send a message, it can pass along a keyboard with predefined reply options. And, users can convey their choice by clicking on any of these buttons
- *Command:* Specific commands help to simplify the chatbot and better understand the user intent. A command is a string with predefined syntax to perform a particular action rather invariably following the pre-established process. A chatbot can be designed to use /start to create a start a process and /quit to stop it. These commands provide better user experience as they are easy to interpret and the outcome is expected

Action interpreter
During a conversation, the user must receive an appropriate response from a chatbot as the failure of this may lead to dissatisfaction of the user. To achieve this, the chatbot should be able to map the user's input with apt intent. Intents are purposes or goals that are expressed in a userâs input, for example, in a food ordering bot, the intent of a user can be "finding popular restaurants near me" or "rating the restaurant".

Once user input is received, it is passed on to the Action interpreter which comprehends the user input and finds the intent. The further flow of conversation would be based on this interpretation.

Database
This is the same Database mentioned in the crowdsensing system framework. It stores, organizes and retrieves data related to the users, tasks and user's answers. If there is new data, the database is updated based on the existing data. As illustrated in Figure 3.3, if the task is to find popular restaurants, initially the database would contain only the query, but as the users start answering and voting for popular ones, it is updated with the names of restaurants and upvotes/downvotes accordingly.

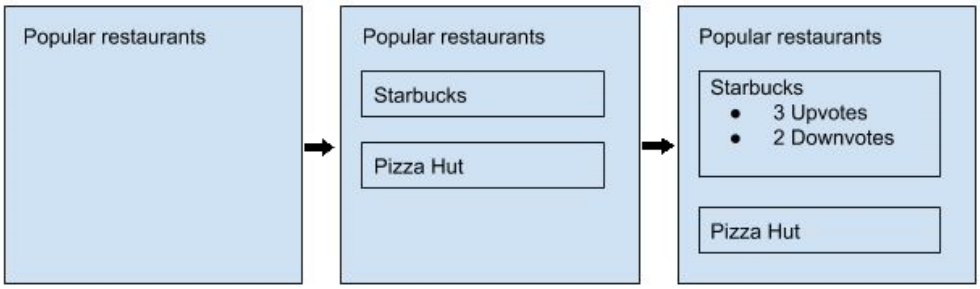


Figure 3.3: An example of updating database - finding popular restaurants

Context
For good communication, the participants need to know what they are talking about. Similarly, while the chatbot is interacting with the user, it should know the context. Knowing the current context helps in better perceiving the user intent. For instance, in a bot that collects reviews about restaurants and movies, if the user input is 'I give this a 4-star rating', the chatbot should be able to understand the context and identify if the user is talking about restaurant or movie.

State
State refers to the previous messages in a conversation with the user. The information about the current state is required to better understand the context and generate an appropriate response. To give an example, in a bot that collects reviews about restaurants and movies, if the user input is 'I like this', the context can be identified using the previous messages and this information can be used to generate chatbotâs response.

Action executioner
Action executioner queries the database considering the current context to get the information required to proceed further with the conversation. For example, if the user input is 'What are some popular restaurants near me?', after the Action interpreter comprehends the intent of 'finding popular restaurants', this module queries the knowledgebase and returns a list of popular restaurants.

Response generator
The Response generator module assimilates the query result as input and the information about the user's previous message from the database to generate a response. For the user input 'What are some popular restaurants near me?' ensuing the list of popular restaurants returned by the Action executioner, this module checks the previous messages from the user to find the location and filters the restaurants by that location to generate the response.

Response
Response is the message from chatbot to the user which is the final output generated after the user's input has been processed through all the components. It can be a predefined text, a question or an answer retrieved from the database, or an attachment.

3.4 Elements and Features of mobile application interface

To compare the user engagement between the mobile application and the conversational interface, it is important to make sure that they have the equivalent experimental environments and settings. So, in this section, we discuss different elements and features of the mobile application that are mapped into the text-based conversational agent implemented in Chapter 4.

3.4.1 Mobile application UI elements

In this section, we describe the basic UI elements that are needed to build a mobile application. These elements are into the text-based conversational interface implemented in Chapter 4.

- *Cards:* Cards serve as an entry point to more detailed information. It shows a preview or gives hints on how further navigation may look like. A card often contains an image, title, description/content and a button.
- *Free text:* Free text is a representation of characters that controls simple arrangement of text which includes alphabets, numbers and special characters. It does not have any graphical representations or executable expressions.
- *Buttons:* Button is a clickable element that is associated with some action. They allow users to make their choices just by a single tap.
- *Single/Multiple selection:* Selects are form controls where users can select one or multiple options from a given set of options. The user would be presented

with a list of all options from which they can select. In a single selection, the user can just select one option from the provided list. But, in a multiple selection, the user can choose multiple options at the same time.

3.4.2 Features

In this section, we describe some of the features in the mobile application that are not readily available in a text-based conversational interface. Further, in chapter 4, we explain the methods used to provide these features in the chatbot.

- *Handling interruption:* Users are not always ready to follow the defined flow. They may start a task and in the middle of the process, they might want to stop it instead of completing it. The mobile application users can do this by simply quitting from the current page. But, for chatbot users, it is difficult as they are already in the middle of a conversation.
- *Reviewing and editing answers:* Users may want to review their answers after they fill it in. While mobile application users can just scroll up, review and edit their previous answers, chatbot users can inspect their answers skimming through their conversation but cannot edit it.
- *Skipping a question:* Users may not want to answer all the questions in a task but still want to continue with it. The mobile application users can do this by simply ignoring the current question and moving onto the next one. So, chatbot users should also be able to skip a question if they would like to.

3.5 Workflow of Crowdsensing System

Before implementing the interaction in both interfaces, in this section, we describe the workflow design in Figure 3.4. To further discuss the flow, it is also important to understand the types of tasks considered. Based on the objective of the task, tasks are grouped into the following three types explained in Table 3.1.

Table 3.1: Types of tasks

Task type	**Objective**
Creation	Create a new item with predefined attributes
Enrichment	Add more content based on given content
Validation	Assess the given content

Figure 3.4 illustrates the workflow design of the interfaces implemented in Chapter 4. After the user starts the app(either the mobile application or chatbot), the app sends them a menu that contains a list of task domains. The user can pick the domain in which they want to perform a task. The app then sends them a list of tasks: Create, Enrich, Validate and also an option to refresh the task list. If the user selects refresh, the app sends them a new list of tasks. Else, if the user selects a task type, the app

sends them the appropriate task. After the user completes the task, the app notifies them that the task has been successfully completed.

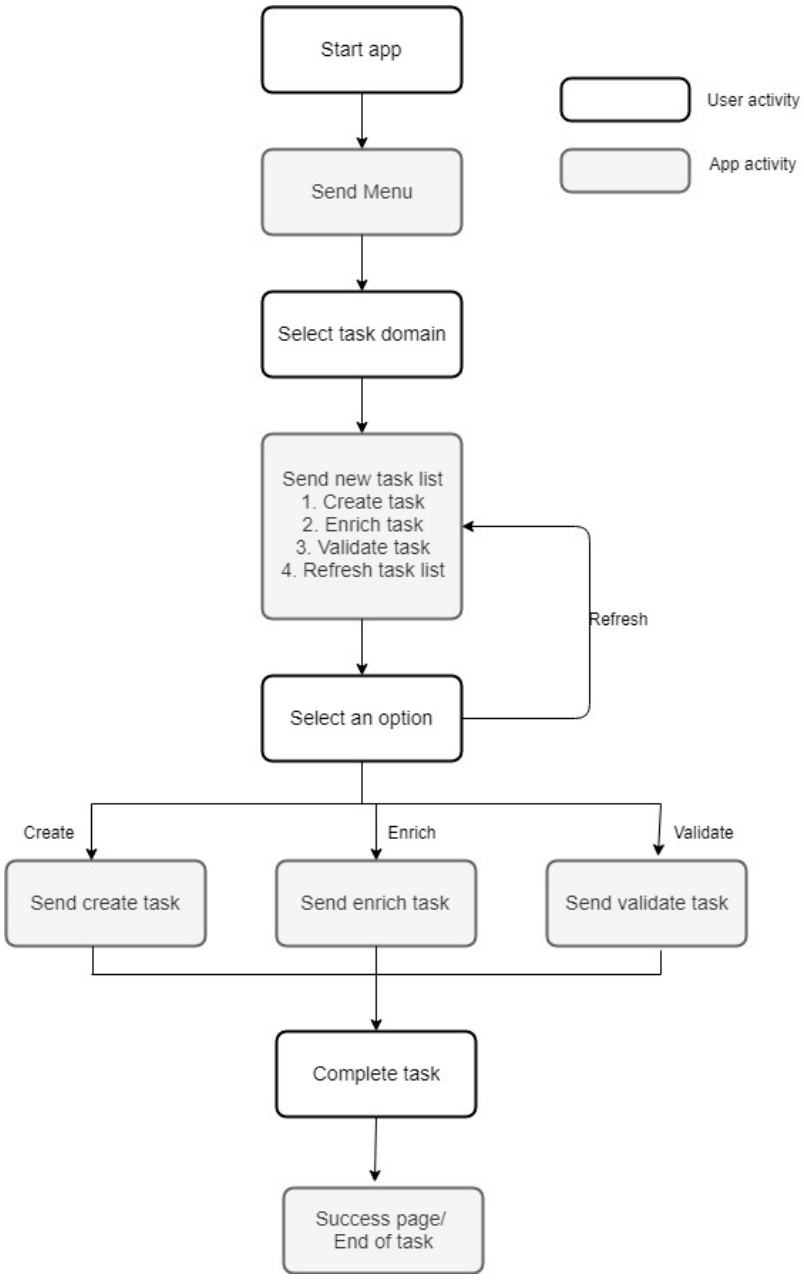


Figure 3.4: Workflow of interface

Chapter 4

System implementation

In this chapter, we elaborate on the implementation of the crowdsensing system designed in Chapter 3. We begin by describing different task domains and task types in each domain in Section 4.1. We further discuss the implementation and deployment of the database in Section 4.2, the mobile application in Section 4.3, a text-based conversational interface in Section 4.4 and the crowd system in Section 4.5. To answer RQ2, we implement the text-based conversational interface by mapping the UI elements and features of the mobile application defined in Section 3.4.

4.1 Task description

Before discussing the implementation of interaction flow in both interfaces, we describe the task domains and task types in this section. We designed the tasks considering the participants and the location of the experiments which are discussed in detail in Chapter 5. We conducted the experiments in the TU Delft campus and the participants are TU Delft students. Taking this into account, We chose the following task domains,

- *Food:* Rate food on campus
- *Place:* Add details about the different types of places on campus such as a Building, Study space, Education room or Parking space
- *Course:* Ask and answer questions related to courses taught at TU Delft
- *Trash Bin:* Find locations of the trash bins on campus and report the trash level

Types of sensors
All the three types of tasks: Create, Enrich and Validate of these four domains include one or more sensing tasks. We used the following sensors for these tasks.

- *Camera:* Camera is used to upload pictures and sense the appearance
- *GPS:* GPS is used to share location coordinates and sense the location

- *User as a sensor:* For subjective attributes, the user is used as a sensor. For example, to know the opinion on the price of food or cleanliness level of a place, the user has been used as a sensor.

We discuss different task types from each domain in detail in the following subsections.

4.1.1 Food

Creation

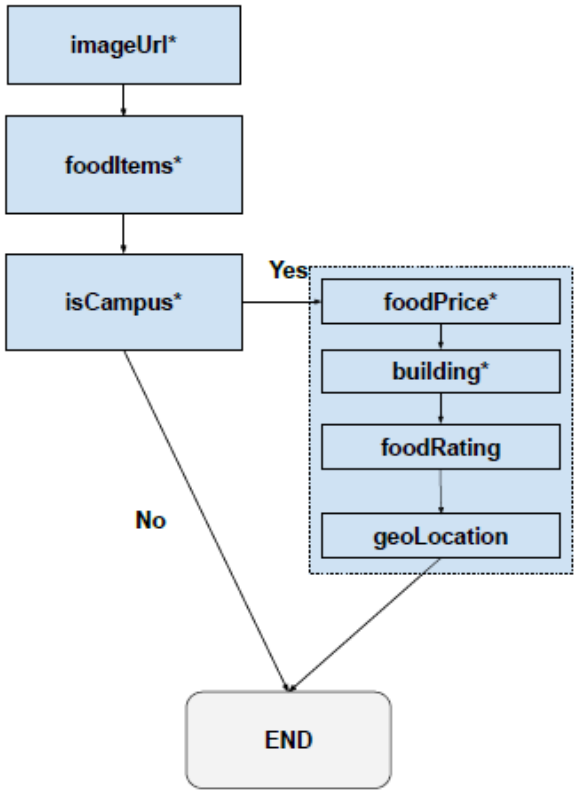


Figure 4.1: Food item creation flow

Figure 4.1 illustrates the flow of item creation in Food. To create an item in food domain, user needs to upload the picture of food (imageUrl*), lists the items in it (foodItems*) and mention if the food is bought from the campus (isCampus*). If it is bought from the campus, user needs to provide more details such as the price of the food (foodPrice*), name of the building from which it is bought (building*), rating for food (foodRating) and GPS coordinates (geoLocation).

Enrichment

Figure 4.2 illustrates the flow of item enrichment in Food. In the food enrichment task, the user needs to categorize each food item in the given picture as appetizer/snack,

main course, beverage or dessert (itemCategory). If the price of food is available, then the user is requested for an opinion on the price.

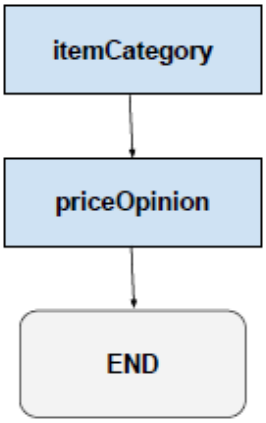


Figure 4.2: Food item enrichment flow

Validation

Figure 4.3 illustrates the flow of item validation in Food. In the food validation task, the user will be asked to confirm if it is a picture of food (isFood*). If the user responds with yes, then they would be asked to check if the given items are present in the picture (foodItemsCheck) and if the category the item is appropriate (itemCategoryCheck). Table 4.1 tabulates all the attributes involved in the food domain and their descriptions.

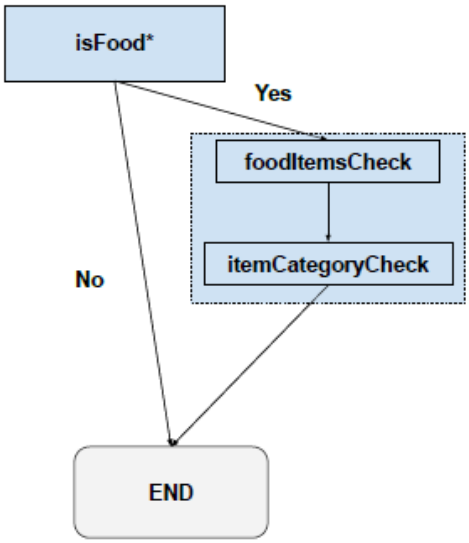


Figure 4.3: Food item validation flow

Table 4.1: Food domain attributes

Attribute name	Description
imageUrl*	Upload a picture of food
foodItems*	List food items in the picture
isCampus*	Mention if food is bought from campus
foodPrice*	Price of food
building*	Name of the building from which food is bought • Aula • Library • EWI • Not in a building • Other If user picks 'Other', they can input the building name
foodRating	Rate food on a scale of 1 to 5
geoLocation	Coordinates of location from where food is bought
itemCategory	Select category of each item in the picture • Appetizer/Snack • Main course • Beverage • Dessert/Fruit
priceOpinion	Mention your opinion on the price of the food • Cheap • Appropriate • Expensive
isFood*	Check if picture contains food
foodItemsCheck	Check if the given list of food items are in the picture
itemCategoryCheck	Check if the given category of food item is correct

4.1.2 Place

Creation

Figure 4.4 illustrates the flow of item creation in Place. In the place item creation, the user needs to upload a picture of the place from the TU delft campus (photo*), add the name of the place (name), select category of place (category) and share geo-coordinates of the place (geoLocation). If the place is of type building then the user is asked to add a number of the building (buildingNumber) else they would be asked for building name in which the place is located (building). If the place is inside a building, then the floor number of the place is asked (floorNumber). Further, other information related to the place such as route to the place (route), presence of electricity outlets (electricityOutlet) and the capacity of the place (seatCapacity) are asked. And, if the user is at location (isUserAtPlace), then dynamic attributes of the place such as currently available number of seats (availableSeats) and the cleanliness level (cleanLevel) are requested.

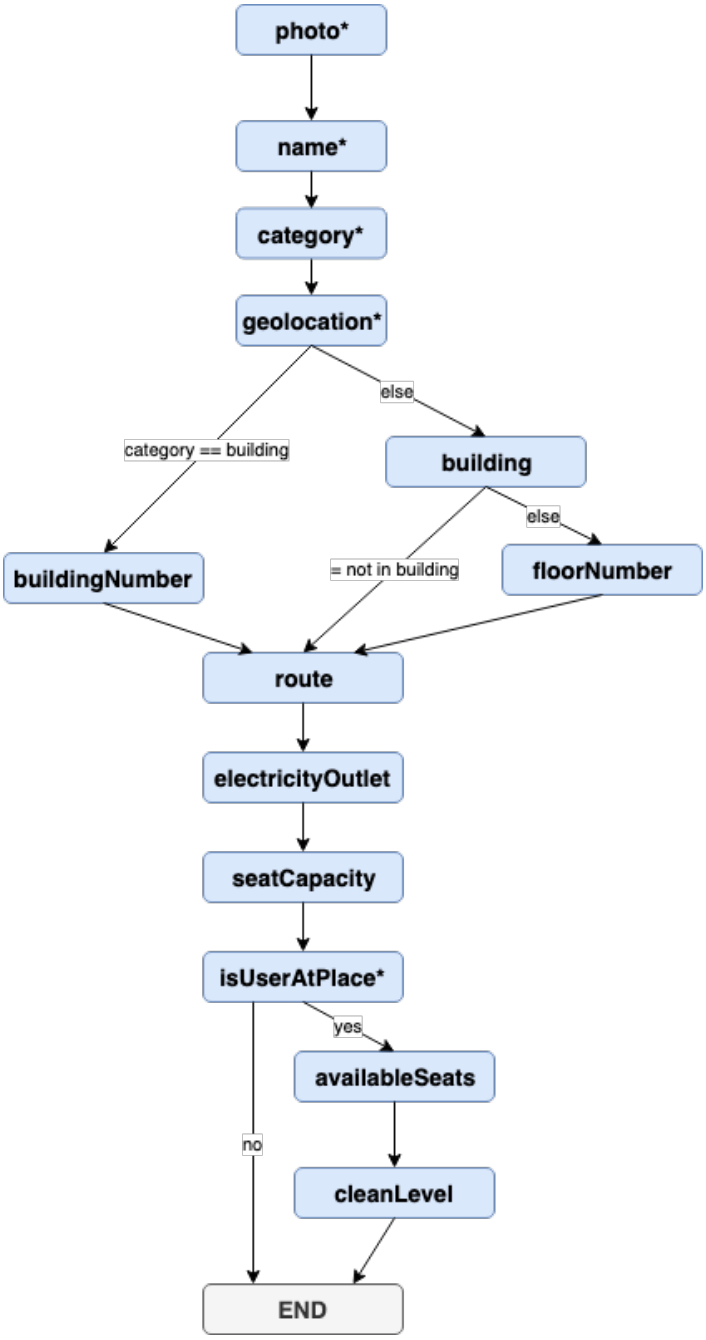


Figure 4.4: Place item creation flow

Enrichment

Figure 4.5 illustrates the flow of item enrichment in Place. For place item enrichment, if the type of the given place is building, then the user is asked to add number of the building (buildingNumber) else they would be asked for building name in which the place is located (building). If the place is inside a building, then floor number of place is asked (floorNumber). Further, other information related to the place such as route

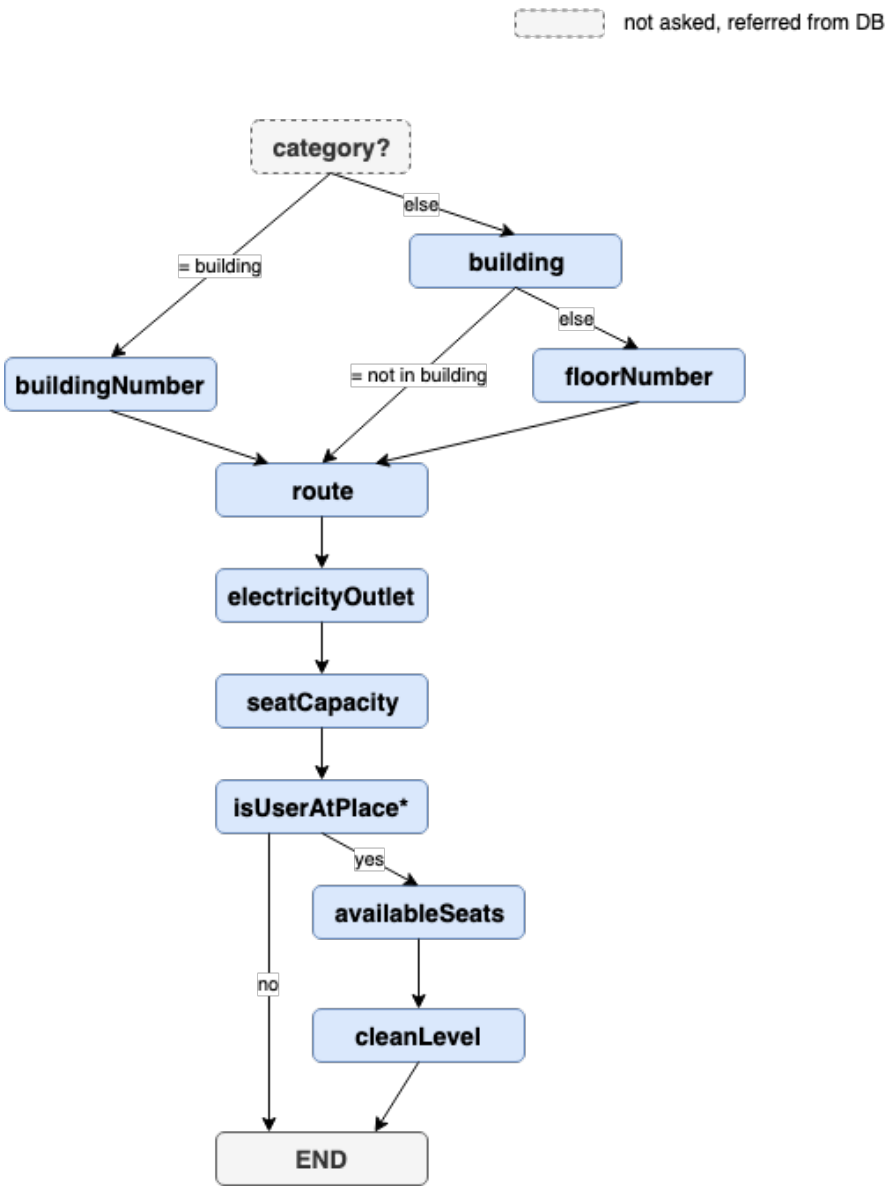


Figure 4.5: Place item enrichment flow

to the place (route), presence of electricity outlets (electricityOutlet) and the capacity of the place (seatCapacity) are asked. And, if the user is at location (isUserAtPlace), then dynamic attributes of the place such as currently available number of seats (availableSeats) and the cleanliness level (cleanLevel) are requested.

Validation

Figure 4.6 illustrates the flow of item validation in Place. For validation task in place domain, the user is asked to check if the photo contains a place from campus (isPlace) and if they respond with yes, they are asked to check if the given category is correct (isCategoryValid). If the type of the given place is building, then the user is asked to

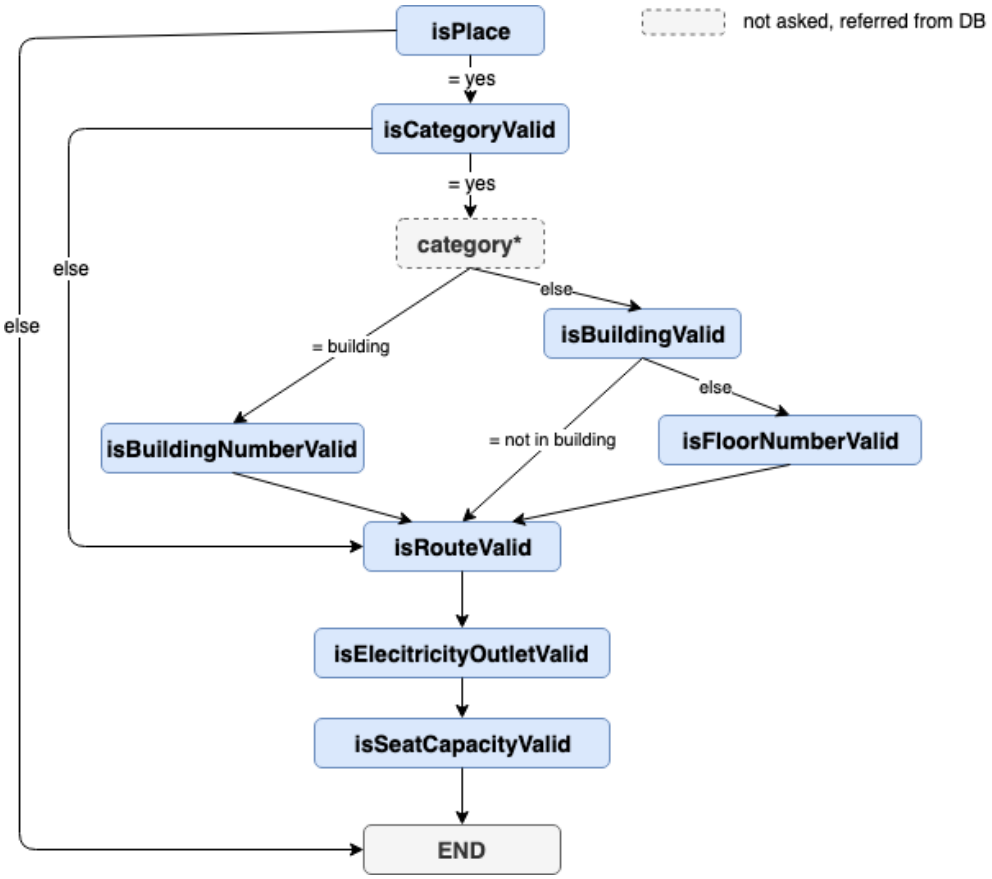


Figure 4.6: Place item validation flow

check if the given building number (isBuildingNumberValid) is correct else they would be asked to validate the given building name (isBuildingValid). If the place is inside a building, then floor number of place also needs to be validated (isFloorNumberValid). Further, the user is asked to check other given information related to the place such as route to the place (isRoute), presence of electricity outlets (isElectricityOutletValid) and the capacity of the place (isSeatCapacityValid). Table 4.2 summarizes all the attributes in the place domain and their descriptions.

Table 4.2: Place domain attributes

Property Name	Description
photo*	Upload a photo of any place from campus
name*	Name of the place
geolocation*	Coordinates of the place

category*	Category of the place • Building • Study Space • Project Room • Lecture Room • Parking Space • Food & Beverage
buildingNumber	The number of the building
building	Name of the building in which the place is located in • Aula • Library • EWI • Not in a building • Other If user picks *other*, they can input the building name
floorNumber	The floor number of the building in which the place is located
route	Short description on how to reach to the place
hasElectricityOutlet	Mention if the place has availability of electricity outlets
seatCapacity	Capacity of place/ number of people the place can accommodate
isUserAtPlace*	Mention if the user is at the place
availableSeats	Number of currently available seats at the place
cleanLevel (1:very dirty - 5:very clean)	Rate the level of cleanliness
isPlace	Check if the photo contains a place from campus
isCategoryValid	Check if the given category of place is correct
isBuildingNumberValid	Check if the given building number of place is correct
isBuildingValid	Check if the given building name of place is correct
isFloorNumberValid	Check if the given floor number of place is correct
isRouteValid	Check if the given route to the place is correct
hasElectricityOutlet	Check if the given place has electricity outlet
isSeatCapacityValid	Check if the given seat capacity of place is correct

4.1.3 Course

Creation

Figure 4.7 illustrates the flow of item creation in Place. For creation task in course domain, user can add a question related to any course (question) and also upload a picture to support the description (imageUrl). Further, the user is asked for the name of the course that the question is related to (courseName), if the course does not exist in the database, the user is also asked for code of the course (courseCode).

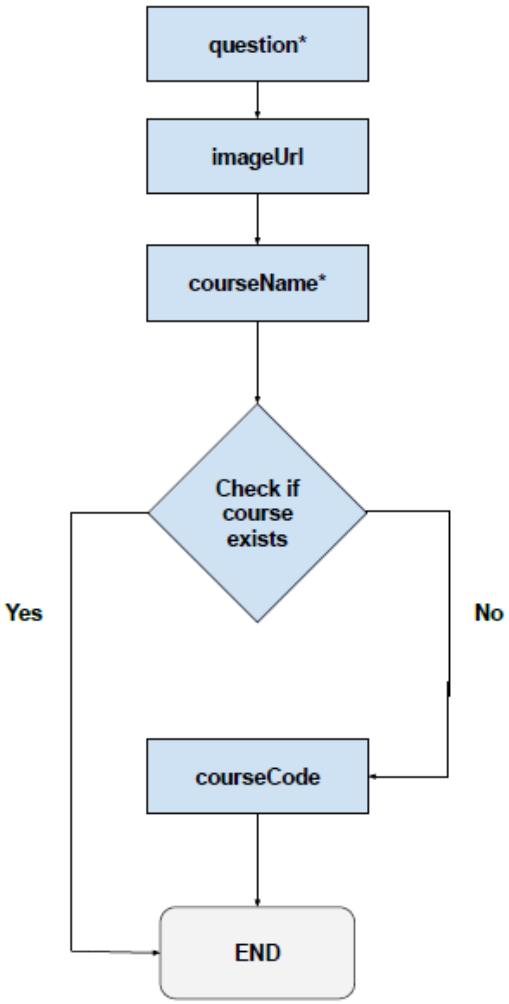


Figure 4.7: Course item creation flow

Enrichment

Figure 4.8 illustrates the flow of item enrichment in Course. For enrichment task in course domain, user can add an answer (answer) for the given question.

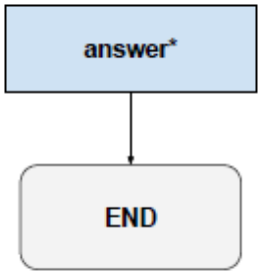


Figure 4.8: Course item enrichment flow

Validation

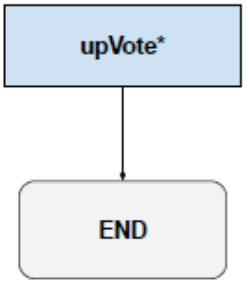


Figure 4.9: Course item validation flow

Figure 4.9 illustrates the flow of item validation in Course. Through validation task in course domain, user can approve or disapprove (upVote) the answers provided for the given question. Table 4.3 summarizes the attributes and their corresponding descriptions for course domain.

Table 4.3: Course domain attributes

Property Name	Description
question*	Question related to any course from TU Delft
imageUrl	Picture to support/ describe the question in detail
courseName*	Name of the course the question is related to
courseCode	Code of the course the question is related to
answer*	Answer to the given question
upVote*	Approve/Disapprove the provided answer to given question

4.1.4 Trash bin

Creation

Figure 4.10 illustrates the flow of item creation in Trash. For task creation in trash domain, the user can add a photo of the trash bin (photo), categorize the type of waste in the bin as general/paper/other (wasteType), share GPS coordinates of the bin (geoLocation) and select the building in which the trash bin is located (building). If the bin is located inside a building then floor number is also requested (floorNumber). Further, other information related to the trash bin such as route to the bin (locationDescription) and size of bin (size) are asked. And, if the user is at location (isUserAtPlace), then dynamic attributes of the bin such as current trash level in the bin (isFull) is requested.

Enrichment

Figure 4.11 illustrates the flow of item enrichment in Trash. For enrichment task in trash domain, user needs to categorize the type of waste in the given bin as

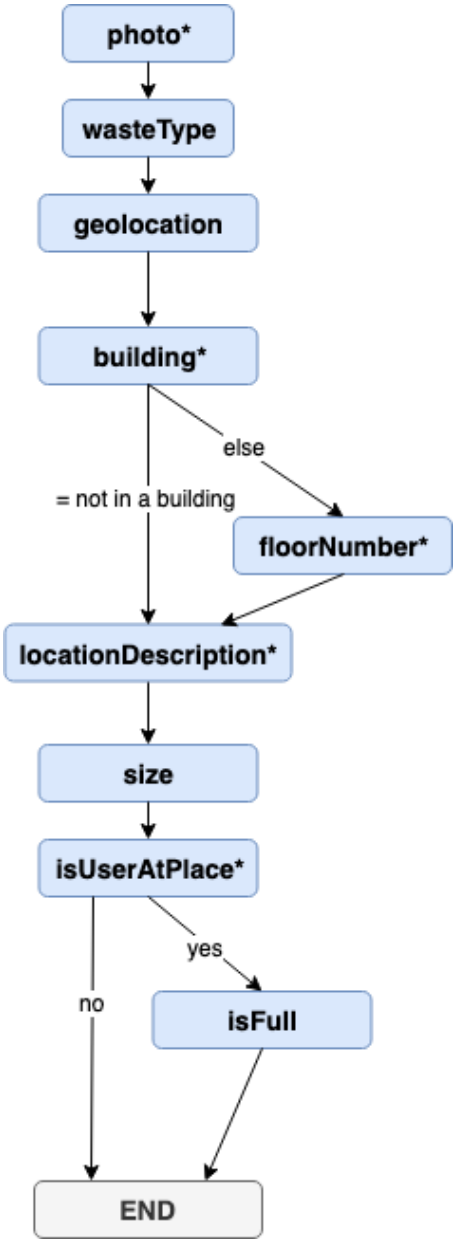


Figure 4.10: Trash Bin item creation flow

general/paper/other (wasteType) and add information such as size (size) and color (color) of the bin. If the user is at location (isUserAtPlace), then dynamic attributes of the bin such as current trash level in the bin (isFull) is requested.

Validation

Figure 4.12 illustrates the flow of item validation in Trash. In validation of trash domain, the user is asked to check if the photo contains a trash bin from campus (isTrashBin) and if they respond with yes, they are asked to check if the given waste

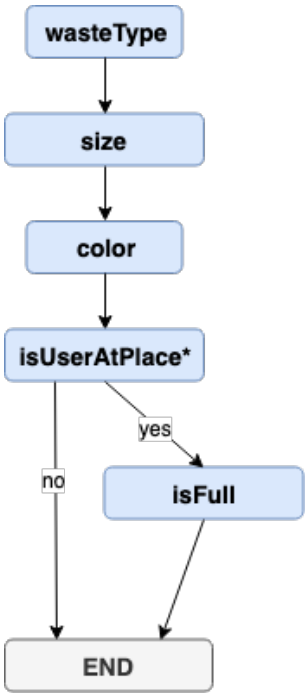


Figure 4.11: Trash Bin item enrichment flow

type (isWasteTypeValid) and color (isColorValid) are valid. Table 4.4 describes the attributes in Trash Bin domain.

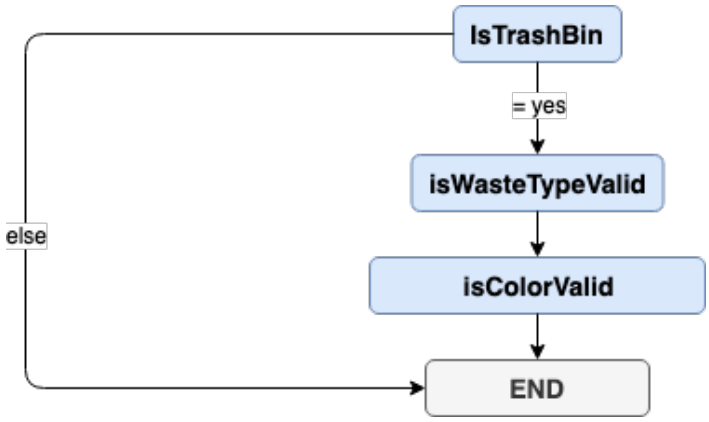


Figure 4.12: Trash Bin item validation flow

Table 4.4: Trash bin domain attributes

Property Name	Description
imageUrl*	Upload photo of a trash bin from the campus

wasteType	Type of waste in the trash bin • General Waste • Paper Cups • Others
geolocation	Coordinates of the location of the trash bin
building*	Name of the building from which food is bought • Aula • Library • EWI • Not in a building • Other If user picks 'Other', they can input the building name
floorNumber*	Floor number of the building in which the trash bin is located
locationDescription*	Short description on how to reach this trash bin
size	The size of the trash bin • Small • Medium • Big • Not sure
isUserAtPlace*	Mention if the user is at the place
isFull	Mention if the trash bin is full
color	The color of the trash bin
isTrashBin	Check if the photo contains a trash bin
isWasteTypeValid	Check if the given waste type of the trash bin is correct
isColorValid	Check if the given color of the trash bin is correct

4.2 Database

We used the database to store two types of data

- *Experimental data:* User model, Tasks generated by Crowd system and Responses from users
- Dialogues/Responses for text-based conversational interface

To store and sync the data, we used a NoSQL and document-oriented database *Cloud Firestore*[1]. This stores the data as documents which are grouped together as a collection. To read data from the database, there are two options: get a collection of items, get a specific document from the database. And, while writing the data to the database, there are also two options to push the data: specify the ID of the document, auto-generate the ID.

[1] https://firebase.google.com/docs/firestore

4.2.1 User model

Table 4.5 displays different attributes of the user model and their description. Once user registers and launches the app, if it is a mobile application, their email address will be added to the data model and on the other hand for chatbot, ID of the platform in which it has been developed would be saved. Initially, preferedLocationNames and preferredCourses are empty, but as the user completes tasks, the corresponding location names for *Place, Trash bin* task domains and course name of the Course domain would also be added to the list. totalTasksCompleted contains count of number of tasks done by the user in each domain and tasksCompleted has the list of timestamps at which the user has completed a task.

Table 4.5: User model

Attribute	Description
email/id	Email address/ Telegram ID
preferredLocationNames	Preferred location names
preferredCourses	Preferred courses
totalTasksCompleted	Count of number of tasks
tasksCompleted	Timestamp of task completion

4.3 Mobile application

To build cross-platform mobile application for native iOS and android, we used ionic framework [2] version 4.0.0 and capacitor [3] version 1.0.0-beta.19. We discuss below the implementation of UI elements defined in Section 3.3.1 and also the usage of sensors in ionic using different APIs.

Sensors
As mentioned in the Task description section, the tasks have been designed to use two in-built sensors from the smartphones namely: Camera, Geolocation.

- *getCurrentPosition* method of the Geolocation API gets the current location of userâs smartphone
- *getPhoto* method of the Camera API prompts the user to pick a photo from an album, or take a new photo with their smartphoneâs camera

Cards
To build the cards, we used *ion-card* which is an ionic component. Each card can contain a title, header and content. The content of cards can contain images, buttons or text. To implement the card in Figure 4.13, *ion-thumbnail* can be used to wrap the image in the list which can be made clickable using the *tappable* property of ion-list.

[2] https://ionicframework.com/
[3] https://capacitor.ionicframework.com/docs/

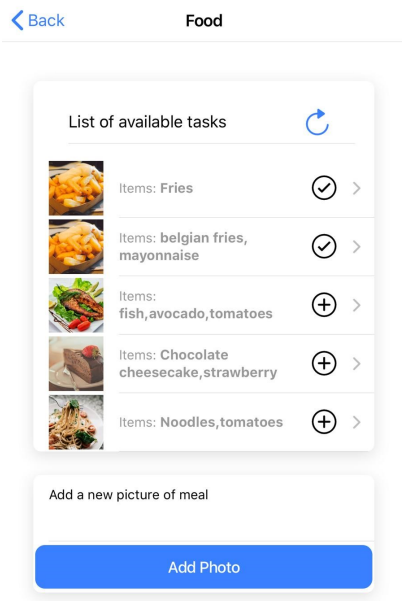


Figure 4.13: Ionic cards

Buttons
To insert a button in the page, we used *ion-button*. As illustrated in Figure 4.13, we can have two types of buttons: Buttons with text (Add Photo) and Buttons with just icon (⟳). i*on-icon* gives the choice to insert an icon from the provided list of ionicons[4].

Free-text
ion-input is the input component in ionic that is meant for text type inputs only, such as text, password, email and number. It is also possible to set properties such as read only, required, minimum/maximum length and default value.

Single/Multiple selection
To implement this, we used *ion-select* which is an ionic component. Each ion-select is associated with multiple *ion-select-option* where each of it declares a value from the list of options. *AlertController* API has been used to display these options in an alert which has two buttons: Cancel and OK. By default, it ion-select provides single selection option, but it can be changed to multiple selection by setting the property multiple to true. If it a single select then the select component receives the value of the selected option else if it multi-select then it receives an array of all of the selected option values.

Notification
We used *Cordova Local-Notification* plugin to display the local notifications on the smartphone. *schedule* method sets properties of the notification such as the title, text and the time to display.

[4] https://ionicframework.com/docs/v3/ionicons/

Reading and writing to database
The app interacts with the firestore using the firebase[5] package. A reference pointing to the document/ collection from the database, is used to read it. *add* method adds new data to the created document and *update* method updates the data in the document.

Android and iOS apps
After we built the application using the ionic framework, we added the android and iOS platforms to the project. We generated the apk file using the *Android studio 3.4.1* and distributed this app through an app-hosting website appho.st and we shared this link (https://appho.st/d/#/07x87HeR) with the participants.

For iOS application, we requested UDID of their device in the google form. We added this UDID to the provisioning profile before building the .ipa file using *XCode 10.1*. We used App box[6] to deploy the app and shared the link https://tiny.app.link/u5C70GbsmW with the participants.

4.3.1 Interaction flow

After user installs the mobile application and launches it, they would be presented with the page in Figure 4.14. To login to the app, user will need a registered account and password

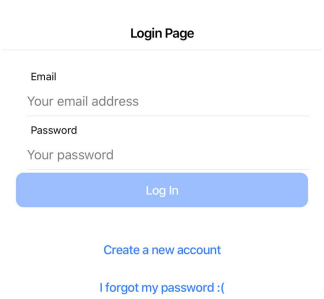


Figure 4.14: Login Page

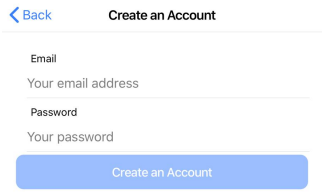


Figure 4.15: Create account page

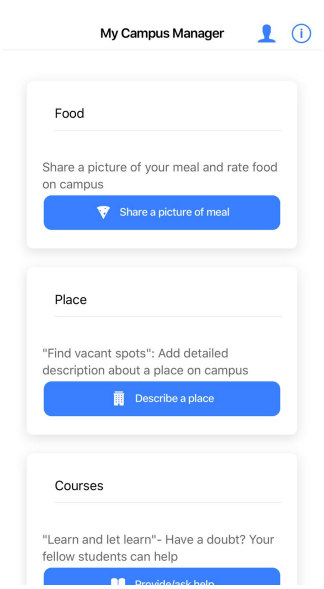


Figure 4.16: Start page

Create an account
To create an account, user should click on *Create a new account* button on login page illustrated in Figure 4.14. It redirects them to the page in Figure 4.15 where the user should register by entering their email address and password. After this, a verification

[5]https://www.npmjs.com/package/firebase
[6]https://getappbox.com/

mail would be sent to registered email address. Once user verifies, they can enter their registered email address and password and click on *LOGIN* button to get started. This redirects them to *Start flow*. If the user wants to reset the password, they can click on *I FORGOT MY PASSWORD* button on the Login page (Figure 4.14). An email with a reset link would be sent to their registered email address.

Start flow

If the login is successful, it will redirect to Start page in Figure 4.16 where the user would be provided with four cards each describing a task domain. Clicking on the *i* button provides a detailed description of the four task domains and task types. User can pick a domain to perform tasks by clicking on the button in the corresponding card. This would forward it to the *Task selection* flow.

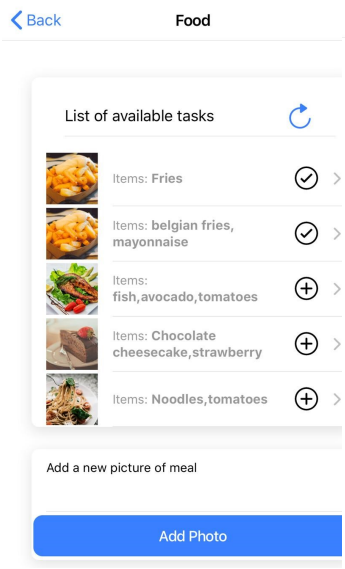


Figure 4.17: Task list page

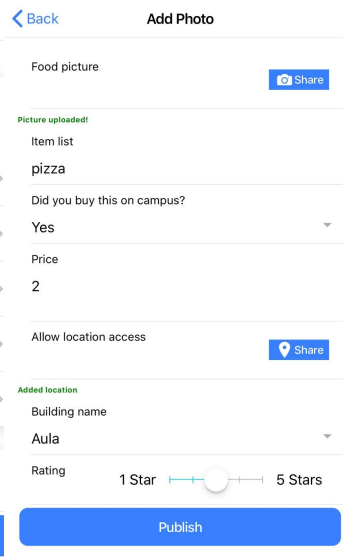


Figure 4.18: Item creation

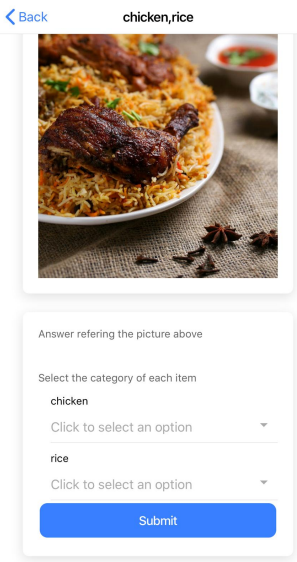


Figure 4.19: Item enrichment

Task selection

After the user clicks on the button corresponding to task domain, they would be redirected to Task list page illustrated in Figure 4.17. Here, they would be provided with a list of 5 tasks each in a different card depicted in Figure 4.17. Each of these cards are clickable and contain few hints about the task and also type of task (+: Enrich, ✓: Validate). The page also has a refresh button (↻) and another card for Create task. If the user clicks ↻ on Task list page (Fig 4.17), they would be presented with a new list of 5 tasks. Clicking on the *Add Photo* button in task create card would forward it to *Create task* flow. User can select a task by clicking on the card. If it is a validate task, then it would be redirected to *Validate task* flow, else if it is a enrich task, it would be redirected to *Enrich task*.

Create task

While performing Task Creation, as illustrated in Figure 4.18, the user would be asked to provide information about different attributes corresponding to the task domain designed in the section task description. After the user successfully answers all of them, they would be redirected to *End of task page* (Fig 4.21).

Enrich task
In Task Enrichment (Figure 4.19), the user would be given details that are already available such as picture of item, and asked to provide more information based on this. After the user successfully answers all of them, they would be redirected to *End of task page* (Fig 4.21).

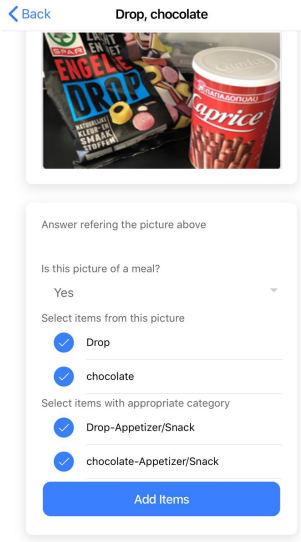


Figure 4.20: Item validation

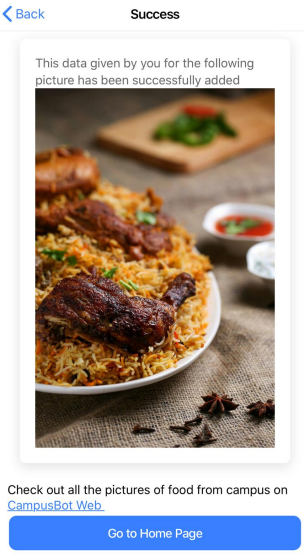


Figure 4.21: End of task page

Validate task flow
In Task Validation (Figure 4.20), the user would be asked to assess the given information. After the user successfully answers all of them, they would be redirected to *End of task page* (Figure 4.21).

End of task flow
After completing the task, the user would be redirected to a success page which provides a glimpse of task that they just completed. This page also contains a link to a web page (CampusBot Web) where they could see answers from their fellow users.

4.4 Text-based conversational interface

We implemented the text-based conversational interface using Python 2.7 and the python-telegram-bot framework which provides a pure python interface to the

Telegram Bot API. In this section, we describe the mapping of different elements and features of a mobile application into text-based conversational interface. Further, we also discuss the conversational flow in this interface.

Message with task preview
Currently, Telegram does not directly support to send a message that maps the card with image content from mobile application. To implement this as in Figure 4.22, Rich link Preview feature of Telegram has been used.

Figure 4.22: Cards with images

Commands
A command in telegram must always start with the '/' symbol and should not be longer than 32 characters. Commands have been used instead of buttons in mobile application. For example, to display the menu in chatbot we used commands with the domain names : /food, /place, /courses and /trashbin.

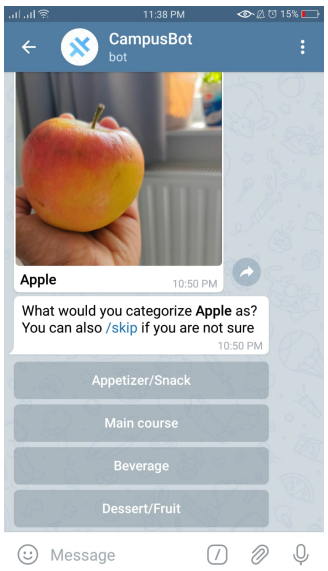


Figure 4.23: Inline keyboard and skip command

Inline keyboard
To integrate the keyboard directly into the message as depicted in Figure 4.23, inline keyboards in telegram have been used. This has been used instead of single/multiple select in mobile application.

Skipping a question

As shown in Figure 4.23, the feature to skip answering to a question in the text-based conversational interface is provided using a command. User can use /skip command if they do not want to answer a question or if they are not sure about the answer.

Handling interruptions

In text-based conversational interface, the feature to interrupt the task has been made possible using a command. /quit command can be used by the user at any point of time while performing a task to quit that task.

Reviewing and editing answers

By default, text-based conversational interface does not have the feature to review and

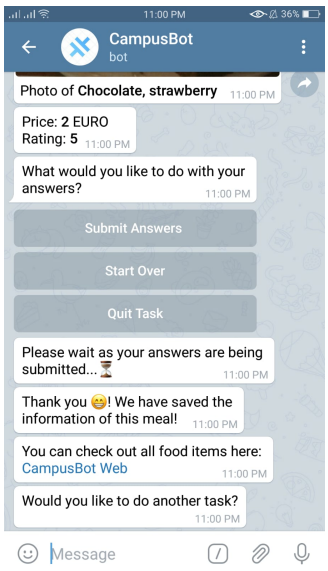


Figure 4.24: Edit answers

edit submitted answers. For this, chatbot sends a summary of answers provided by the participant to all the questions in the current task and provides an inline keyboard with options to Submit Answers, Start Over or Quit task as illustrated in Figure 4.24. User can review their answers from here and if they want to edit something they can start the task (Start Over).

Deployment

We deployed the text-based conversational interface on a virtual machine running with Ubuntu 18.04 operating system and using Gunicorn as the Web Server Gateway Interface (WSGI). To expose it to the public network, we used Ngrok which generates an url that acts a webhook which can be automatically called once there is any update. To read and write from text-based conversational interface to the database, we used Python client for Firestore. We shared the link to telegram telegram.me/v1_campusbot with the participants and to uniquely identify the participant to track their chatbot usage, we appended a unique id for /start parameter to the end of the link (telegram.me/v1_campusbot?start=19)

4.4.1 Conversation flow

After the user launches the chatbot, they would be presented with the start message represented in 4.25. The conversational flow after this is described below.

Start

When the user launches the text-based conversational interface for the first time, they

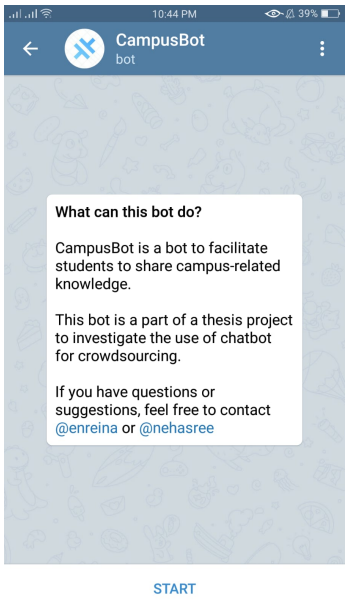


Figure 4.25: Start message

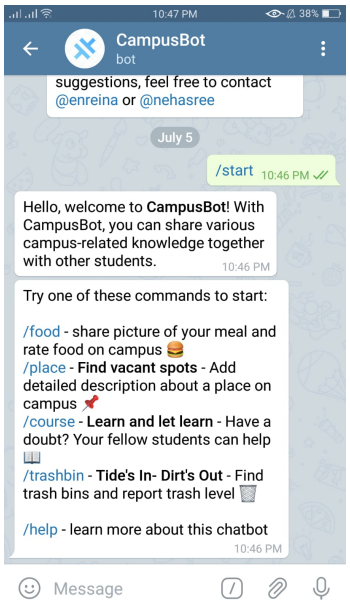


Figure 4.26: Menu message

can see the message illustrated in Figure 4.25 which has a brief introduction of the chatbot and contact details in case they have any queries or suggestions. They can start the bot by clicking on the *START* button. After clicking on this start button, as depicted in Figure 4.26, the chatbot sends a welcome message and another message that has commands for each task domain and also for help. Clicking on the /help command provides a detailed description of the chatbot and different domains of task. User can pick a domain by clicking on the corresponding command. This would lead to the *Task selection* flow.

Task selection flow

After the user sends a selected task domain command (For e.g. food), the chatbot would respond with a list of 5 tasks each in a different message (Figure 4.27 and 4.28) with task type. In another message, bot sends description of Enrich, Validate tasks, commands to refresh tasks and Create task (Fig 4.29).

As depicted in Figure 4.27 and 4.28, the message corresponding to a task consists of a command with task domain and task number (/food5, /food10), food items in the picture, picture of the food and also a task type label: Enrich/Validate. User can select the command with task number (/food5, /food10) to perform that task and depending upon selected task type: *Validate* or *Enrich task*, remaining flow would be

implemented. If the user clicks on /refresh command, they would be provided with a new message list of 5 tasks and clicking on /create would lead to *Create task* flow .

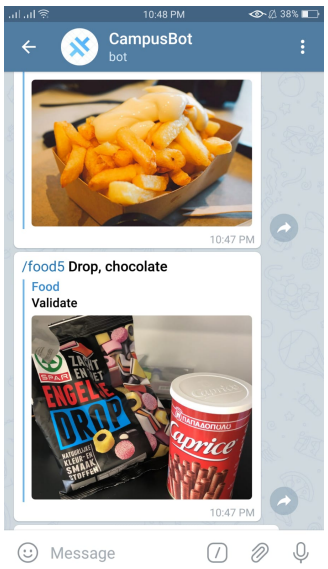


Figure 4.27: Validate task message

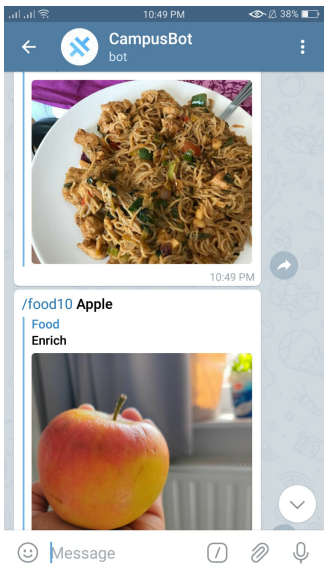


Figure 4.28: Enrich task message

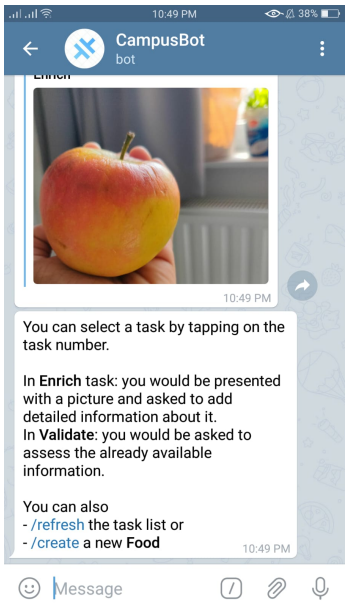


Figure 4.29: Task list

Create task

For Task Creation, as illustrated in Figure 4.30, the user would be asked to provide information about different attributes corresponding to the task domain designed in the section task description. After the user successfully answers all of them, the bot will execute *End of task flow*.

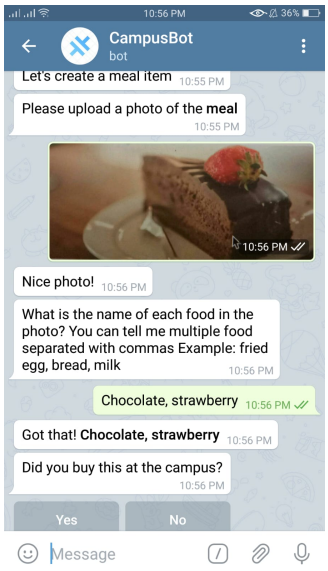


Figure 4.30: Create task

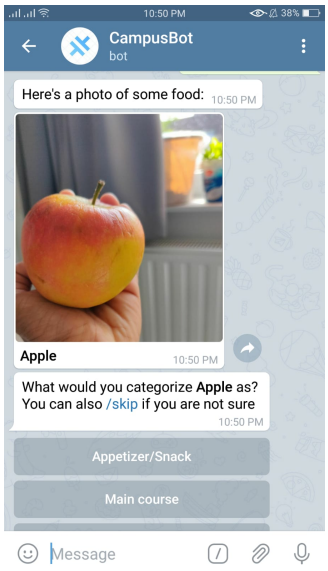


Figure 4.31: Enrich task

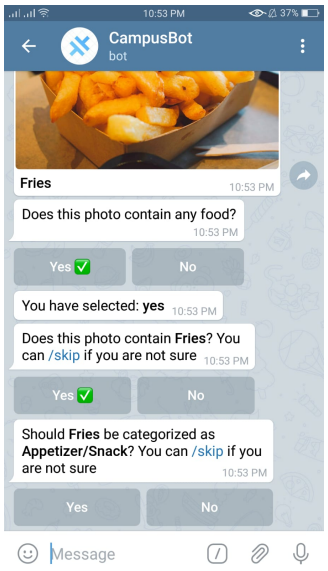


Figure 4.32: Validate task

Enrich task
In Task Enrichment, the user would be given details that are already available such as picture of item as depicted in Figure 4.31, and asked to provide more information based on that. After the user successfully answers all of them, the bot will execute *End of task flow*.

Validate task flow
As shown in Figure 4.32, in Task Validation the user would be asked to assess the given information. After the user successfully answers all of them, the bot will execute *End of task flow*.

End of task flow
After completing the task, the user would be provided with an overview of all their answers for current task and as illustrated in Figure 4.33, they would be asked if they want to submit these answers, edit answers or quit the current task. Also, they would be provided with a link to a web page (CampusBot Web) where they could see answers from their fellow users.

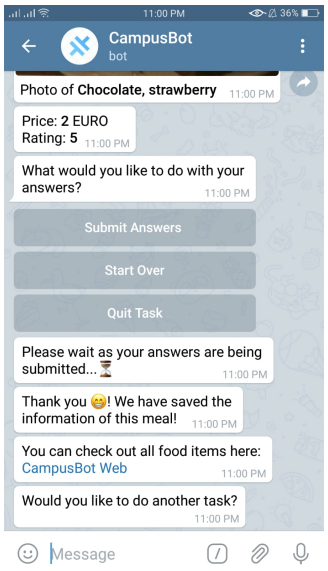


Figure 4.33: End of task

4.5 Crowd modules API

The API endpoints used to execute the task generation and assignment strategies, are implemented using Flask API. These endpoints that have been used by both mobile application and text-based conversational interface to trigger the tasks are described below in detail.

Generating an enrichment task:

POST /api/[food|place|question|trashbin]/generate-enrichment-task/<itemId>

This endpoint is hit through both the application after a new task has been created.

Generating a validation task:

POST /api/[food|place|question|trashbin]/generate-validationtask/<userId>/<enrichmentTaskInstanceId>

This point is hit every time the user completes an enrichment task. The API checks if there are required number of answers (numberOfAnswersRequired) for the task. If the condition is satisfied, the API aggregates all the answers and generates the validation tasks.

Assigning tasks:

POST /api/[food|place|question|trashbin]/assign-task/<taskId>

This endpoint is hit after the tasks are generated. API reads the list of users and assigns the tasks to the users selected through the task assignment strategy corresponding to the domain.

New user:

POST /api/[food|place|question|trashbin]/assign-task-to-user/<userId>

This endpoint is hit when the user registers/launches the application for the first time. It assigns the generated tasks to this user.

Task generation strategy
We designed the Task generation strategy on the basis of types of tasks and their expiration. After Creation task, Enrichment task is generated. But, as Place and Trash bin contain dynamic attributes, like current occupancy of the place or trash level of the bin, an expiration date has been set to the enrichment tasks in these domains. For place enrichment tasks, the expiration has been set to 1 day and for trash bins it has been set to 3 days. This has been set considering the dynamic attributes of these domains. Enrichment tasks created in these domains expire in this set time and new tasks for the same object are recreated.

After there are a sufficient number of answers in the enrichment task, Validation task is generated with the consolidated answers. The reason behind this strategy is: To add more information to an object (Enrichment), first the task needs to be created (Creation). And, as Validation task assesses the information, it needs data from both enrichment and creation tasks.

Task assignment strategy
We designed the Task assignment strategy based on the task domain. Table 4.6 summarizes the task assignment strategy for each domain. In all the domains, the task is not assigned to the user that created the task (author). In Food, the task is assigned to the first few users retrieved from the list of participants that is sorted in ascending order of number of tasks completed. In Courses, before sorting the list, the users are

filtering based on the course related to current task (Preferred courses). In Place and Trash bin, before sorting the list, the users are filtering based on the building name in current task (Preferred locations). The reason behind this is: for a Course task, the user is expected to have some knowledge about the course to complete the task and for Place and Trash bin task, the user is expected to be at the location or have an understanding about the location to complete the task.

Table 4.6: Task assignment strategies

Task domain	Task assignment strategy
Food	Sort users in ascending order of number of tasks completed - Author
Courses	Filter users by Preferred courses and sort in ascending order of number of tasks completed - Author
Place	Filter users by Preferred locations and sort in ascending order of number of tasks completed - Author
Trash bins	Filter users by Preferred locations and sort in ascending order of number of tasks completed - Author

Result aggregation strategy
Depending on the attribute, we used two result aggregation strategies. If it is a close ended question where list of options are already given to the user, the number of users who have picked an option also called as count is calculated. The final answer would be the option with majority count. If it is an open ended question, then all the answers from the user are saved. So the final answer in this case would contain a list of answers from the users.

Deployment
We deployed the Crowd system API on a virtual machine with Ubuntu 18.04 operating system and using Gunicorn as the Web Server Gateway Interface (WSGI). To expose it to the public network, we used Nginx as a reverse proxy with an already set up domain address. To read and write from API to the database, we used Python client for Firestore.

Chapter 5

Experiments & Results

In this chapter, we discuss in detail on how we evaluated the implemented Crowdsensing system which includes both text-based conversational interface and the mobile application. We conducted two experiments. In the first experiment, we conducted a naturalistic study where the participants were asked to use the application in their everyday life and we analyzed the system based on the usage of the interface. We would then discuss the results obtained from this experiment and also study the limitations. Further, to investigate the effect of these limitations on the results, we organized another experiment. In this, we experimented in a controlled environment where we would be monitoring the participants while they use the interface. For both the experiments, the location of the experiments is TU Delft campus and participants are bachelor and master students from TU Delft. We will explain these two experiments in detail in the following sections.

5.1 Goal

To answer RQ3, we conducted two experiments. Through these experiments we would like to investigate if the text-based conversational interface can be used as an alternative to a traditional mobile application for performing crowdsensing tasks and also investigate the differences in *User engagement* and *Usability* of these interfaces. Which means we use the independent variable *Interface type:* Mobile application and text-based conversational interface to measure the dependent variables *User engagement* and *Usability*.

The first experiment is a *between-subjects study* that includes both interfaces implemented in the previous chapter. The goal of this experiment is to find out if there is any significant difference between the User engagement and Usability of these two interfaces for performing crowdsensing tasks. To do this, we performed a quantitative analysis based on the usage of the app and a questionnaire and, to get a better perception of these results and understand the limitations, we conducted qualitative analysis. The second experiment is a *within-subjects study* whose objective is to study the effect of the limitations found on the results and also to validate the results.

5.2 Independent and Dependent variables

For a crowdsensing application to be successful, it has to be able to retain and engage the participants for a longer duration. So, in these experiments we intend to measure the variables *User engagement* and *Usability* with *Interface type* as the independent variable.

5.2.1 Interface type

As the main objective of this thesis is to explore the usage of the text-based conversational interface as an alternative to the traditional mobile application for performing crowdsensing tasks, we have two types of interfaces: Mobile App and Chatbot. For this experiment, we would be using the crowdsensing system implemented in the previous chapter.

5.2.2 User engagement

User engagement is used to measure whether the user is interested in the application. If a user chooses to use an application, this means that they are signaling that they found value in it. A highly engaged user generally uses the application for a longer duration thus contributing more data. For the current research, we would be using the app usage statistics and UES scale [27] to study different metrics that measure user engagement. All these metrics are discussed in the following section.

5.2.3 Usability

Usability is defined as *"the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use"*. It used to measure the User experience in terms of ease of use and ease of access to the application. If the application has good usability, the user would be able to easily become familiar with it and complete the tasks. In these experiments, we study the Usability of the applications using the SUS scale[3].

5.3 First experiment

This is a between-subjects study which means each participant would be using any one of the interfaces. We advertised the experiments in various bachelor and master courses held at TU Delft in Quarter 4 (May-July), 2019. Further, we also displayed digital and printed posters illustrated in Figure 5.1 all over the TU Delft campus and distributed flyers of the same posters among the students.

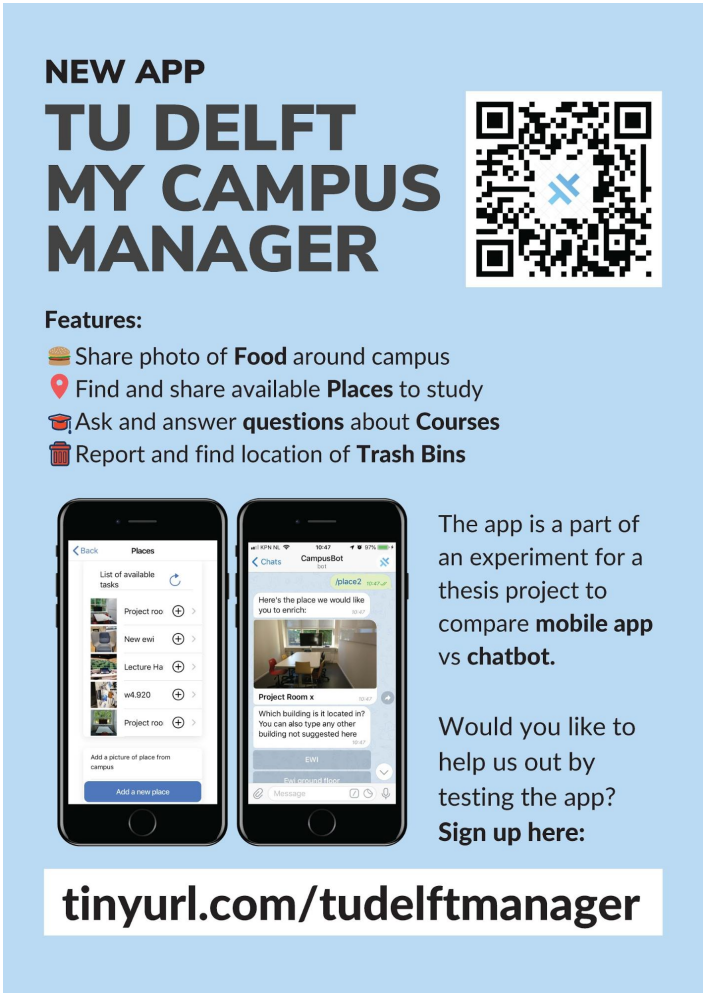


Figure 5.1: Poster

5.3.1 Participant Recruitment

We asked the participants to register through the google form which can be accessed through the QR code on the poster or by going to the link given: tinyurl.com/tudelftmanager. After the participant scans the QR code/ accesses the link, they would be redirected to a google form illustrated in Figure 5.2. This form contains a detailed description of the experiments and two questions regarding participant's email address and platform of the smartphone they use (iOS/ Android). If the participant is an android user they can submit the form else, before submitting the form, they would be redirected to another page depicted in Figure 5.3 where the device UDID is requested and the steps to get the UDID is mentioned.

After the participant successfully submits the form, we used *Stream quota sampling* to assign them to one of the two groups: Mobile app users and text-based conversational interface users. To implement this, we used a Google script that assigns the participants to one of these groups on a rotational basis as soon as they submit the form, so the first participant would be assigned to the mobile app interface and the second one to the text-based conversational interface and so on.

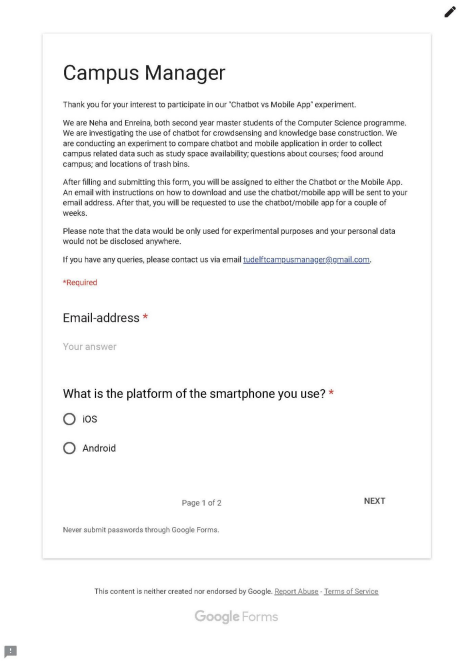


Figure 5.2: Google form initial

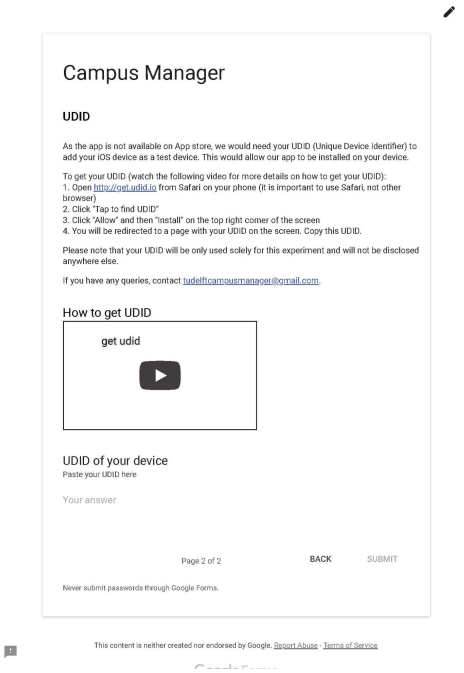


Figure 5.3: Google form UDID

After the participant is assigned to a group, we send an email to their registered address. This email has information about the assigned interface (chatbot/mobile app) as depicted in Figure 5.4 and also the instructions to download the corresponding application. After the participant downloads and launches the app, they can get started. Further, to make sure that the participants do not find an empty list of tasks when they first start the application, we pre-populated some tasks.

5.3.2 Quantitative analysis

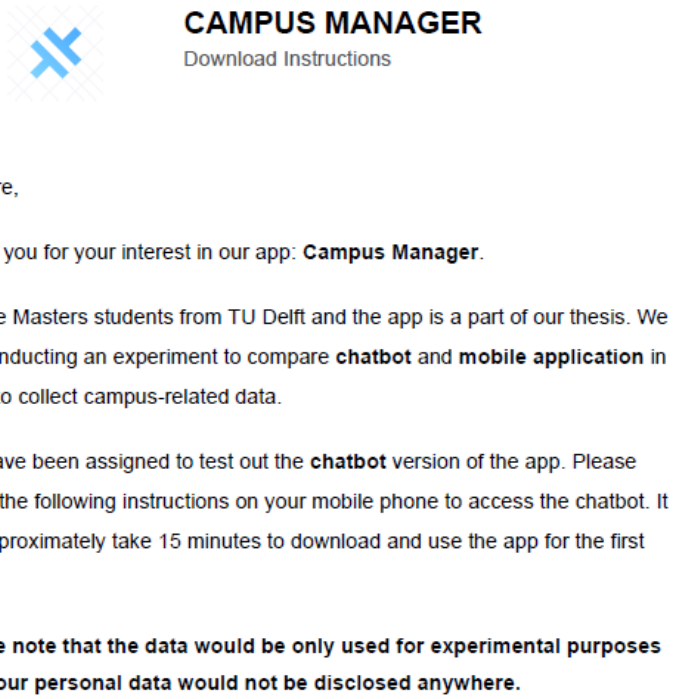
To perform the quantitative analysis on user engagement and usability of both interfaces, we used the app usage statistics and also a questionnaire to measure different metrics which are discussed below.

Based on App usage

To understand the difference between the mobile app and text-based conversational interface in terms of user engagement, we used metrics such as *User acquisition, Activation rate, Total number of tasks completed and Total time spent on performing tasks* which are discussed below.

User acquisition:
User acquisition (UA) refers to the act of gaining new users. This measures the percentage of participants that have installed the application.

$$UA = \frac{\text{Number of participants that installed the application}}{\text{Number of participants assigned to the application}} \tag{5.1}$$



CAMPUS MANAGER

Download Instructions

Hi there,

Thank you for your interest in our app: **Campus Manager**.

We are Masters students from TU Delft and the app is a part of our thesis. We are conducting an experiment to compare **chatbot** and **mobile application** in order to collect campus-related data.

You have been assigned to test out the **chatbot** version of the app. Please follow the following instructions on your mobile phone to access the chatbot. It will approximately take 15 minutes to download and use the app for the first time.

Please note that the data would be only used for experimental purposes and your personal data would not be disclosed anywhere.

Figure 5.4: Instructions to download the app

After registration through the google form, there were 44 participants assigned to the mobile application and 36 to the text-based conversational interface. Among the 44 participants of the mobile application, 16 of them installed the app and out of 36 participants for the text-based conversational interface, 24 launched the chatbot. So, in these experiments, while the UA rate for the mobile app is 36.36%, for the text-based conversational interface it is 66.67%. This shows that the participants are more convinced to install/launch the text-based conversational interface than the mobile app.

Activation rate

Activation rate (AR) refers to the act of gaining active users from acquired users. This measures the percentage of participants that have used the application after installing it.

$$AR = \frac{\text{Number of participants that used the application}}{\text{Number of participants that installed the application}} \tag{5.2}$$

Among the 16 participants who have installed the mobile application, 7 of them used the app and completed at least one task. On the other hand, out of 24 participants that launched the chatbot through telegram, 14 used it to complete at least one task. So, in these experiments, while the AR for the mobile application is 43.75%, for the text-based conversational interface it is 58.33%. This implies that the participants are more convinced to use the text-based conversational interface than the mobile application.

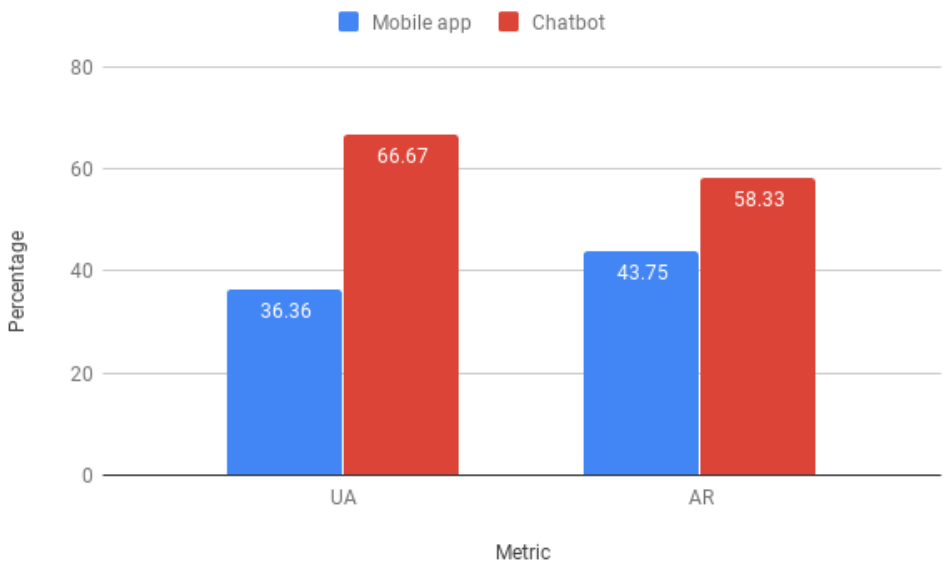


Figure 5.5: UA and AR comparision

Figure 5.5 illustrates the difference between the two interfaces in terms of User acquisition and Activation Rate.

Total number of tasks completed
Table 5.1 shows the total number of tasks completed by the participants per each task type in each domain for both the mobile application and text-based conversational interface. While participants using chatbot have completed a total of 71 tasks, participants using the mobile application completed only 24 tasks. This can imply that the participants using the text-based conversational interface are more engaged and motivated to complete the tasks than the participants using the mobile app. Figure 5.6 depicts the count of number of participants on y-axis and number of tasks completed on x-axis. It shows that out of 16 mobile application users, 15 of them completed less than 5 tasks and only one is in the range of 5-10 tasks. On the other hand, the text-based conversational interface has 3 users than have completed more than 10 tasks individually.

Table 5.1: Total number of tasks completed per interface

Domain	Task type	Mobile App	Chatbot
Food	Create	1	1
Food	Enrich	12	12
Food	Validate	2	1
Place	Create	7	9
Place	Enrich	0	34
Place	Validate	1	4
Course	Create	0	3
Course	Enrich	0	4
Course	Validate	1	2
Trash bin	Create	0	0
Trash bin	Enrich	0	1
Trash bin	Validate	0	0
Total		**24**	**71**

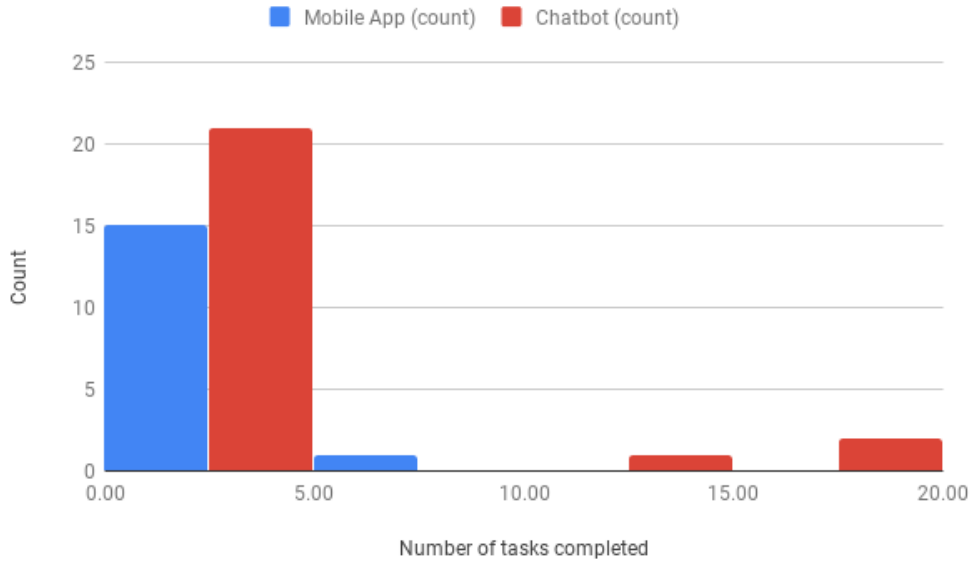


Figure 5.6: Count of number of tasks completed per participant

Total time spent on executing the tasks

The time spent (TS) on performing a task is calculated by measuring the difference in time in seconds between the start of a task and the submission of that task. The total time spent by a participant on an interface is the summation of TS of all the tasks completed by that participant. While the chatbot users spent a total of 7547.439 seconds on performing the tasks, participants using the mobile app spent 536.826 seconds. Figure 5.7 depicts a boxplot for each interface which illustrates the average time spent on executing the tasks by a participant.

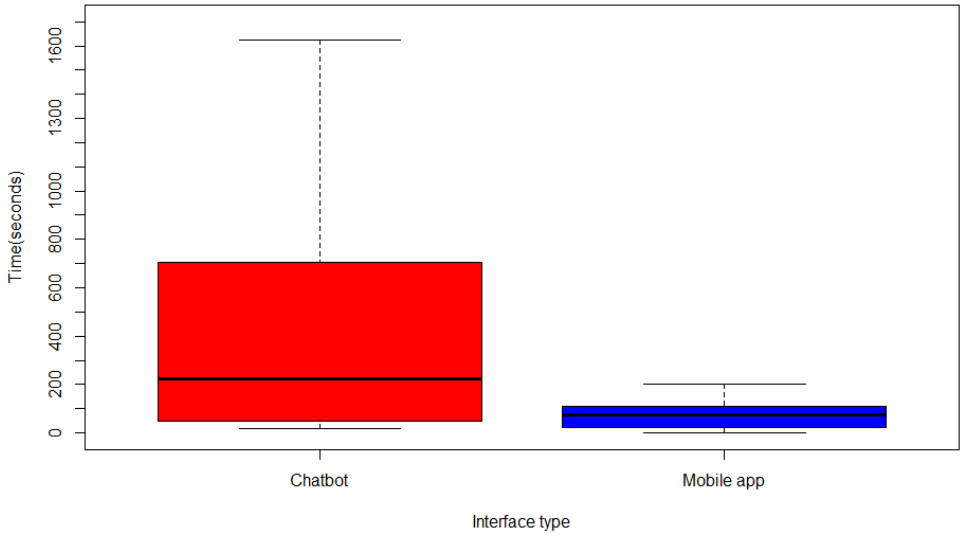


Figure 5.7: Total time spent on executing the tasks per interface

Output quality

We used accuracy as a metric and checked the output quality of the data provided by the participants. To measure this, we calculated the accuracy of the data by measuring the percentage of the number of correct answers to the number of answers provided. As we do not have the ground truth, we manually annotated and inspected the data. For subjective attributes such as route description to a place, we considered the answer as correct if it is meaningful. In this experiment, the accuracy of the data submitted by participants through the mobile application is 92% and through chatbot is 93.33%. This shows that the conversational interface can provide similar output quality as the mobile application.

Based on questionnaire

We also sent a questionnaire to the participants that have used the application. The questionnaire consists of a total of 18 Likert scale questions with 5 responses option(1:Strongly Agree, 5:Strongly Disagree). Out of these, 10 are to measure Usability and the remaining 8 measure different components of User engagement. The 8 User engagement questions are from User Engagement Scale(UES)[27] and are used to measure *User engagement* in terms of *Focused Attention, Perceived Usability, Aesthetic Appeal, Reward Factor*. These components are discussed in detail below.

Out of the 21 participants that have used the system(14: chatbot and 7: mobile application), 7 have completed the questionnaire out of which 5 used the text-based conversational interface and 2 used the mobile application. But, to ensure the quality of the output of the questionnaire, we only considered the participants that have spent at least 300 seconds in completing the tasks or completed more than 2 tasks using the interface. This resulted in 2 chatbot users and 1 mobile application user from the

participants that have completed the questionnaire.

Focused Attention
Focused Attention (FA) measures the feeling of absorbed in the interaction and losing track of time. We measured FA of the participants towards the interface through the questions: *The time I spent using the app just slipped away* and, *The experience of using the app was interesting*

Perceived Usability
Perceived usability (PU) measures the negative affect experienced by the participant as a result of the interaction with the application. To measure this, we asked the participants to express their opinion on the questions: *The app is confusing to use* and, *I felt frustrated while using the app*.

Aesthetic Appeal
Aesthetic appeal (AE) measures the attractiveness and visual appeal of the interface. To study this metric, we investigated the view of the participants on the questions:*The app is attractive* and, *The app appealed to my senses*

Reward Factor
If the participants feel that their time spent on the app is fruitful, they would use the application for a longer duration. Reward factor (RF) measures this. To analyze RF, we asked the participants if *they think the time they spent on the app is worth it* and if *their experience with the app is rewarding*.

Table 5.2 tabulates the score of four metrics discussed above for each interface. To calculate this score, we considered the average of the score given by the participants for questions corresponding to the metric. It can be interpreted from these numbers that the text-based conversational interface has better User engagement than the mobile application in terms of FA, PU, AE, and RF.

Table 5.2: FA, PU, AE and RF scores for Mobile application and Chatbot

Metric	Mobile Application	Chatbot
Focused Attention	2	4
Perceived Usability	3	1.5
Aesthetic Appeal	1	4.5
Reward Factor	2	3.75

Usability
We used the following 10 questions from the System Usability Scale (SUS)[3] to measure *Usability* of the interfaces.

- I would use the app frequently
- The app was easy to use

- I found the various features in the app were well-integrated
- I would imagine that most people would learn to use the app very quickly
- I felt very confident when I use the app
- I found the app unnecessarily complex
- I would need the support of a technical person to be able to use the app
- There is too much inconsistency in the app
- I found the app very awkward to use
- I needed to learn a lot of things before I could get going with the app

To find the final SUS score given by the participant to the interface they used, we calculated the sum of the score contributions from each question which ranges from 0 to 4. For the first five questions, the score contribution is the scale position(1-5) minus 1 and for the next five, the contribution is 5 minus the scale position. Finally, we multiply the sum of the scores by 2.5 to obtain the overall value[3]. We referred [1] to find the acceptability of the interface for the given SUS score range which is shown in Table 5.3.

Table 5.3: Acceptability of the interface

Score	**Acceptability**
<50	Not acceptable
50-70	Marginal
>70	Acceptable

As we only have the opinion of one participant that used the mobile application, we consider it as it is and for the chatbot, we calculated the average score given by two participants. In this survey, participants gave the mobile application a score of 62.5 and 90 to the chatbot. This implies that the acceptability level of the mobile application is *marginal* and that of a text-based conversational interface is *acceptable*.

5.3.3 Qualitative analysis

To understand the reasons behind the participation pattern of the students, we also conducted a qualitative analysis. Our main focus here is to understand *Why participants did not use/download the application?*. To study this, we divided the participants into two groups based on their participation level and asked them the following questions.

- Participants who have not downloaded/launched the application
  - What was the reason you did not download the app?
  - What would have convinced you to download the app?

- Participants who have downloaded/launched the application but not completed any task
  - What was the reason you did not use the app?
  - What would have convinced you to use the app?

The reasons found from this survey are categorized as follows,

Unavailability of the participants

The participants of this experiment who are students from TU Delft mentioned that they were busy with their courses and exams during the period of execution of the experiments. Some of them stated that they were momentarily occupied when they got the email invitation to participate in the experiment and forgot about it afterward. Moreover, some participants also referred that their absence on the TU Delft campus is the reason behind their inactivity.

Unsuitable tasks assigned to the participants

Some of the participants reported that they did not complete location-specific tasks related to the Place and Trashbin domains as they were provided with tasks from the locations they are not aware of or from the places that they do not visit often. This is because the user model is initially empty and does not have the preferred locations of the user. Moreover, participants who have completed a task from a particular location stated that they were provided with the task from the same location every day which deteriorated their interest in using the application.

5.3.4 Discussion

Through analysis of the results from the first experiment, we infer that a text-based conversational interface could indeed be an alternative to the traditional mobile application for performing crowdsensing tasks. The quantitative analysis of the app usage statistics shows that the chatbot has better *User Acquisition* and *Activation rate* than that of the mobile application. Moreover, the number of tasks performed and time spent on executing the tasks is more for the text-based conversational interface when compared to the mobile application. However, we observed that more than half of the tasks completed through the chatbot are performed by three participants.

Through a quantitative analysis on the questionnaire, we also showed that the chatbot is more desirable in terms of FA, PU, AE, RF, and Usability. But, out of 21 participants that used the system, only 7 responded to the questionnaire. To ensure the quality, we only considered the responses from 3 of them. However, we did not get a response from the three participants that have completed most of the tasks through the chatbot. As we sent the questionnaire in the holiday period, we suspect this as the reason behind less participation in the questionnaire.

We conducted a qualitative survey to study the reasons behind the participation patterns of the participants. Results from this analysis show the unavailability of the participants during the experiment period and assignment of unsuitable tasks to the participants are the reasons behind the non-involvement of the participants in the

experiment. However, these are experimental flaws and do not shed light on the comparison of the interfaces in the crowdsensing system. Moreover, because of the momentary failure of Google script that assigns the participants to an interface, we have an imbalance in the number of participants initially assigned to the mobile application (44) and the chatbot (36).

To understand if these limitations have influenced the results and also validate the results from this experiment, we organized another experiment.

5.4 Second experiment

This is a within-subjects study which means each participant would try both interfaces. To avoid any bias, we requested 10 TU Delft students who did not participate in the first experiment to participate in the second one. We also explained to them beforehand what is expected of them. We conducted this experiment in a more controlled environment where in each session we interviewed a participant in person. We provided all the participants with the same smartphone which has both applications needed for the experiment. Each interview lasts for an hour and the schedule is as follows:

1. 20 min: One interface
2. 8 min: Questionnaire related to that interface
3. 20 min: Another interface
4. 8 min: Questionnaire related to that interface
5. 4 min: Questionnaire comparing interfaces

We further divided these participants into two groups where one group uses the mobile application first and the other one uses chatbot first. We asked the participants to use the interface for 20 minutes and then asked questions regarding that interface. This questionnaire for FA, PU, AE, RF, and usability is the same as the one used in the first experiment. Following this, we asked them to use other interface for 20 minutes and asked them to fill in the same questionnaire for current interface. After the participant uses both the interfaces, we gave them a questionnaire in which we ask them to choose one of the given two interfaces to use in real life and also provide a reason behind the choice. Further, we also asked them suggestions on what more they expect from the selected interface.

5.4.1 Analysis

In this experiment we measured user engagement, usability of both interfaces and also made a comparative analysis.

User engagement

To measure this, we used the same 8 UES questions from the first experiment that are used to measure user engagement in terms of Focused Attention, Perceived Usability,

Aesthetic Appeal, and Reward Factor. Table 5.4 summarizes the average score and standard deviation of FA, PU, AE and RF of both interfaces. For all the metrics, the chatbot has a better score than the mobile application which confirms the analysis from the first experiment.

Table 5.4: FA, PU, AE and RF scores for Mobile application and Chatbot

Metric	Mobile Application	Chatbot
Focused Attention	2.45 ±0.71	4.13 ±0.83
Perceived Usability	1.95 ±1.02	1.25 ±0.45
Aesthetic Appeal	3.6 ±0.97	4.25±0.62
Reward Factor	3.55 ±1.08	4.335 ±0.51

Usability

We used the same 10 questions from the first experiment which are referred from SUS [3] to measure Usability of the interfaces. Figure 5.8 illustrates the SUS score given by each participant for both interfaces. To find the average SUS score of a interface, we calculated the mean of SUS score given by all participants. In this experiment, the mobile application got a score of 61 and the text-based conversational interface has 85. Which means the acceptability level of the mobile application is 'marginal' and that of chatbot is 'acceptable'. This validates the acceptability of usability of both interfaces found in the first experiment.

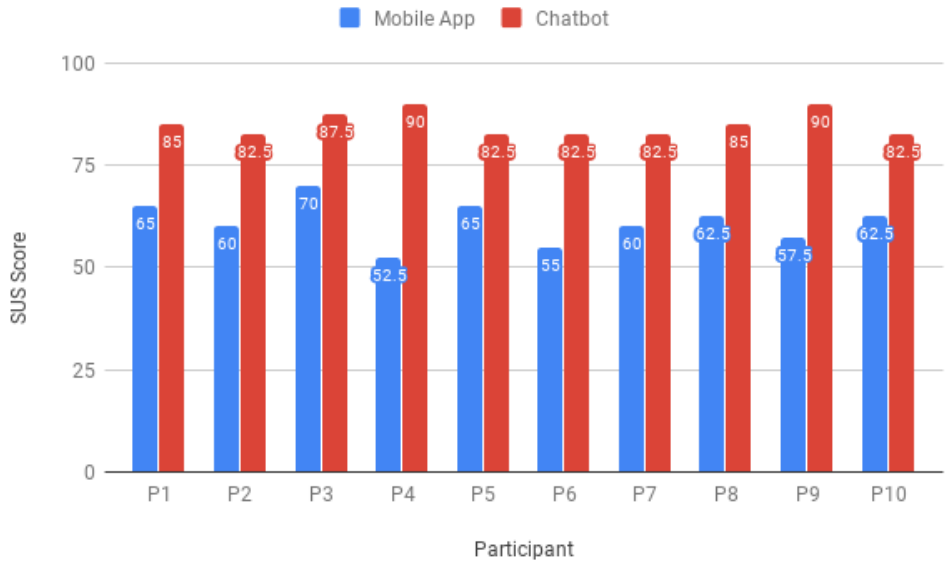


Figure 5.8: SUS Score comparison

Mobile Application vs Chatbot

We also asked the participants to mention with the reason the interface they would choose if they have it use it in everyday life. While all the 10 participants chose a text-based conversational interface over a mobile application, one of them (P3) mentioned he would prefer chatbot but he would not mind using the mobile application as well. The SUS score given by this participant also supports his statement. The reasons given by the participants are as follows,

- I have a lot of apps on my phone and do not want to download another one. I already have Telegram installed so I prefer Chatbot.
- Chatbot is more easy to use than mobile app.
- I like the UI of chatbot
- Chatbot is simple to use
- I like chatting
- In the mobile app, I had to navigate through a number of menu options to explore and perform tasks. But in chatbot it was really easy through messages
- I think chatbot is cool and trendy
- I am fond of trying new things so my first preference even before using them was chatbot. Even after using them i like the way chatbot works and would not mind using it on my phone.
- I like both of them but if I had to pick only one then I will go with chatbot
- Chatbot is more user friendly

5.4.2 Discussion

Results of this experiment show that the text-based conversational interface has better user engagement and usability than mobile application. Moreover, participants are more interested in chatbot and prefer it over the latter. By analysing this we found out that the ease of use, availability on social messaging platform and human like conversation are the main reasons behind their choice.

5.5 Discussion

The analysis of the results from the first experiment shows that the text-based conversational interface has better user engagement than the mobile application in terms of FA, PU, AE, and RF. Moreover, the acceptability level of the mobile application was *marginal* and chatbot was *acceptable.* Results from the second experiments validated and confirmed this.

As a result of both experiments, we conclude that the text-based conversational interface can be used as an alternative to the mobile application to execute crowdsensing tasks. A crowdsensing application with the chatbot interface has better

user engagement and usability than the mobile application. However, we state the following potential threats to the validity of our results:

Version issues: After the experiments started, we made some changes in both applications based on the bugs reported by the participants. While the chatbot launched through Telegram is automatically updated, we had to notify the mobile application users about the updated version. This makes us unsure if the participants were using the latest version of the mobile application. Moreover, we acknowledge that despite our best effort, the quality of the mobile application interface may not resemble the look and feel of the latest mobile applications.

Other sensors: In this experiment we designed tasks to use the Camera, GPS, and participant as a sensor. However, we acknowledge the potential to explore tasks involving other sensors such as accelerometer and gyroscope.

Representative sample: The participants of the experiment are students in the age group of 18-26. This does represent the whole population. Thus, the result may not pertain to the people in other age groups.

Input type in Chatbot: The chatbot we implemented in the crowdsensing system is a button-based chatbot. But there is also a possibility to use other types of input such as speech.

Chapter 6

Conclusion & Future work

In this chapter we conclude the research and provide an answer to our main research question: ***How can a text-based conversational interface be used as an alternative to a mobile application to execute crowdsensing tasks?*** in Section 6.1 and then we discuss the future opportunities in progressing usage of text-based conversational interfaces for crowdsensing applications in Section 6.2.

6.1 Conclusion

Recently there has been lots of research done in the field of crowdsensing especially on incorporating crowdsensing into the real-world applications through a mobile application. However, most of these applications are not as successful as expected. We observed that app fatigue and non-consideration of the human factor are some of the main reasons behind this. On the other hand, we are witnessing widespread interest in text-based conversational agents especially because of their ability to engage the users and their availability on popular messaging platforms. Thus, we raised the question of the applicability of the text-based conversational interface as an alternative to a mobile application to execute crowdsensing tasks. To answer this question, we organized the research around the following three sub-questions:

RSQ 1: What is the state-of-art in mobile crowdsensing?
From the literature, we studied different components of crowdsensing and also the evolution of crowdsensing techniques from the usage of an add-on sensor device to the smartphone as a sensor. However, previous works have not considered *User engagement*, thus limiting the capability of the crowdsensing application to attract and retain users. We propose the usage of a text-based conversational agent to overcome the current limitations of crowdsensing applications. Inspired by this study, we designed a crowdsensing system that supports both the mobile application and text-based conversational interface. In this thesis, our primary focus is towards the exploration of the usage of a conversational interface to execute crowdsensing tasks. We implemented, deployed this crowdsensing system and used it for the experiments we designed.

RSQ 2: How can a text-based conversational interface be designed to provide different elements and features of a traditional mobile application?
To study the comparison between the usage of Mobile and Chatbot interface to execute crowdsensing tasks, we designed and implemented the crowdsensing system that supports both interfaces. To be able to perform similar tasks in both interfaces, we developed a mobile application according to the state-of-the-art research and designed and implemented a text-based conversational interface by mapping various features and elements of the traditional mobile application.

RSQ 3: How does the conversational interface affect user engagement and usability in a crowdsensing application?
To study the differences between the two interfaces in terms of user engagement and usability, we conducted two experiments in the TU Delft campus with the students as participants.

In the first experiment, we organized a between-subjects study with 80 participants. We measured user engagement and usability of both the interfaces based on the application usage statistics and a questionnaire in a quantitative fashion. The results show that the text-based conversational interface has better user engagement than the mobile application in terms of user acquisition, activation rate, total time spent on performing tasks, the total number of tasks executed, focused attention, perceived usability, aesthetic appeal, and reward factor. Moreover, while the mobile application received system usability (SUS) score of 62.5, chatbot got 90. This implies that the acceptability level of the mobile application is *marginal* and that of the text-based conversational interface is *acceptable*. Further, to understand the reasons behind the participation patterns, we conducted a qualitative survey. Analysis of these results revealed that the main reasons behind the non-participation are the unavailability of the students and the assignment of inappropriate tasks. But these are experimental flaws and do not unveil much about the two interfaces in the crowdsensing system.

To deepen our analysis and validate the results of the first experiment, we organized a second experiment in a controlled environment with 10 participants. Results from this experiment show that all the participants unanimously expressed their preference in using chatbot over the mobile application to perform crowdsensing tasks. In this experiment, the mobile application received a SUS score of 61 and the text-based conversational interface got 85, which means the acceptability level of the mobile application is 'marginal' and that of the chatbot is 'acceptable'.

As a result of both experiments, we conclude that the text-based conversational interface can be used as an alternative to the mobile application to execute crowdsensing tasks and a crowdsensing application with the chatbot interface is more engaging than the one with the mobile application.

6.2 Future work

In this research, we have shown that the conversational interface can be used as an alternative to the mobile application to execute crowdsensing tasks and that it has better user engagement and usability than the latter. Moreover, the participants

preferred chatbot over the mobile application. The following list outlines potential future directions for this research,

- **Chatbot with NLP:** To implement the text-based conversational interface, we used response buttons for the users to interact with the chatbot. However, this decreases the user freedom as the conversation is more guided. A chatbot with NLP gives the user more freedom and makes the conversation look more natural.
- **More tasks:** We designed simple tasks in this research based on the location and participants of the experiment. To further enhance the application, other task types can also be explored.
- **Input type for the conversational agent:** In this research, we implemented a button-based chatbot. However, we see the opportunity to use other types of input as well. For example, it can be a voice-enabled chatbot where the users can interact with the application through speech.
- **Other complex sensors:** For the tasks designed in this experiment, we used Camera, GPS, and participant as a sensor. Current smartphones are also equipped with other sensors such as Accelerometer, Gyroscope, Barometer, Magnetometer, etc.,

Bibliography

[1] Aaron Bangor, Philip Kortum, and James Miller. Determining what individual sus scores mean: Adding an adjective rating scale. *Journal of usability studies*, 4(3):114–123, 2009.

[2] Petter Bae Brandtzaeg and Asbjørn Følstad. Why people use chatbots. In *International Conference on Internet Science*, pages 377–392. Springer, 2017.

[3] John Brooke et al. Sus-a quick and dirty usability scale. *Usability evaluation in industry*, 189(194):4–7, 1996.

[4] CENS/UCLA. Participatory sensing / urban sensing projects.

[5] Seth Cooper, Firas Khatib, Adrien Treuille, Janos Barbero, Jeehyung Lee, Michael Beenen, Andrew Leaver-Fay, David Baker, Zoran Popović, et al. Predicting protein structures with a multiplayer online game. *Nature*, 466(7307):756, 2010.

[6] Rodrigo De Oliveira and Nuria Oliver. Triplebeat: enhancing exercise performance with persuasion. In *Proceedings of the 10th international conference on Human computer interaction with mobile devices and services*, pages 255–264. ACM, 2008.

[7] Prabal Dutta, Paul M Aoki, Neil Kumar, Alan Mainwaring, Chris Myers, Wesley Willett, and Allison Woodruff. Common sense: participatory urban sensing using a network of handheld air quality monitors. In *Proceedings of the 7th ACM conference on embedded networked sensor systems*, pages 349–350. ACM, 2009.

[8] Raghu K Ganti, Fan Ye, and Hui Lei. Mobile crowdsensing: current state and future challenges. *IEEE Communications Magazine*, 49(11):32–39, 2011.

[9] Lin Gao, Fen Hou, and Jianwei Huang. Providing long-term participation incentive in participatory sensing. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, pages 2803–2811. IEEE, 2015.

[10] Ting-Hao Kenneth Huang, Walter S Lasecki, and Jeffrey P Bigham. Guardian: A crowd-powered spoken dialog system for web apis. In *Third AAAI conference on human computation and crowdsourcing*, 2015.

[11] Bret Hull, Vladimir Bychkovsky, Yang Zhang, Kevin Chen, Michel Goraczko, Allen Miu, Eugene Shih, Hari Balakrishnan, and Samuel Madden. Cartel: a distributed mobile sensor computing system. In *Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 125–138. ACM, 2006.

[12] Business Insider intelligence. Messaging apps are now bigger than social networks, 2015.

[13] Howe. J. The rise of crowdsourcing, 2006.

[14] Sunyoung Kim, Christine Robson, Thomas Zimmerman, Jeffrey Pierce, and Eben M Haber. Creek watch: pairing usefulness and usability for successful citizen science. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2125–2134. ACM, 2011.

[15] Nicholas D Lane, Emiliano Miluzzo, Hong Lu, Daniel Peebles, Tanzeem Choudhury, and Andrew T Campbell. A survey of mobile phone sensing. *IEEE Communications magazine*, 48(9):140–150, 2010.

[16] Walter S Lasecki, Rachel Wesley, Jeffrey Nichols, Anand Kulkarni, James F Allen, and Jeffrey P Bigham. Chorus: a crowd-powered conversational assistant. In *Proceedings of the 26th annual ACM symposium on User interface software and technology*, pages 151–162. ACM, 2013.

[17] Xulei Liang, Rong Ding, Mengxiang Lin, Lei Li, Xingchi Li, and Song Lu. Ci-bot: A hybrid chatbot enhanced by crowdsourcing. In *Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint Conference on Web and Big Data*, pages 195–203. Springer, 2017.

[18] Miguel Angel Luengo-Oroz, Asier Arranz, and John Frean. Crowdsourcing malaria parasite quantification: an online game for analyzing images of infected thick blood smears. *Journal of medical Internet research*, 14(6):e167, 2012.

[19] Suhas Mathur, Tong Jin, Nikhil Kasturirangan, Janani Chandrasekaran, Wenzhi Xue, Marco Gruteser, and Wade Trappe. Parknet: drive-by sensing of road-side parking statistics. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, pages 123–136. ACM, 2010.

[20] Sam Mavandadi, Stoyan Dimitrov, Steve Feng, Frank Yu, Uzair Sikora, Oguzhan Yaglidere, Swati Padmanabhan, Karin Nielsen, and Aydogan Ozcan. Distributed medical image analysis and diagnosis through crowd-sourced games: a malaria case study. *PloS one*, 7(5):e37245, 2012.

[21] Panagiotis Mavridis, Owen Huang, Sihang Qiu, Ujwal Gadiraju, and Alessandro Bozzon. Chatterbox: Conversational interfaces for microtask crowdsourcing. In *Proceedings of the 27th ACM Conference on User Modeling, Adaptation and Personalization*, pages 243–251. ACM, 2019.

[22] Emiliano Miluzzo, Nicholas D Lane, Kristóf Fodor, Ronald Peterson, Hong Lu, Mirco Musolesi, Shane B Eisenman, Xiao Zheng, and Andrew T Campbell. Sensing meets mobile social networks: the design, implementation and evaluation of the cenceme application. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 337–350. ACM, 2008.

[23] Prashanth Mohan, Venkata N Padmanabhan, and Ramachandran Ramjee. Nericell: rich monitoring of road and traffic conditions using mobile smartphones. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 323–336. ACM, 2008.

[24] Min Mun, Sasank Reddy, Katie Shilton, Nathan Yau, Jeff Burke, Deborah Estrin, Mark Hansen, Eric Howard, Ruth West, and Péter Boda. Peir, the personal environmental impact report, as a platform for participatory sensing systems research. In *Proceedings of the 7th international conference on Mobile systems, applications, and services*, pages 55–68. ACM, 2009.

[25] Nielsen. So many apps, so much more time for entertainment, 2015.

[26] Robert Ighodaro Ogie. Adopting incentive mechanisms for large-scale participation in mobile crowdsensing: from literature review to a conceptual framework. *Human-centric Computing and Information Sciences*, 6(1):24, 2016.

[27] Heather L OâBrien, Paul Cairns, and Mark Hall. A practical approach to measuring user engagement with the refined user engagement scale (ues) and new ues short form. *International Journal of Human-Computer Studies*, 112:28–39, 2018.

[28] Ming-Zher Poh, Kyunghee Kim, Andrew D Goessling, Nicholas C Swenson, and Rosalind W Picard. Heartphones: Sensor earphones and mobile application for non-obtrusive health monitoring. In *2009 International Symposium on Wearable Computers*, pages 153–154. Citeseer, 2009.

[29] M Jordan Raddick, Georgia Bracey, Pamela L Gay, Chris J Lintott, Phil Murray, Kevin Schawinski, Alexander S Szalay, and Jan Vandenberg. Galaxy zoo: Exploring the motivations of citizen science volunteers. *arXiv preprint arXiv:0909.2925*, 2009.

[30] Sasank Reddy, Andrew Parker, Josh Hyman, Jeff Burke, Deborah Estrin, and Mark Hansen. Image browsing, processing, and clustering for participatory sensing: lessons from a dietsense prototype. In *Proceedings of the 4th workshop on Embedded networked sensors*, pages 13–17. ACM, 2007.

[31] Jonathan Silvertown. A new dawn for citizen science. *Trends in ecology & evolution*, 24(9):467–471, 2009.

[32] Akash Singh, Faizy Ahsan, Mathieu Blanchette, and Jérôme Waldispühl. Lessons from an online massive genomics computer game. In *Fifth AAAI Conference on Human Computation and Crowdsourcing*, 2017.

[33] Michael Von Kaenel, Philipp Sommer, and Roger Wattenhofer. Ikarus: large-scale participatory sensing at high altitudes. In *Proceedings of the 12th Workshop on Mobile Computing Systems and Applications*, pages 63–68. ACM, 2011.