Vrije Universiteit Brussel

FACULTY OF SCIENCE
Department of Computer Science
Web & Information Systems Engineering

A Mobile, Context-aware Crowdsourcing Framework

Graduation thesis submitted in partial fulfilment of the requirements for the degree of Master in Computer Science

Elona Dervishi

Promotor: Prof. Dr. Olga De Troyer
Advisor: Dr. William Van Woensel

2013-2014

Acknowledgements

The road to this final thesis was a long but very interesting one. I have learned a lot about the world of crowdsourcing and the SCOUT framework. Furthermore, I have gained hands-on experience on Android and improved my programming skills in Java, PHP and MySQL. I would like to take this opportunity to thank some people that were involved in the process of finalizing my thesis.

First of all, I would like to thank Prof. Dr. Olga De Troyer for providing me the opportunity to study this topic. Furthermore, I also thank Dr. William Van Woensel for his feedback on the implementation part as well as on previous versions of this thesis. Without their guidance, support, feedback and suggestions, this research would not have been possible.

Finally, I would like to thank my parents and brother for their support, care and concern over the past few years during my education away from home. They have been a constant source of care and comfort.

Abstract

Over the years, the World Wide Web has evolved beyond being a platform for seeking information, and has become a ubiquitous medium supporting various forms of communication, collaboration, and peer to peer interactions, as well as the creation of user-generated content. In Web 2.0, most of the online information is produced by the regular end users around the globe. Wikipedia provides a compelling and well-known example, in which content is being created by a massive number of people on the Internet. YouTube is another example of a collaborative project, where many users share their videos on the Internet; Twitter has become an important channel of mass communication. In these examples, large numbers of users making small contributions led to a completely new type of application, enabled by the pervasive availability of the World Wide Web. End-user participation has significantly transformed the web, and paves the way towards crowdsourcing; where users are involved in producing small amounts of information and fulfilling tasks.

Crowdsourcing is a new paradigm for utilizing the power of “crowds” to facilitate large scale tasks, which are too costly, time consuming or complex to perform with traditional methods (e.g., automated reasoning). The concept describes a distributed problem-solving and product model, in which small tasks are outsourced to a group of people in the form of open call for solutions. Crowdsourcing tries to attract interested, capable and motivated crowds in return for incentives, which are often mainly small amounts of money. It combines the effort of the crowd workers, where each of them adds a small portion that combines into a greater result. Crowdsourcing has an enormous potential that can be truly unleashed when extended to sensor-rich mobile devices such as smartphones.

The past few years have witnessed the proliferation of smartphones standardly, equipped with Internet connectivity, location awareness and various utilities such as cameras, NFC readers and Bluetooth. People carry their phones with them the entire day, providing the opportunity to contribute at any time. Moreover, new form of contributions become possible, where contributors utilize their mobile hardware to supply sensed data such as light and temperature readings, and pictures of places or things. Mobile crowdsourcing offers even greater potential when automatically considering the mobile user’s context.

In general, context-aware applications take into account the user’s context (e.g., personal preferences, characteristics) and environment (e.g., nearby places and things) to enhance functionality and improve usability. In mobile settings, huge amounts of context become continuously available, increasing the opportunities for improving the user experience. In the case of crowdsourcing, context can be exploited to automatically push relevant tasks to *suitable* and *interested* users, based on their profile and surroundings. In doing so, both crowdourcer’s and contributor’s work is facilitated, making it easier to crowdsource tasks and to fulfil them.

In this dissertation, we present a framework for mobile, context-aware aware crowdsourcing. Compared to the state of the art, our framework is fully context-aware, taking into account the

user’s entire context and allowing a crowdsourcer to define context-sensitive tasks. In particular, it allows a crowdsourcer to create context-sensitive tasks or queries, which are then distributed to participants based on their location and profile. When defining a context-sensitive task, the crowdsourcer can choose the right type of workers, based on their age, location and level of education, habits, preferences or other attributes. This way, workers are provided with the right tasks at the right time, leading to a fully context-aware mobile experience. The framework consists of two mobile Android applications, one for each stakeholder (crowdsourcers and workers), as well as a server-side component for disseminating tasks. Furthermore, it is built on top of the SCOUT framework, a mobile context-aware framework that supplies a wide range of context information to mobile apps.

Contents

List of Listings

List of Figures

1. Introduction

This chapter presents the research context and problem statement, as well as the approach and structure of this dissertation.

1.1. Research Context

The World Wide Web has become an essential platform for seeking information, communication and collaboration for many users. In Web 2.0, content is no longer created and published by individuals, but is instead produced and modified by all users in a participatory and collaborative manner. These Web 2.0 applications, particularly blogs, wikis and social media, have been increasingly adopted by many people as they offer the opportunity for powerful information sharing, ease of collaboration and expansion of human knowledge. Wikipedia has turned out to be the largest and most popular general reference work on the Internet, because of the mutual contribution and sharing knowledge of different people. OpenStreetMap is an additional example where an online map is made available by the mutual content contribution of users in different locations.

The idea of user-generated content and mutual contribution has introduced the concept of crowdsourcing. Crowdsourcing is a relatively new phenomenon which enables organizations or end users confronted with a problem and desiring a goal to scale up the task environment and enormously expand the solver base by outsourcing the problem to an online community through the internet [1]. The problem that the crowdsourcing users need to solve varies from simple tasks such as tagging content and labelling images to more complex tasks such as translating, editing content or more investigative type of tasks (e.g., investigating the root cause of a problem, answering a research question). By performing the task, the user is often compensated by receiving a small amount of money. For instance, Amazon Mechanical Turk[1] is one of the most well-known crowdsourcing platforms, which allows users to outsource tasks to a large, flexible workplace of human workers.

Advances in mobile technologies offer huge potential for crowdsourcing. For instance, nowadays it is common that people spend a lot of time travelling, commuting or waiting for various events. As time is precious, mobile crowdsourcing provides a better solution by directly engaging people through their mobile devices any time of the day. Furthermore, as mobile devices are equipped with location-aware sensors or with GPS technology, the location information can be used for sending location-based tasks to their mobile phones. Utilities of mobile devices, such as sensors or cameras, make it possible to easily and directly capture data for crowdsourcing. Therefore, mobile phones support the idea of mobile crowdsourcing, in which the user’s context plays an important role.

[1] https://www.mturk.com/mturk/

Context-aware crowdsourcing applications introduce a technique for using sensing and inference in order to define the user's context and take appropriate action. This approach, which draws significant attention to what is happening around the user, is able to predict the user's needs and automatically select the relevant resources, thus simplifying the interactive process. With the combination of mobile devices, crowdsourcing can be extended beyond the traditional digital domain and be linked to tasks in the real world [2]. Context-aware crowdsourcing in mobile devices has been object of studies in different fields in computer science and it is being explored in several research projects. However, most of the current systems are only focusing on the location of the user, limiting them to simply being location-based services, without taking into account other elements of the user's context.

In our research, we present a context-aware crowdsourcing framework for mobile devices. The framework is able to automatically adopt its behaviour by presenting to the respective user only tasks that are relevant to the user's context (e.g., location, profile, characteristics) and allows a crowdsourcer to define a task in a context-sensitive way.

1.2. Problem Statement

Today, mobile users are searching for a more dynamic personalized experience that would allow them to locate and search information that is mostly relevant to them. In a mobile setting, users are reluctant or unwilling to spend a lot of time browsing and searching for information they need at a particular moment and time. For instance, when the user is walking around in his free time and wishes to view information of his favourite restaurant serving his favourite cuisine, or find out what items a specific shop sells, he is not willing to continuously browse and search for this information. Therefore, it is important that in a context-aware setting the user gets the information at the time and place he needs it with the minimal effort.

Context-aware crowdsourcing platforms should be able to automatically push relevant tasks to the most suitable users, taking into account the full user's context. However, in the current platforms the tasks are delivered to an undefined audience rather than to a specific group of people. Thus, the task assignment initiated by the crowdsourcer does not allow the system to have an influence upon assignments and to leverage the profile and characteristics of involved workers. In such settings, mobile workers can come and complete tasks posted by an individual or a company, and work for as long or as little as they wish. This flexibility and freedom of choice may result to several inefficiencies such as for instance the suitability of the participant for a specific task or the quality of a completed piece of work. Not all human workers are equal, nor is all human produced work. Some tasks, such as translating a paragraph into a specific language, are easier for some workers than others, requiring less effort to produce high quality results. Most of the systems deal with this issue by controlling the quality of a user's answers and accuracy. This is done by asking more than one person to complete one task and inferring the correct answer from the responses of multiple users [3]. However, inferring the correct answer from the responses of multiple error-prone respondents has been a problem addressed throughout a variety of academic literature. Concerning the up-to-date research on the field, there is no system that allows a crowdsourcer to dynamically choose the right type of users that his tasks will be distributed at.

Furthermore, another challenge that can be identified in the current context-aware crowdsourcing platforms is the fact that the crowdsourcer cannot utilize his own context (e.g. location) to specify tasks. Initiating the process of crowdsourcing requires the initiator to be sitting behind a desktop and upload tasks via a web interface. As a result, such approaches are not fully mobile and context-aware.

1.3. Approach

In the context-aware crowdsourcing domain, aspects of context that were initially investigated were related to where you are and what resources are nearby [4]. In this thesis, we propose a system which does not only consider nearby places and entities, but also the mobile user's characteristics, preferences and knowledge. As mentioned, workers might not be suitable for a task, or be able to provide high quality results for a particular topic. In this thesis, we tackle this issue by allowing the crowdsourcer to limit participation in the task to a particular group of workers. The crowdsourcer may do this by providing requirements based on personal profile, expertise and knowledge. Based on worker's profiles, relevant tasks are then automatically pushed to suitable workers, improving the overall result's quality.

In addition, the system allows the crowdsourcer to directly enter tasks on-the-fly, using a mobile application. In this respect, our context-aware contribution is further extended by also allowing the definition of tasks related to nearby places or objects. The crowdsourcer can utilize his own context to specify tasks (e.g., location based), without sitting behind a desktop. In doing so, we overcome current limitations of existing platforms and create a fully *mobile* and *context-aware* crowdsourcing system.

Our framework consists of two mobile context-aware applications, both built on top of SCOUT, which respectively allow crowdsourcers to specify tasks and workers to fulfil suitable tasks. In addition, the framework comprises a server-side application, which collects crowdsourced tasks and forwards them to relevant workers.

1.4. Structure

The remainder of this thesis is presented in the following way:

- The second chapter introduces the underlying concepts of this thesis, such as crowdsourcing, context-awareness, mobile context-awareness, SCOUT, RDF, SPARQL, etc.

- Chapter three discusses and compares related work.

- Chapter four elaborated on our approach.

- Chapter five illustrated our implementation of building a mobile context-aware platform on top of the SCOUT framework.

- Chapter six draws conclusions and presents some future work.

2. Background

This chapter discusses various topics which are the building blocks of this thesis. The topics vary from core concepts (crowdsourcing, context-awareness) to frameworks (SCOUT) and technologies (Semantic Web).

2.1. Crowdsourcing

The term ”Crowdsourcing” was introduced by Jeff Howe and Mark Robinson in a Wired Magazine article in 2006 [5]. Crowdsourcing is described as ‘a type of participative online activity in which a crowdsourcer proposes to solve a problem or perform a task, which is then solved by a group of individuals who bring their knowledge and experience together’ (Wikipedia). It is a form of “peer production” that outsources work traditionally performed by an employee to an “undefined, generally large group of people in the form of an open call” [6]. Notable examples of crowdsourcing tasks include evaluations (restaurants, books, and websites), expert tasks (translations, mathematical problems), voting tasks, or more investigative type of tasks.

2.1.1. Types of Crowdsourcing

Crowdsourcing is still in the phase of experiment and innovations. Different types of crowdsourcing exist in the domain and it is currently difficult to predict, which, if any, of these types will become dominant in the future. Howe [5] defines four basic categories of crowdsourcing applications. Crowdsourcing is a complex phenomenon, however, and often involves the combination of multiple categories, making them hard to distinguish. A short description can be found below.

- Crowd creation or user-generated content

  Crowd creation revolves around the creation of a new idea or an innovative process. The concept behind this type of crowdsourcing is that a large crowd holds much more creativity within it than an individual or a small group of people. Examples include platforms where people are asked to create new ideas or inputs regarding something new – logo, design, advertisement, etc. Pico Jobs [7] is a platform, which utilizes the crowd-creation power to allow companies to gather customer inputs, novel ideas and search for innovation.

- Crowd wisdom

  Crowd wisdom refers to the process of relying on the collective opinion of a group of people to solve complex problems. The idea is that the average opinion over a large number of responses is usually nearer to the truth. Some examples of crowd wisdom applications are Wikipedia[2], Yahoo[3], Yelp[4] and many forum communities.

- Crowd voting

  This form of crowdsourcing gives people the power to vote, judge or rank content such as newspapers, movies music, etc. It is one of the most popular forms of crowdsourcing, which generates the highest levels of participation. Crowd voting can be exemplified by platforms such as Threadless [8], which relies on the aggregation of people votes on existing T-shirt designs; depending on the number of votes the best designs will be printed.

- Crowd funding

  Crowd funding refers to contributions made by people in order to support efforts initiated by others [9]. This concept of crowdsourcing uses the "crowd" to obtain funds and feedback in order to support corporate activities. Crowd funding is utilized in platforms such as "Kickstarter[5]", which is a crowd funding company, where project creators can receive money from the crowd to realise their project.

2.2. Context and Context-Awareness

2.2.1. Context

Context-aware systems offer entirely new opportunities for application developers and end-users, by gathering context data and adapting system behaviour accordingly. Such context-aware systems adapt according to the location of the user, the people and objects that are nearby as well as to changes to such things over time. In order to be able to compose a specific definition of context-awareness, we will examine how context is defined in the literature, and present our definition and characterization.

Schilit and Theimer [10] define context as location, nearby people and objects and changes to those objects. A similar definition was introduced by Brown at al [11] who describes context as the nearby entities around the user, where an entity can be a person, object or place, the time of the day, the temperature, the season, etc. These definitions that describe context by using examples are difficult to apply. For the type of information that is not included in these examples it is difficult to determine whether it is listed in the definition of context or not.

[2] http://en.wikipedia.org/wiki/Main_Page
[3] http://be.yahoo.com/
[4] http://www.yelp.com/
[5] http://www.kickstarter.com/

Other definitions describe context as the user’s physical, social, emotional or informational state [12], or as the elements of user’s environment [13], or as the aspects of the user’s current situation [14]. These definitions are also difficult to apply as they simple use synonyms.

Dey and Abowd [15] stated that “Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves”. This definition does not have the aforementioned issues, and is also the most references one in the literature. Therefore, we will be using this definition in this thesis.

2.2.2. Context-awareness

The aforementioned definition of context allows a developer to easily determine if a specific type of information constitutes context. The developer simply has to determine whether that information can be used to describe the situation of a participant in an interaction; if so, then that information can be classified as context. As such, the definition of context makes it easier for an application developer to understand the boundaries of context-awareness, to enumerate the context for a given scenario and decide what features to implement in order to provide a context-aware experience to the user.

Context-aware was described by Schilit and Theimer [10] in 1994 as the software that “adapts according to its location of use, collection of nearby people and objects, as well as changes to those objects over time”. In other words, software that somehow adapts to the user’s context. Since then it has been a popular research topic and several definitions have been added to the academic literature. For example, Dey and Abowd [12] describe context-awareness as the “use of context to provide task-relevant information and services to users, wherever they may be”. Furthermore, Salber et al [16] express that context-aware is the ability to provide maximum flexibility of a service based on real-time sensing of context. Other definitions have simply provided synonyms for context: adaptive [17], reactive [18], responsive [19], context-sensitive [20] and others.

In this thesis, the definition of context-awareness that we adopt was firstly introduced by Dey et al in 1999 [21]. Dey stated that “A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user’s task”. We consider this definition as a reference for our work, since it is more general than the previous definitions and its main focus is on the user. With this definition, a system can be considered context-aware if it only responds to context; meaning adaptation, real-time sensing of context or reactivity are not mandatory. This definition can be useful for applications that do not adapt to context but simply reflect the context to the user or the ones that do not detect or sense context, allowing the detection and interpretation to be performed by other computing entities.

2.2.3. Mobile Context-awareness

Combining context-awareness with mobile devices can increase usability tremendously. Mobile devices have become increasingly sophisticated and equipped with a set of embedded sensors, such as wireless sensor networks, accelerometers, GPS and cameras that can be used to automatically sense the mobile user’s current context.

All these factors offer great potential to context-awareness and have introduced a whole new area of research and development.

2.3. SCOUT Framework

SCOUT is a mobile framework that allows the development of context-aware mobile applications that provide information and services to the user at the right time and place. It has the ability to exploit the knowledge on the user and his environment and be aware of the user’s context, his current physical environment and the people, objects and places in it. SCOUT uses detection techniques to locate information related to physical entities (people, places, objects) in the user’s environment. It is mainly based on web technologies for communication and data acquisition and it uses different sensing technologies to be aware of the user’s surrounding environment.

One of the state of the art aspects of SCOUT is that it does not depend on a single centralized server to detect this information, as it can acquire and integrate data from different decentralized sources. This can be interpreted by the fact that SCOUT is based on semantic web technologies to obtain useful environmental information. Often, information on the physical entities is already available on the web and each identifiable entity is responsible to provide and manage its own data and services in the form of a Web presence, such as for example a web site or an RDF file. Due to this, SCOUT is the ideal platform for deploying web presences. By exploiting semantic web standards and vocabularies to describe web presences in a uniform and expressive way, the SCOUT framework allows seamless integration and querying of data from several different entities, thereby providing applications with a richer and more complete view on the global environment [14].

2.3.1. Scout Layers

The SCOUT framework consists of a layered architecture and each layer is shortly described in the following section.

2.3.1.1. Detection Layer

The Detection layer is responsible for providing detection mechanisms such as RFID and Bluetooth that allow mobile devices to communicate with the entities in the user’s vicinity. These mechanisms are able to retrieve useful information about the entities and since they are independent of each other and implement a common interface, the framework can easily switch between them [14].

2.3.1.2. Location Management Layer

The responsibility of this layer is to determine whether or not an entity is nearby the user, depending on the results received from the detection layer. This is done by creating "positional" relations with nearby entities and validating these relations when the corresponding entities are no longer nearby [14]. Furthermore, this layer can listen for location updates and whenever the user's location changes the identification process repeats. This layer notifies the Environment layer (described below) passing along references to the entities online data sources.

2.3.1.3. Environmental Layer

The environmental layer provides an integrated view on the user, his environment and the physical entities in it. This layer maintains models, functionality and API's that allow the development of context-aware mobile applications. The models that it is composed of are described below.

The user model is responsible for holding the user's personal information such as preferences and characteristics, storing those using existing ontologies (e.g. FOAF). Furthermore, it allows application to pose queries to access the user model and retrieve specific information on the entity.

The proximity model keeps time-stamped positional relations between the user and the physical entities in combination with references to the entities' associated online data sources. Such a positional relation represents the fact that an entity is or has been nearby the user or another entity. The proximity model keeps the data up-to date, based on nearness and remoteness notifications from the Location Management layer. This involves adding new positional relations or invalidating existing ones, and recording the new locations of the entities.

The environmental model encompasses the User and Proximity model and extends them with information obtained from the physical entities online semantic sources. It enables a strong capability of SCOUT, as it combines metadata from multiple web presences, obtained from both past and current positional relations, and integrating it with the metadata of the mobile user [14]. Furthermore, this model it allows the client to query any piece of the user's context and environment, e.g. accessing the distributed information in one single query.

2.4. Building blocks of SCOUT

SCOUT utilizes Semantic Web Technologies to represent and retrieve context data. Below, we will shortly describe relevant concepts.

2.4.1. The Semantic Web

According to Tim Berners-Lee, the first generation of the internet, representing the Web 1.0, could be considered as the “read-only” web. In other words, the early web allowed users to search and read information, without involvement or interaction with the interface. With Web 1.0 the web master had constantly the responsibility of updating the web site to keep visitors informed and engaged.

The lack of active interaction of common users with the web led to the birth of Web 2.0. Following Berners-Lee’s method of description, currently users are experiencing the infancy of the Web 2.0, or the “read-write” web. The ability to contribute content and interact with other users has dramatically changed the landscape of the web in a short time. People are consuming as well as contributing information through blogs or sites, such as YouTube, Facebook, Flickr, etc.

Both these Web versions have been designed for direct human processing, but the next generation Web, which Tim Berners-Lee and others call the “Semantic Web”, aims at machine-readable information. The Semantic Web will enable intelligent services, such as information brokers, search agents, and information filters to directly consume data from the Web, offering greater functionality and interoperability than current stand-alone services. Web 3.0 will make it possible to make heterogeneous collections of different kind of data mutually interoperable. Information will be exchanged between applications, allowing programs to process and exchange content and information easily.

The Semantic Web will only be possible once further levels of interoperability have been established. Standards must be defined not only for the syntactic form of documents, but also for their semantic content. Notable among recent W3C standardization efforts are RDF and RDF Schema, OWL, SPARQL, which facilitate semantic interoperability. These standards are briefly described in the following section.

2.4.2. RDF

The RDF[6] (Resource Description Framework) is used to describe resources on the web, whereby a web resource is any concrete thing, person or concept identifiable via a URI (Uniform Resource Identifier). In particular, RDF allows making statements about those resources, in the form of subject-predicate-object expressions (known as RDF-triples). Importantly, the object of a statement may be a resource as well, possibly described by other statements. This gives rise

[6] http://www.w3.org/RDF/

to a contextual graph structure. An RDF graph may be serialized in a variety of formats, such as XML, N-TRIPLE, N3 or TURTLE.

2.4.3. RDFS

RDFS[7] (Resource Description Framework Schema), built on top of RDF, is the most basic schema language commonly used in the Semantic Web technology stack. RDFS allows defining vocabularies for describing web resources, including types and relations between those types. For this purpose, it supplies a meta vocabulary including terms such as “rdfs: Resource”, “rdfs: Property”, “rdfs: subClassOff”, etc.

2.4.4. OWL

The Web Ontology Language[8] (OWL), same as RDF Schema is a data modeling language that creates vocabularies to describe data represented in RDF format. Compared to RDFS, OWL provides a larger vocabulary for describing data, giving the ability to express more complicated and subtle constraints on those data. OWL adds, among others, cardinality restrictions, relations, between classes and more properties. OWL has three increasingly-expressive sublanguages: OWL Lite, OWL DL, and OWL Full.

- OWL Lite is used to define a classification and simple cardinality constraints.
- OWL DL supports maximum expressiveness and guarantees computational completeness.
- OWL Full supports maximum expressiveness, and makes no computational guarantees.

2.4.5. SPARQL

SPARQL[9] is both a query language and a protocol. It is used for querying RDF data and returns mapped result sets. As a protocol, SPARQL allows transmitting queries and results between a client and a SPARQL engine via HTTP requests.

[7] http://www.w3.org/TR/rdf-schema/
[8] http://www.w3.org/TR/owl-features/
[9] http://www.w3.org/TR/rdf-sparql-query/

3. Related Work

This chapter describes the current state of the art regarding the core topics and concepts of this thesis. In the first section, we discuss related work regarding mobile crowdsourcing. Afterwards, we discuss literature on context-aware crowdsourcing carried out on mobile devices.

3.1. Mobile Crowdsourcing

MClerck [22] is a new platform for mobile crowdsourcing in developing countries. With mClerck, workers can digitize handwritten words using mobile phones via a protocol that enables binary picture messages. The users receive an image of a word via SMS, and send back the typed version. The responses are aggregated into a digital document and the users are paid accordingly.

Another application, which is similar to mClerck, is MobileWorks [23]. MobileWorks is a crowdsourcing platform, where mobile users transcribe images of text on mobile phones. MobileWorks tasks are generated via processing scanned paper documents. MobileWorks's initiative was to expand the mircrotask market so that it can engage the members of the crowd no matter where they are and what kind of mobile device they have. Because of the fact that MobileWorks can be easily used from any kind of device, it is considered to be a simple way of creating self-organized crowds for gathering data. The difference from mClerck is that MobileWorks requires the use of a mobile web browser enabled by data connection and uses English as the primary text language, without supporting other languages.

VizWiz [24] is intended to provide people with visual problems with nearly-real time solutions to everyday issues. VizWiz allows blind users to take a picture with their phone, speak a question and then receive multiple spoken answers. VizWiz can, for example help visually impaired users answer questions about menus in a restaurant, find out the colour of a T-shirt, find free picnic spots in a park and other everyday problems. The tasks of VizWiz are posted to an already existing micro-tasking marketplace, namely Amazon Mechanical Turk[10]. Workers at Mechanical Turk are presented with a web-based interface, which includes the initial spoken question, the recorder question and the accompanying text. Answers are then submitted and sent to the mobile phone of the initial crowdsourcer, namely the visually impaired user.

Txteagle [3], is a mobile crowdsourcing tool that has been deployed in Kenya and is establishing a broader use worldwide. Same as mClerck, this system uses SMS text messages to provide tasks such as translations to local language dialects, audio transcription and market research. Txteagle supports the usage of proprietary messaging protocol with an integrated mechanism that allows the system to distribute tasks by asking users questions via text and paying them for their participation. Although txteagle functions via text messages, it does not support SMS

[10] https://www.mturk.com/mturk/

images to send graphical tasks. A stated advantage of txteagke and mClerck is that they support low-range mobile devices, as they only rely on SMS, which is an advantage in third-world countries.

Tasks in the aforementioned systems are limited to SMS images or document and imaging processing, while our framework, provides a richer and more dynamic interaction between the system and participants. In particular, we provide the ability to the crowdsourcer to directly enter a variety of tasks on the mobile device, potentially utilizing his context to facilitate task definition. Based on their profiles and context, workers automatically have suitable tasks (e.g., transcribing images in a certain language) forwarded to them.

3.2. Mobile Context-Aware Crowdsourcing

Our scope of study is related to context-aware crowdsourcing carried out on mobile platforms. However, most of the existing mobile approaches focus on enabling location-based tasks assigned and performed by humans. We provide a short description of the context-aware applications that exist in the research field.

MCrowd [25] is a mobile-based crowdsourcing platform that enables users to post and work on sensor-related crowdsourcing tasks. It enables mobile users to fully utilize the sensor-equipped iPhone to accomplish crowdsourcing tasks including image tagging, collecting images for a location (e.g., obtaining images for a landmark) and others. MCrowd is comprised of two systems; an iPhone based client available for workers and a backend website available for crowdsourcers to enter tasks, manage users and submitted data. The tasks are posted to Amazon Mechanical Turk via the mCrowd proxy, which are then shown in MTurk as a special category, easily identified by mCrowd users.

Alt et al [26] present another mobile platform by considering geographically-specific crowdsourcing tasks. They focus on location-based requests and match potential crowd workers based on their location. Workers are requested to perform real-world tasks, such as for example photo tagging, information tasks and action tasks. They conducted an evaluation and found that users tended to prefer tasks that are nearby their location and it was more pleasant to complete tasks that were related to photos.

Another platform that is worth mentioning is Askus [27], developed by Konomi. Askus takes into consideration the user’s location and distributes tasks based on the geographic conditions. It also includes information, such as the user’s availability (busy or available) when choosing potential crowd workers. Askus platform was not aware of much context, except from the user’s location and status. What is worth mentioning is that tasks are matched by the Askus system to specific workers based on their location and status; emails are then sent to those workers asking them to complete the task. The system pushes tasks to workers, rather than distributed them with the form of an open call.

Andrei Tamilin [28] has developed a crowdsourcing platform in order to create communication capabilities between public administration and citizens. They presented sensoRcivio, an application used by public administration to launch surveys, campaigns or questions that are redirected to citizens possessing a mobile phone. The application takes into consideration dimensions that can be determined by the sensing capabilities of the user's device, such as geographical dimensions and radius. Based on these dimensions tasks are being pushed to the users.

The previous works can be classified using two dimensions. The first dimension determines whether a system focuses on a particular goal (and thus type of task) or allows a variety of crowdsourcing tasks. The second dimension indicates the extent of the supplied context-awareness; for example are they only location-based approaches or do they allow for more personalization.

Regarding the first dimension, sensoRcivio was implemented for a particular goal, namely bringing the gap between public administration and citizens. The same can be said about the mobile crowdsourcing solutions of txtgeagle and mclerck, whose goal was to support image transcribing tasks with low-range mobile devices or WizWiz, whose purpose was to support visually impaired users. We distinguish from these researches, by proving a more generic approach, involving a larger user group with a larger set of tasks. Mcrowd, Askus and the system implemented by Alt et Al, also supply a more generic approach. However, mCrowd uses existing services to post tasks; we created a client-side mobile application allowing a crowdsourcer to directly specify tasks to be performed. Comparable to mCrowd, the tasks provided by Alt's et al platform had to be uploaded via a web interface.

Regarding the second dimension, which includes applications that support only the location-awareness in their approaches we distinguish in the following respects. Firstly, we go beyond using the sensors equipped with mobile devices by including further attributes such as the user's profile and surroundings. Secondly, another major difference is that with the systems that are purely location-based, tasks are distributed to an undefined group of people, while in our research we limit the participants based on their personal knowledge and profile. What is worth mentioning is that the tasks provided by Alt's et al platform, may be location-based, but the user had to manually input the nearby places where he prefers to take up a task. Based on this information, tasks are being generated to the users. However, in our platform we overcome this limitation by integrating the SCOUT platform, which enables the automatic sensing and detection of entities that are nearby the user's surrounding. Askus is the first platform that has a personalization aspect for the workers (busy or available). However, the context-awareness is limited to location and status, without taking into consideration other aspects of the user's context, such as personal characteristics, age, gender, favourite places, preferences, etc.

Finally, an important and shared drawback of the current state of the art is that crowdsourcers cannot utilize their own context to specify tasks (e.g., location), and need to be sitting behind a desktop to define tasks. In contrast, our framework provides a client-side, context-aware mobile application for crowdsourcers to define tasks. As a result, our approach can be considered fully *mobile* and *context-aware*.

4. Approach

In our research, we provide the users with a mobile context-aware crowdsourcing platform, which automatically senses context information (including location and profile) and presents the right tasks to the relevant users. Our framework consists of three main parts. Firstly, it includes two mobile context-aware applications, built on top of SCOUT, which respectively allow crowdsourcers to specify tasks and workers to fulfil suitable tasks. In addition, the framework comprises a server-side application, which collects crowdsourced tasks and forwards them to relevant workers.

Our general approach is the following:

- We provide a generic framework, allowing a crowdsourcer to define a variety of tasks.

- Our framework is *fully mobile*, as the crowdsourcer can enter new tasks via a mobile application and the worker can complete those tasks by simply using a mobile application.

- Lastly, our framework provides both types of stakeholders with a *fully context-aware* experience. On one hand, the crowdsourcer can reference his own context when defining tasks (e.g. location). On the other hand, the worker completes tasks suited towards his current context, which includes not only his location and nearby places, but also the worker's characteristics, preferences and profile.

In the next section, we illustrate the high level architecture of our framework. Afterwards, we summarize the roles and operations performed by the stakeholders. Then, we elaborate on the interfaces provided to each stakeholder.

4.1. Architectural Overview

The following high level architecture represents the logical view of our framework.

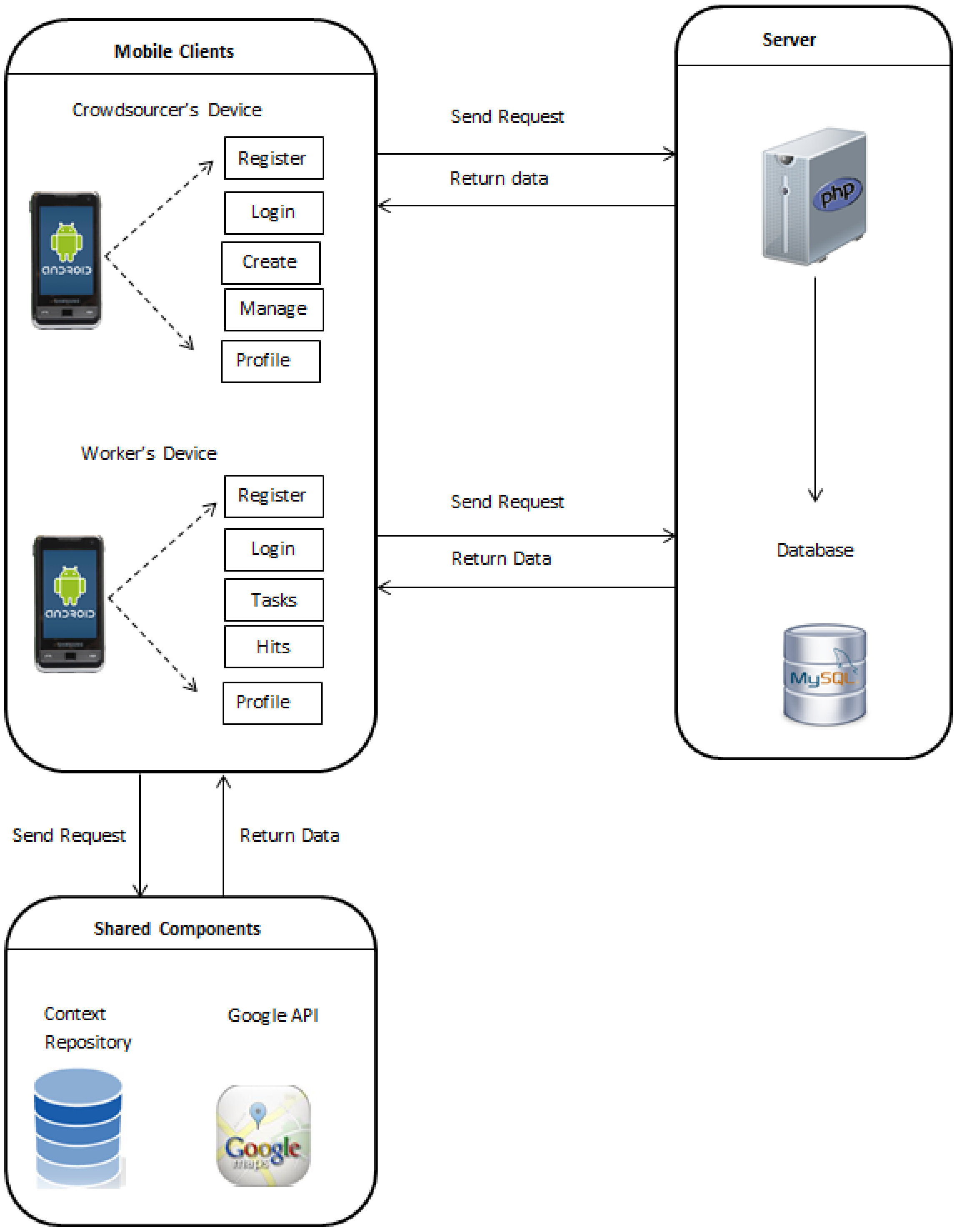


Figure 1: Architectural Overview

The framework consists of three components: (1) A mobile client application, used by crowdsourcers to define tasks associated with nearby places (or arbitrary geographic locations) and for certain kinds of workers. (2) A mobile client application used by workers, which pulls relevant tasks from the database depending on the user's location and profile. (3) A server, which is responsible for storing the tasks in the MySQL database and distributing them to the right workers. Note that these two mobile clients both rely on a context repository as well as the Google Maps API (currently, SCOUT is utilized as a context repository).

4.2. Roles and Operations

The figure below shows an overview of roles and their operations in the crowdsourcing process. Some of these operations, such as for example the operation "create task" are separated into several sub operations which will be discussed later.

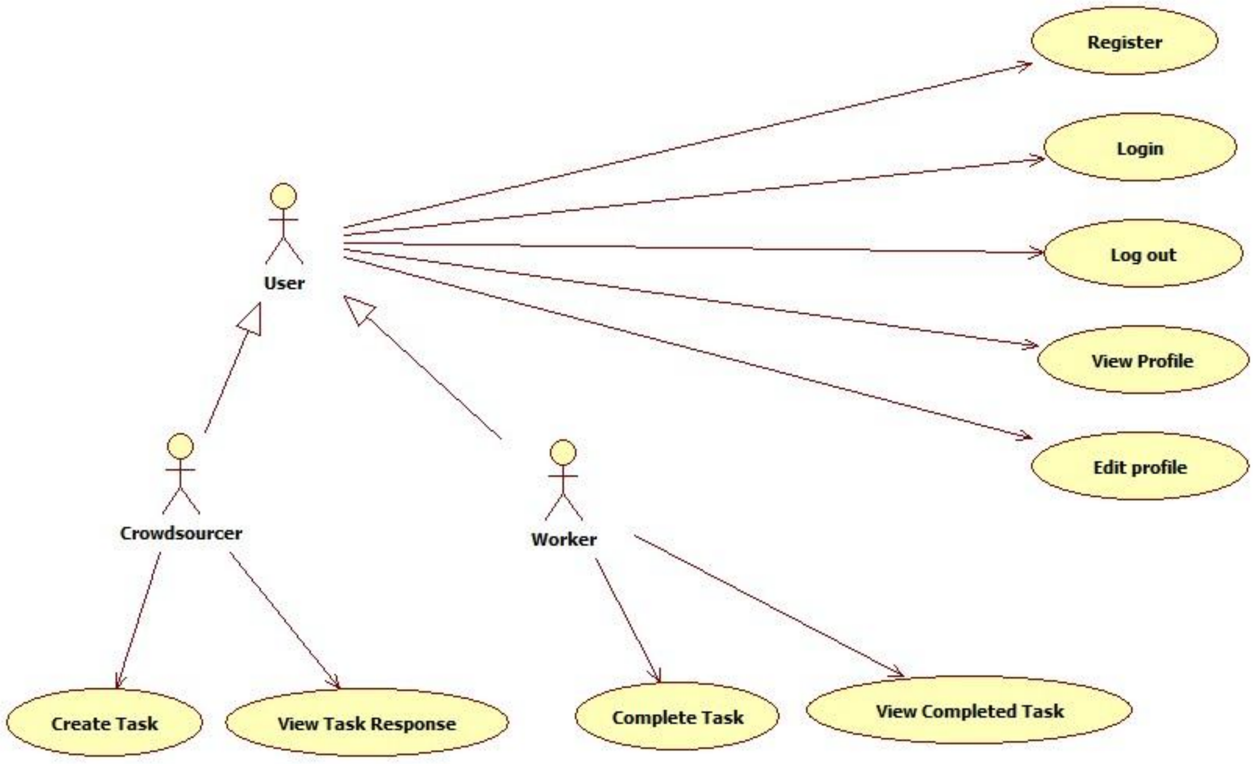


Figure 2: Roles and Operations

The crowdsourcer is a user that submits a task request – initiating the process of crowdsourcing, by specifying a task and accompanying criteria. He is also able to view the responses of his submitted task and manage his profile. The crowdsourcer has to register or login, before being able to interact with the system.

The worker is a member of the crowd that undertakes the execution of a task. The provided functionality includes registering or logging in to the system, performing a task based on certain criteria, view completed tasks and manage their profile.

4.3. Crowdsourcer's Mobile Application

This layer represents the user interface that is provided to the crowdsourcer.

The concrete functionality provided to the crowdsourcer is illustrated in the following sections. Since the registration process requires entering different information for crowdsourcers and workers, we also shortly discuss that task.

4.3.1. Register

The crowdsourcer must register in order to start using our system. The fields that are being stored in the database are the following:

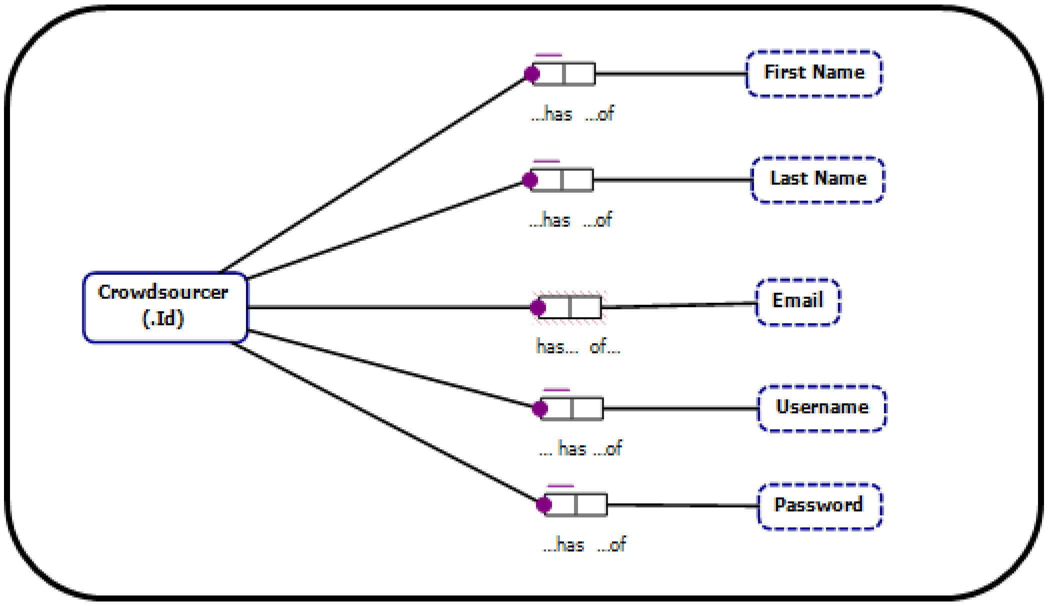


Figure 3: Crowdsourcer's registration

The above fields are all mandatory and error messages are being displayed to the user in case fields are left empty or not written properly (e.g. email, already existing username, and password requirements).

4.3.2. Create Task

Our framework gives the possibility to the crowdsourcer to create a task in three steps. Below, we illustrate the process that the crowdsourcer needs to follow in order to submit a task.

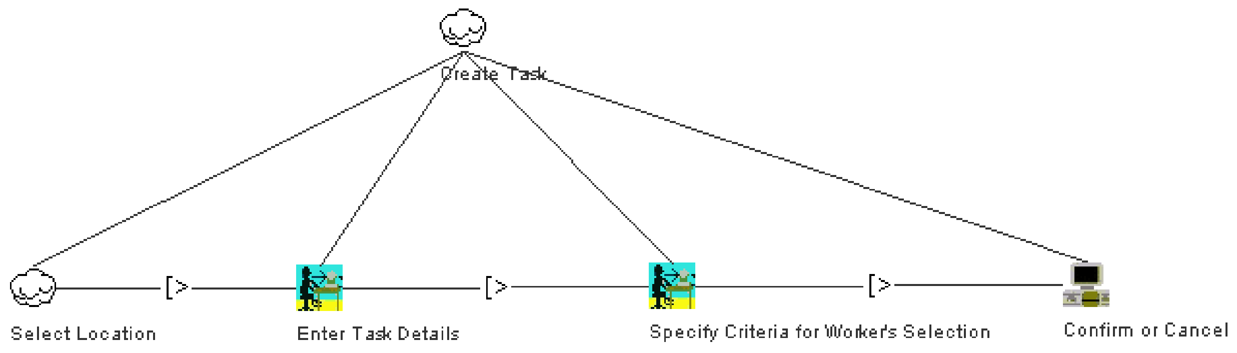


Figure 4: Create Task CTT

A crowdsourcing request has a central role in our system and implies a number of design requirements. Firstly, the framework must allow a crowdsourcer to select the location of the task, to parameterize its description and requirements and select the proper crowd.

Below we will elaborate and describe each of these steps in more details.

4.3.2.1. Select Location

Here the crowdsourcer is able to choose between two different options. The first option is to choose as task location any nearby places or things. As the crowdsourcer application is built on top of SCOUT framework, nearby entities can be queried and set as the location where the task has to take place. These entities may include places or objects that are nearby the user's current location, such as for example a nearby restaurant, cafeteria or university. The user can simply select one of these entities without having to provide further information (e.g., name, description), as they will be taken from the SCOUT framework. Secondly, we provide the possibility to the user to specify an explicit location for his task via Google Maps. This will allow him to define tasks not only around his current location, but also to other areas. For example the crowdsourcer might want to ask the crowd for the opening hours or cuisine of a restaurant that is not located nearby.

Below we illustrate the process that the crowdsourcer has to follow in order to select the location to define the task.

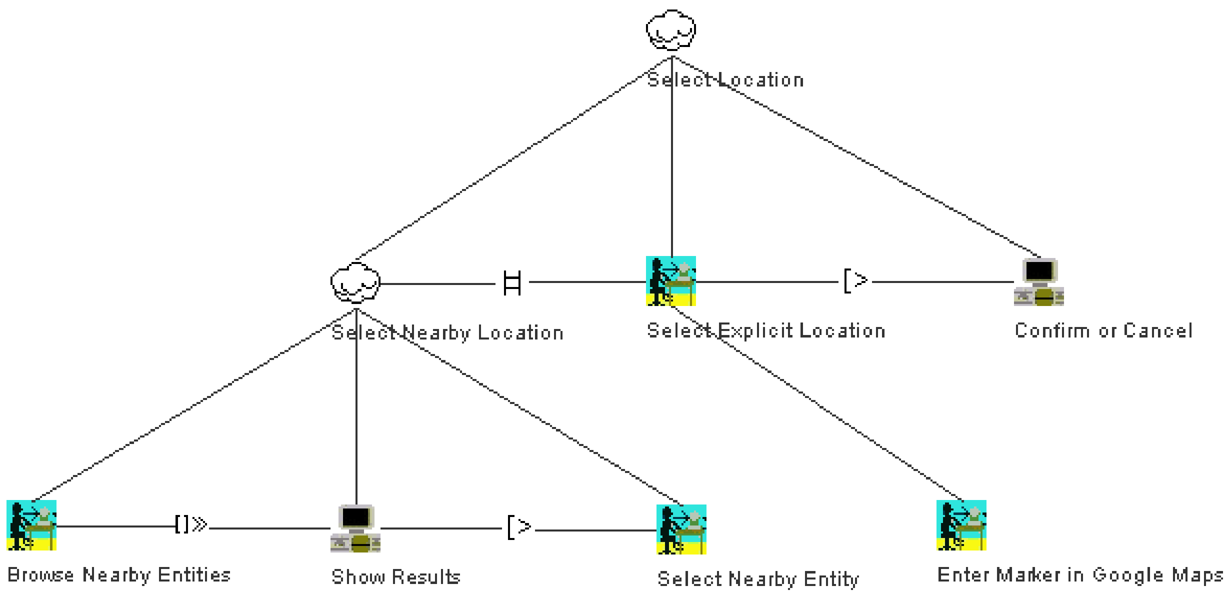


Figure 5: Select Location CTT

If the user chooses the first option, all the information related to the selected nearby entity (e.g., name, description, etc) will be queried from the SCOUT framework. If the user selects an explicit location, he has to put a marker in the Google Map.

4.3.2.2. Enter Task Details

Our framework currently supports five types of tasks: (i) image tagging tasks, for instance tagging objects in a query image; (ii) data collection tasks, for instance asking questions to the users for any kind of information; (iii) voting tasks for collecting the crowd's opinions and votes for places such as pubs, restaurants or cafes; (iv) translation tasks for translating a text from a given language to a required language; (v) temperature tasks, for instance "what is the temperature in the center of Brussels?" Each type of tasks has a different number of properties. Below we will describe the attributes of our tasks. Note that only the attributes entered via the form are shown (e.g., location attributes, result attributes are not shown). As you can view the "project name" and description are used for all task categories.

- Image Tagging Task

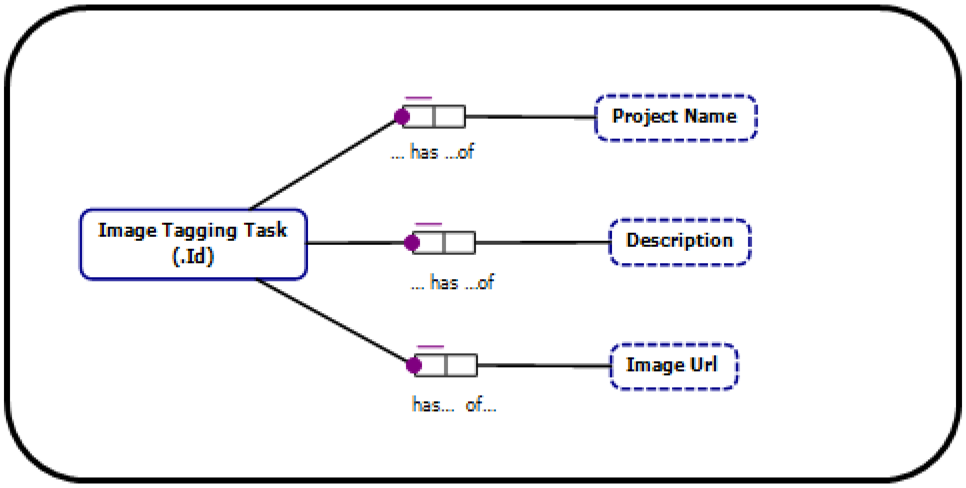


Figure 6: Crowdsourcer's image tagging task

- Data Collection Task

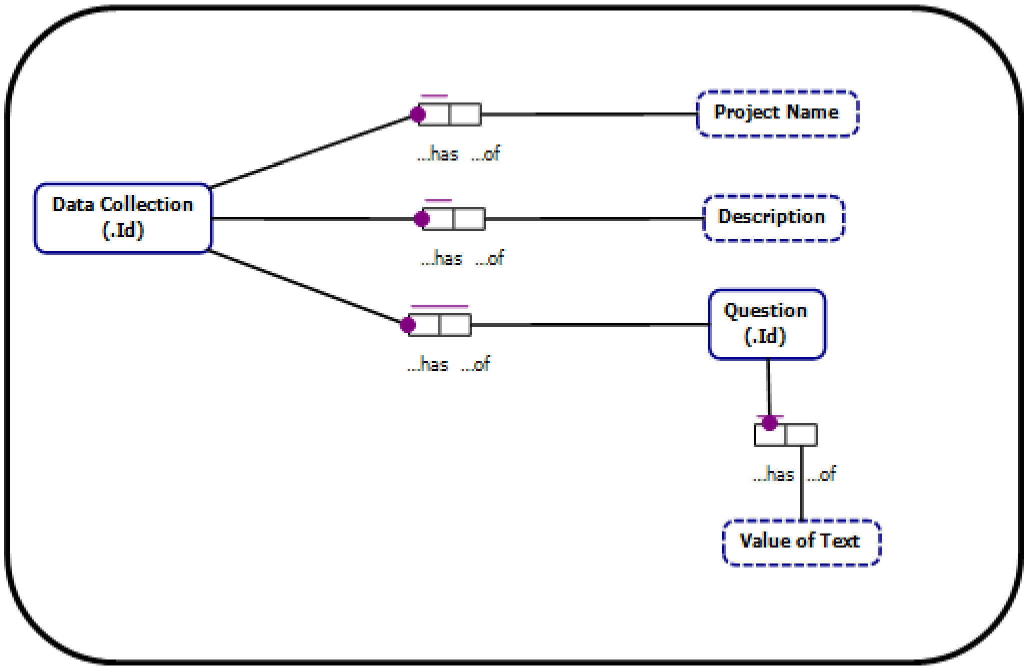


Figure 7: Crowdsourcer's data collection task

- Voting Task

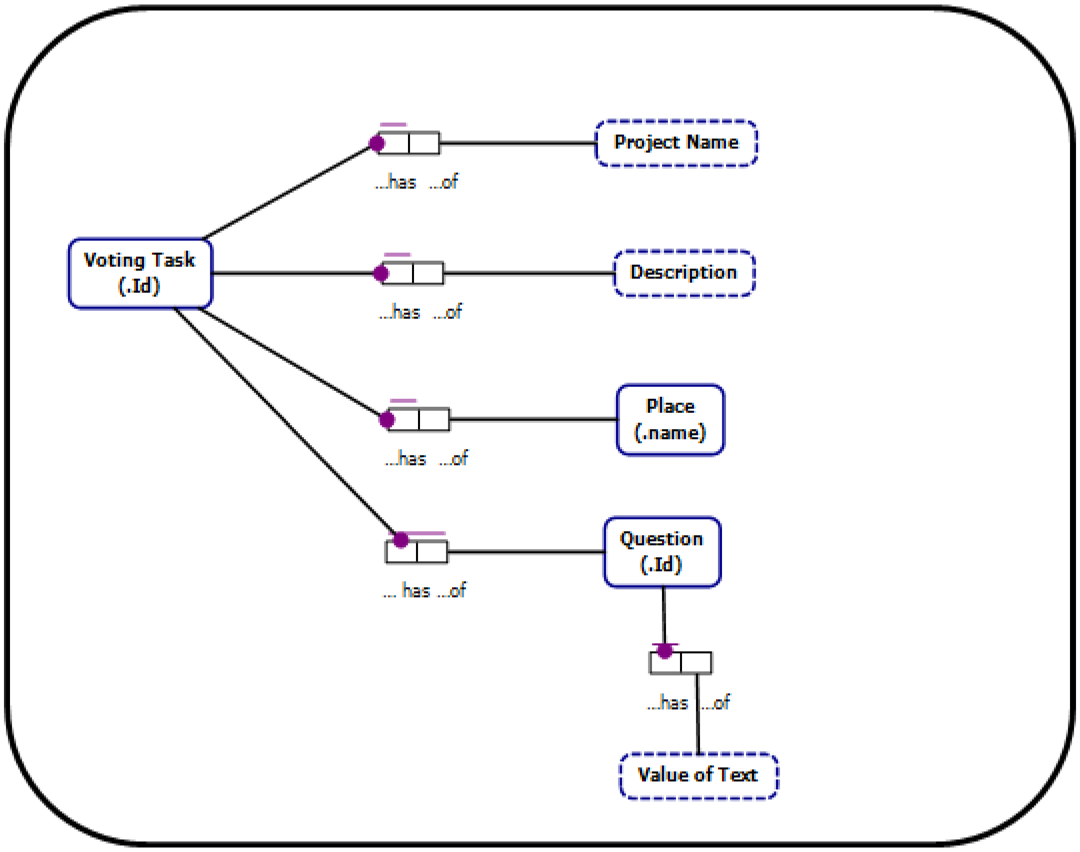


Figure 8: Crowdsourcer's voting task

Here it is worth mentioning that for the voting tasks there are some default attributes that will be shown to the worker, such as the rating box on a scale 1 out of 5 and a comment text.

- Translation Task

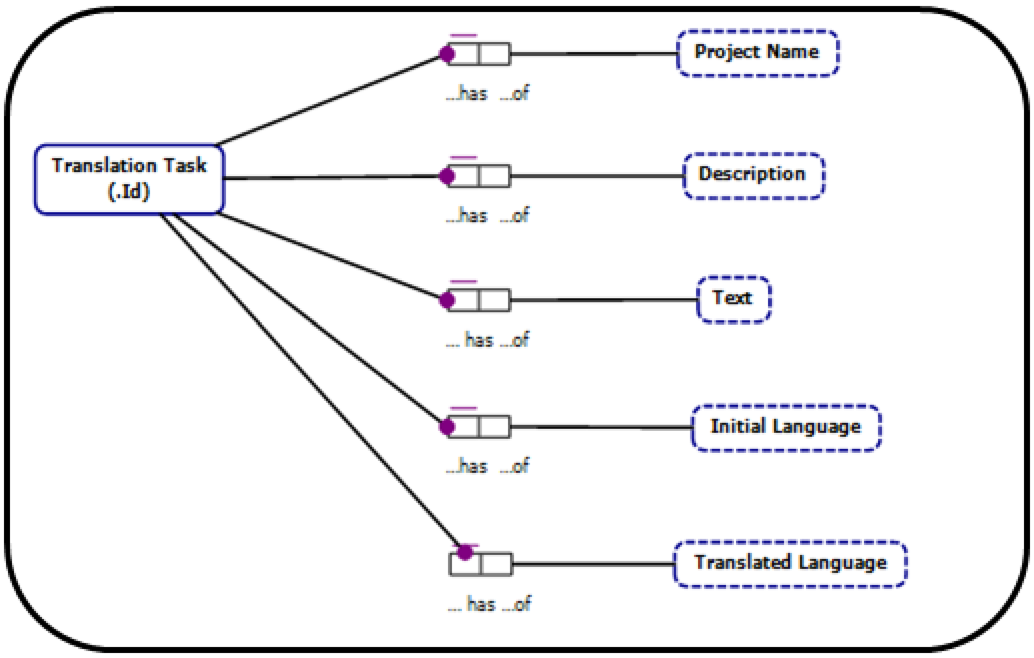


Figure 9: Crowdsourcer's translation task

- Temperature Task

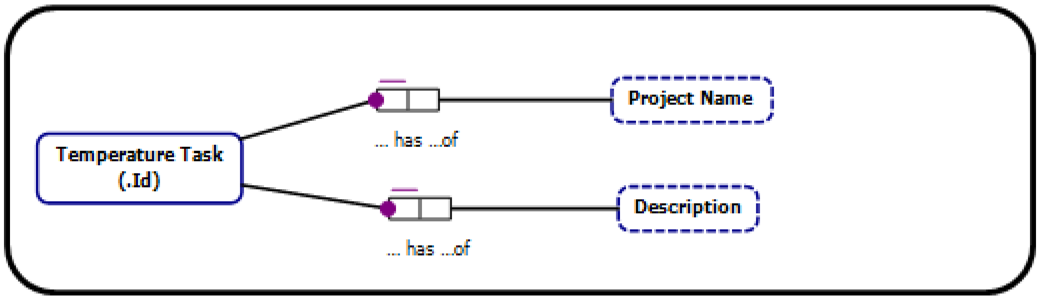


Figure 10: Crowdsourcer's temperature task

It is worth mentioning that in this category, we do not permit the worker to select nearby entities for collecting the temperature, as this would bring ambiguity to our system (e.g. an entity such as a restaurant cannot have a temperature). The user enters a marker in the Google Maps for the location that he wants to know the temperature of, and this location is considered as the task location (which is the section 4.3.2.1 for the rest of the tasks).

4.3.2.3. Specifying Criteria for Selecting Workers

Our system needs to include means for finding suitable workers, who are not only in the right location and place but also with the right profile and knowledge. Some tasks such as for example proofreading a paragraph of text and translating it to another language, are easier for some workers and require less effort and time (and the results will be of higher quality). Furthermore, some other tasks such as for instance requesting information related to restaurants or food could be better answered if they are addressed only to vegetarians or to people who prefer eating meat. Other tasks that may request crowd's opinion about pubs or cafeterias would be better replied by people who use to go often in such places. The attributes that the crowdsourcer has to select in order to deliver tasks to the right workers are the following:

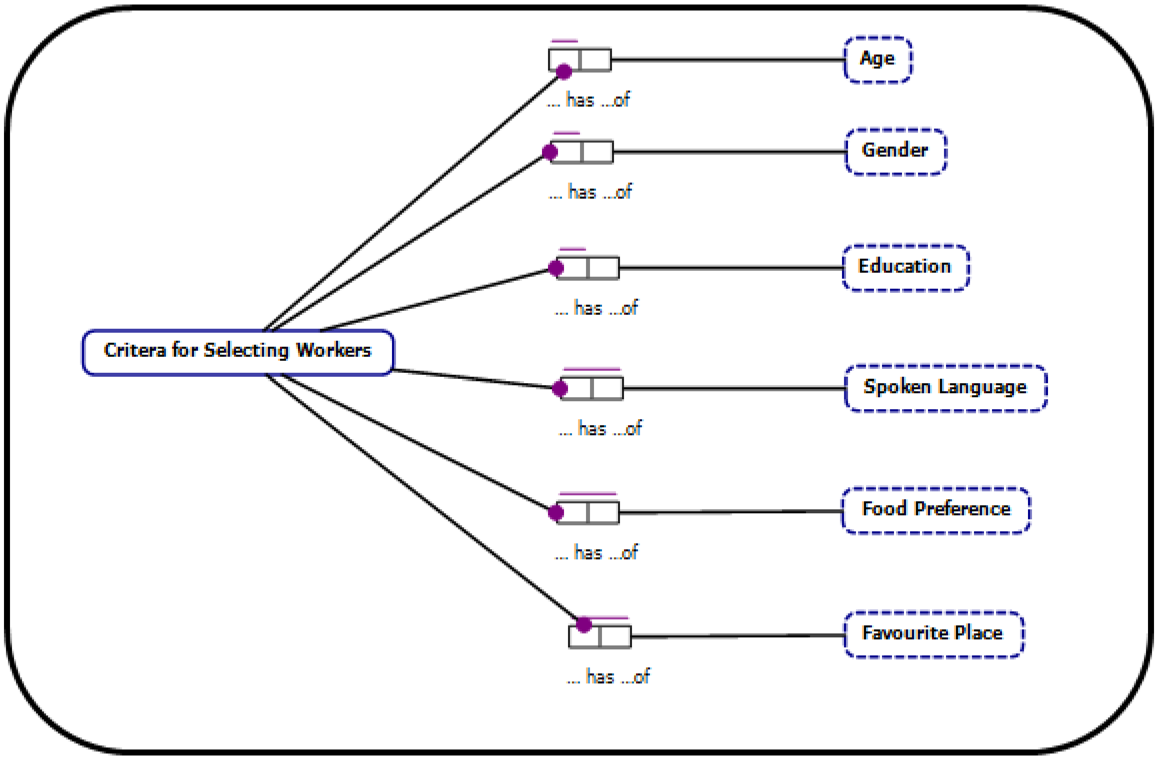


Figure 11: Specifying Criteria for Selecting Workers

4.3.3. View Tasks Response

The crowdsourcer's area also includes an overview of all tasks they previously specified, where they can easily track their tasks and find out if a task has been completed. They have the ability to view the responses of the workers.

4.4. Worker's Mobile Application

The functionality that is provided to the workers is illustrated in the following sections.

4.4.1. Register

Same as with the crowdsourcer, the worker has to be registered in order to start using our system. The fields that he has to enter are illustrated below. As our goal is to provide him with the most relevant tasks, the worker has also to enter some personal fields while registering.

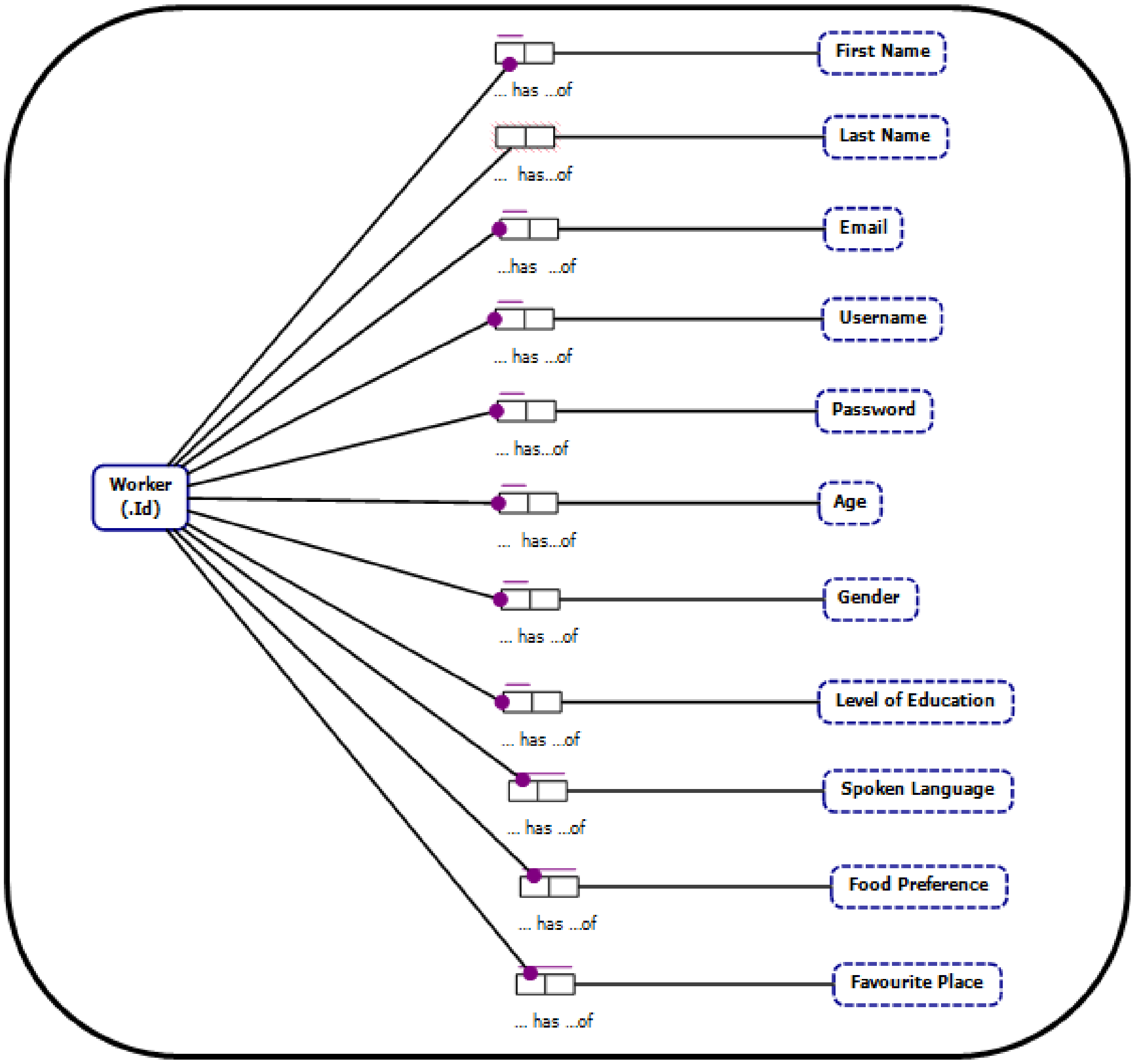


Figure 13: Worker's registration

4.4.2. Complete Task

For workers, the application lets users retrieve tasks from the database based on their location and profile attributes. It should be noted that, regardless of whether the crowdsourcer defined a task to nearby entities or an explicitly specified location, any issued task related to the user's current location (and suiting his profile) are pushed to him. Once the user chooses to solve a task, he will be provided with a particular form to be filled in, based on the task's type (see section 4.3.2.2.). Below we provide an overview of the form fields.

- Image Tagging Task

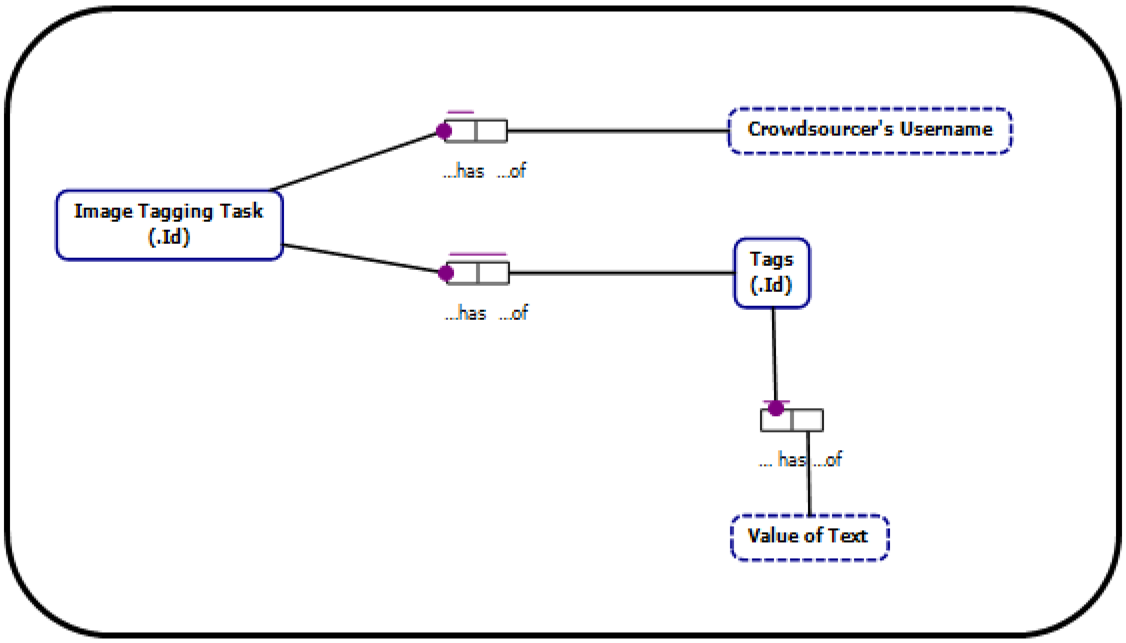


Figure 14: Worker's image tagging task

- Data Collection Task

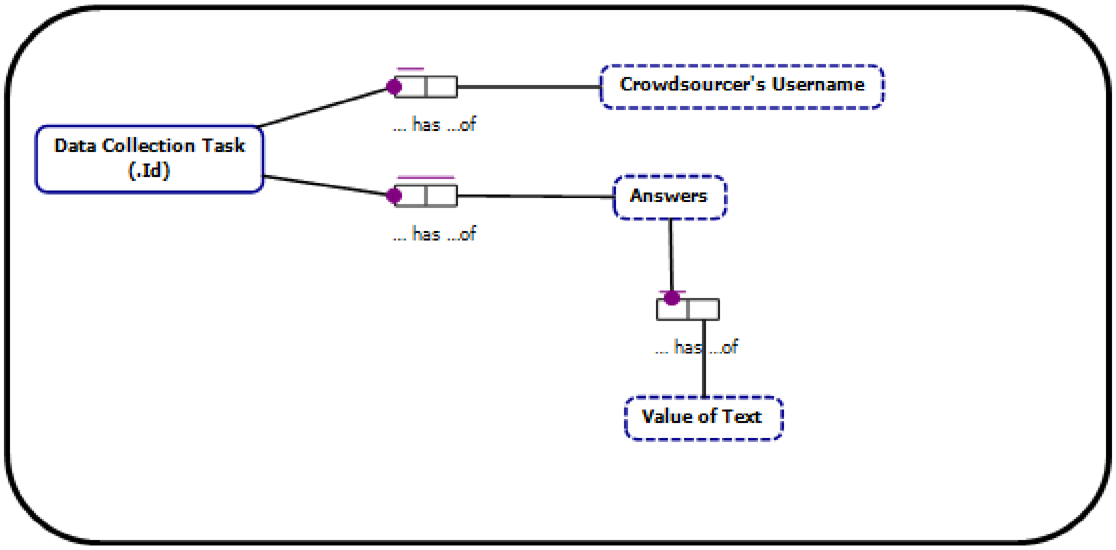


Figure 15: Worker's data collection task

- Voting Task

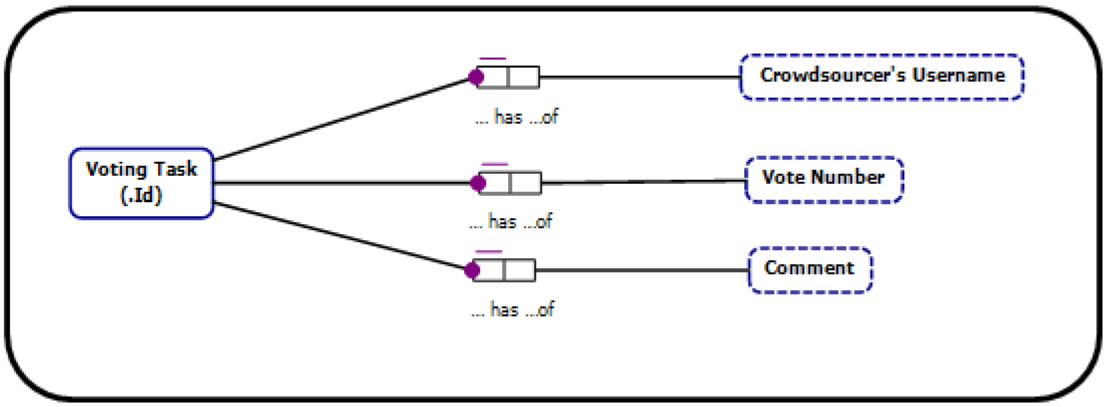


Figure 16: Worker’s voting task

- Translation Task

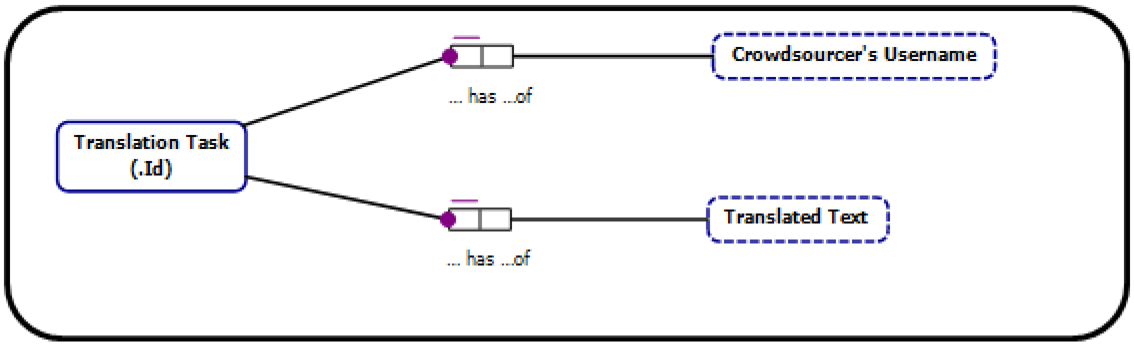


Figure 17: Worker’s translation task properties

- Temperature Task

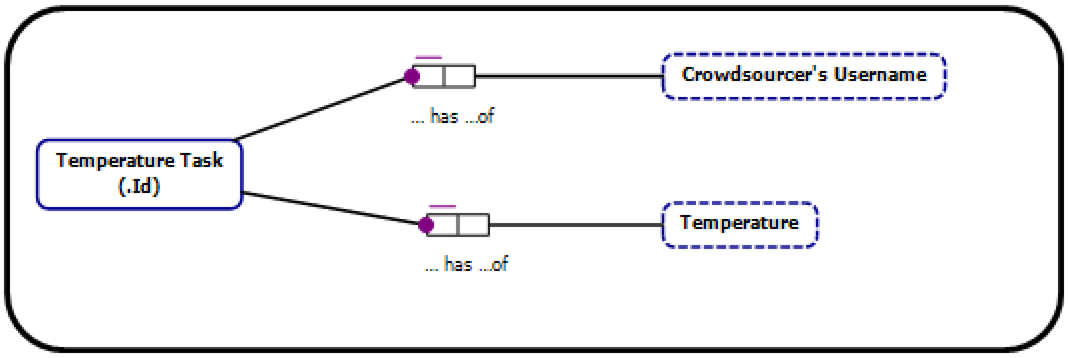


Figure 18: Worker's temperature task properties

4.4.3. View Completed Tasks

In this area, the worker is provided with an overview of the tasks he has already submitted. He can view the task properties, as well as the responses he has given.

4.5. Shared Components

The shared components, on top of which we have implemented the two mobile applications, ensure the following functionality:

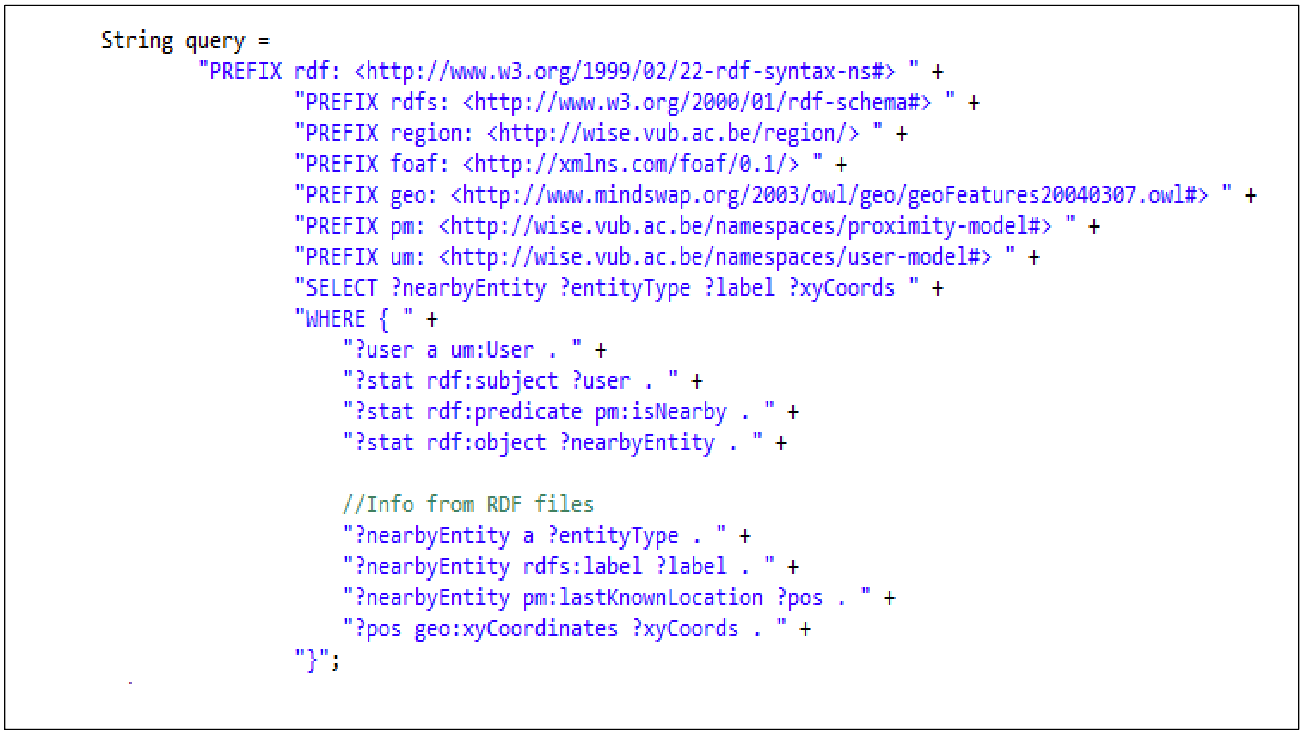
- Interacting with the context repository (SCOUT Framework) by querying the nearby entities of the user and their properties.

- Interacting with the Google Maps API, for selecting locations.

Below we will briefly summarize the functionality of the shared components.

4.5.1. Context Repository

Our system is generic enough to allow any context repository that supports a certain interface (e.g. retrieving nearby entities) to be plugged in. SCOUT is a context repository that supports this interface. In order to query information from SCOUT, we use a SCOUT service that is built on top of the framework, which provides the ability to implement the interface that needs to be supported by the context repository. We use this service in order to detect the entities that are nearby the user compared to his current location. The service includes a SPARQL query to the SCOUT Environment Layer in order to obtain the users' nearby entities. The SPARQL query utilizes the proximity model "isNearby".



```
String query =
        "PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> " +
                "PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#> " +
                "PREFIX region: <http://wise.vub.ac.be/region/> " +
                "PREFIX foaf: <http://xmlns.com/foaf/0.1/> " +
                "PREFIX geo: <http://www.mindswap.org/2003/owl/geo/geoFeatures20040307.owl#> " +
                "PREFIX pm: <http://wise.vub.ac.be/namespaces/proximity-model#> " +
                "PREFIX um: <http://wise.vub.ac.be/namespaces/user-model#> " +
                "SELECT ?nearbyEntity ?entityType ?label ?xyCoords " +
                "WHERE { " +
                    "?user a um:User . " +
                    "?stat rdf:subject ?user . " +
                    "?stat rdf:predicate pm:isNearby . " +
                    "?stat rdf:object ?nearbyEntity . " +

                    //Info from RDF files
                    "?nearbyEntity a ?entityType . " +
                    "?nearbyEntity rdfs:label ?label . " +
                    "?nearbyEntity pm:lastKnownLocation ?pos . " +
                    "?pos geo:xyCoordinates ?xyCoords . " +
                "}";
```

Listing 1: SPARQL – Select nearby entities

After this query, we will retrieve the nearby entities of the user as well as all attributes that are related to them. These attributes are the following.

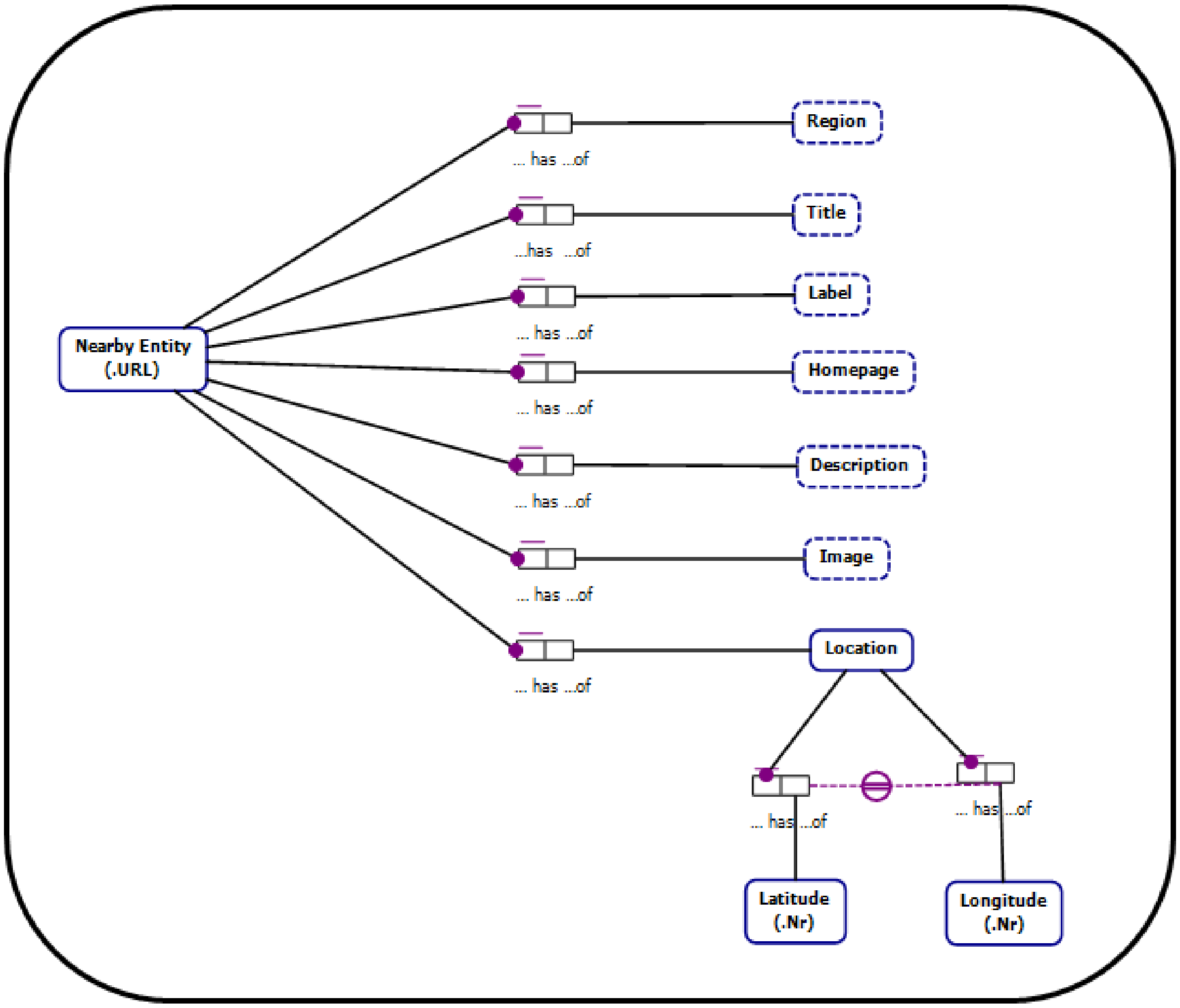


Figure 19: Entities' properties

4.5.2. Google Maps API

The Google Maps API allows the crowdsourcers to enter the location for their task (different from their current location).

The Google Maps API for Android allows the user to explore the word with rich maps provided by Google. We can identify locations with custom markers and embed our map as fragment into our android activity by using an XML snippet.

4.6. Server Interaction

The server enables crowdsourcers to submit their tasks and workers to complete task requests. The server issues authentication credentials for both types of stakeholders, stores details about workers and crowdsourcers, and saves details about entered and completed tasks. The dissemination of the right tasks to the right users requires matching the task criteria to the user's characteristics. These criteria are the following:

- Location: Location is a pair of latitude and longitude. The server will compare the task location with the location of the worker. Two locations are considered nearby if their distance is smaller than 1 km.
- Age: Age is the age of the worker. The age has the values: <18, 18 – 30, 30 – 60 and >60. This age specified by the crowdsourcer has to match the age of the worker.
- Gender: The gender can be a male or a female
- Level of education: The level of education can be highschool, master, doctorate
- Languages Spoken: The languages supported by our system are English, Dutch, French and German.
- Food preferences: The food preferences can be vegetarian or meat fans
- Favourite places: The favourite places can be restaurants, pubs, cafes or others

4.6.1. Interaction with the Crowdsourcer's Layer

There are four main requests that the crowdsourcer layer sends to the server.

- A request to store the registration details of the crowdsourcer.
- A request to validate the login information.
- A request to modify profile information.
- A request for adding newly defined tasks.

In turn, the crowdsourcer's layer receives the following:

- A confirmation message that allows the crowdsourcer to register.
- A confirmation message that allows the crowdsourcers to log in.
- The attributes of his profile and the confirmation message in case of modifications
- Also, the completed tasks of the workers are received by the crowdsourcer layer, allowing the crowdsourcer to view the responses of the tasks he has entered.

4.6.2. Interaction with the Worker's Layer

The worker's layer represents the user interface that is provided to the worker. There are five main requests that the worker sends to the server's layer.

- A request to store the registration details of the worker.
- A request to validate the login information.
- A request to modify profile information.
- A request for querying all the tasks that are relevant to the user, according to his location and profile.
- A request for storing the responses of the completed tasks.

In turn, the worker's layer receives the following:

- A confirmation message that allows the worker to register.
- A confirmation message that allows the worker to log in.
- The attributes of his profile and the confirmation message in case of modifications.
- The tasks that are relevant to the user, according to his location and profile.
- A confirmation message for completing a task response.

4.6.3. Technologies

The server side component uses the PHP scripting language to receive requests and interact with the MySQL database. To connect to the server-side, the mobile apps utilize the HTTP protocol and encode the sent data in JSON format. JSON is a lightweight text-based open standard designed for human readable data interchange.

The client-server architecture is illustrated in the below figure.

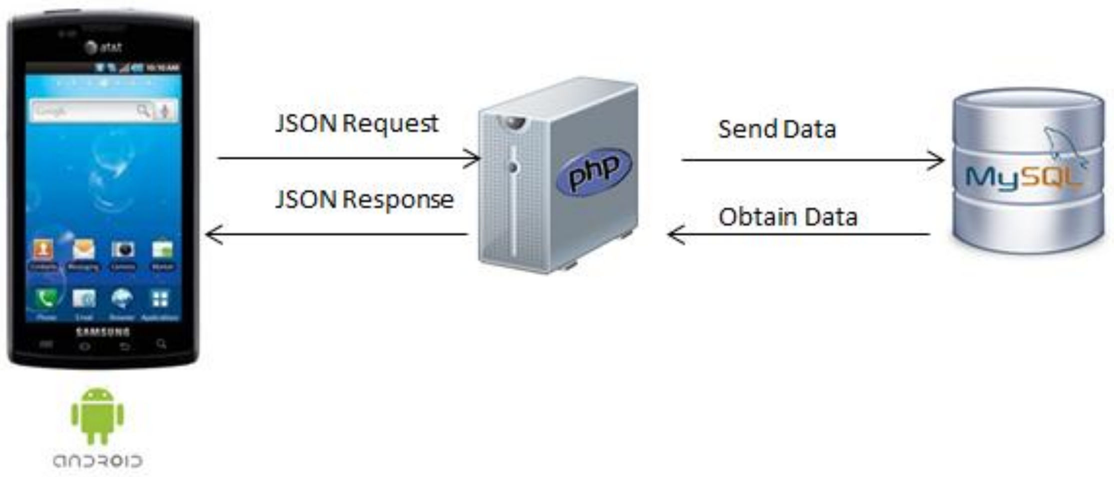


Figure 20: Client-server architecture

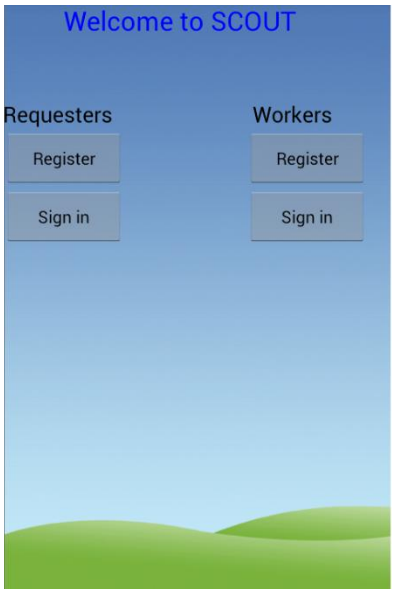
The server-side component was deployed using the XAMPP server, which is one of the most well-known ways to deploy MySQL, Apache and PHP server.

5. Implementation

The goal of our demonstration is to show how our framework can facilitate context aware crowdsourcing. All users will be able to use our system to post context-aware tasks and obtain results as well as work on jobs posted by other crowdsourcers.

5.1. Home Screen

The user interface of our framework is shown in the figure below. This is the first screen displayed to the user, from where he can choose to login or register as a crowdsourcer or as a worker.

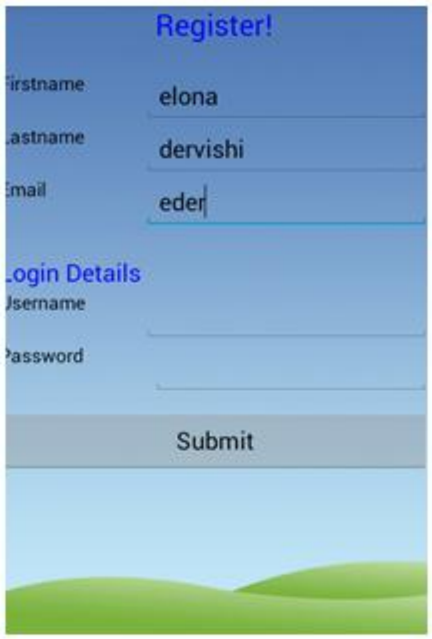


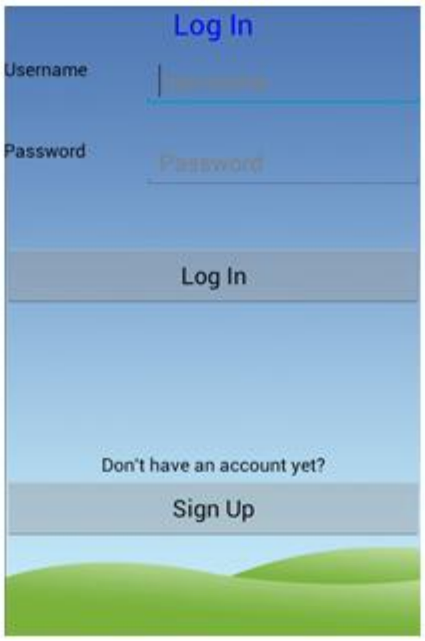
Listing 2: Home screen

5.2. Crowdsourcer's View

5.2.1. Register / Login

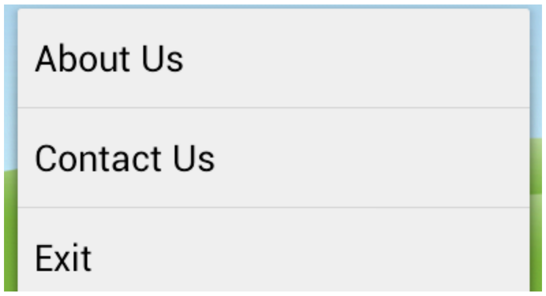
A crowdsourcer can register or login to our system





Listing 3: Crowdsourcer's registration and logging in

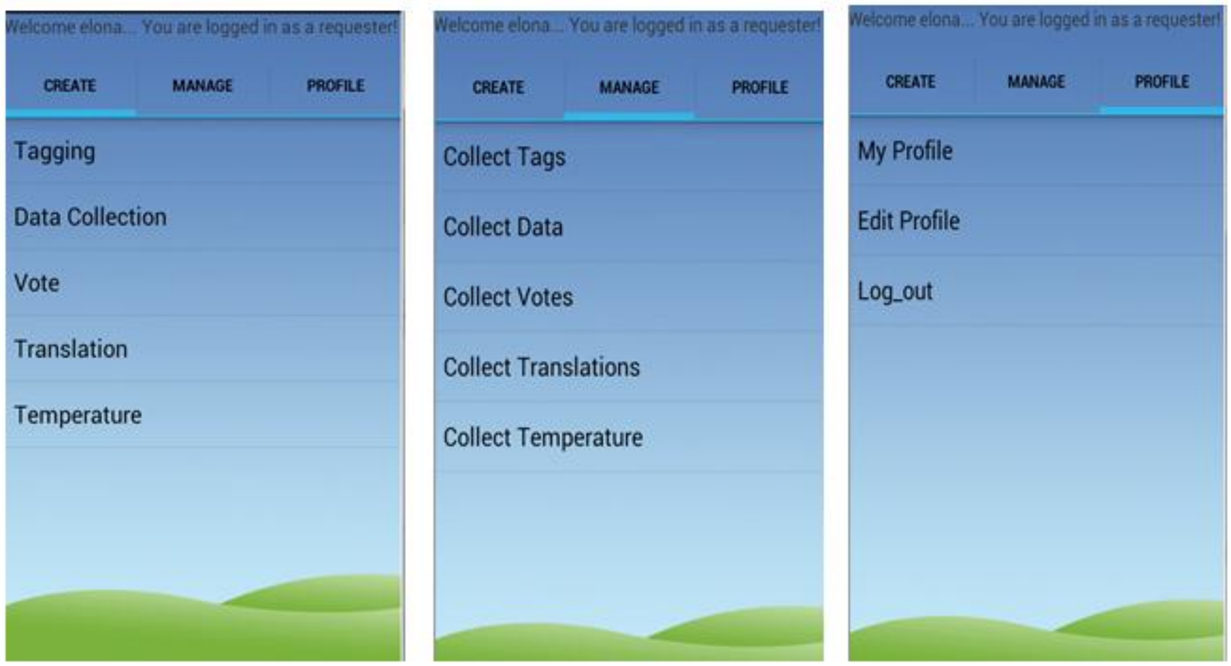
The authentication is taken care by the PHP scripting language and error messages are being displayed to the user in case fields are not filled in, or in case the username/password do not match any existing user. We should also mention that the username is unique for each user. Furthermore, once the user clicks on the menu button of android he will be presented with the options in the figure below. The first button will show a short description of the application; the second will redirect to a contact form where the users can contact the system administrators for comments and suggestions and the third one will simply exit the user from the application.



Listing 4: Simple menu

5.2.2. Tab Menu

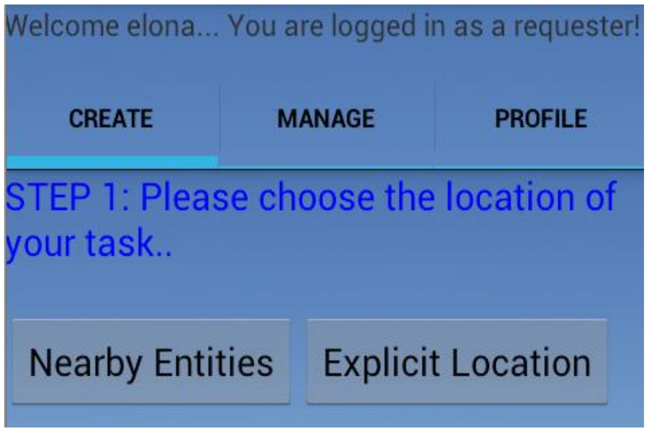
Once the crowdsourcer registers or logs in successfully a welcome message will be shown as well as a tab menu with 3 different functionalities: (i) enter a new task (ii) manage tasks and (iii) manage profile. This tab menu always remains at the user's disposal allowing him to easily navigate to different areas.



Listing 5: Crowdsourcer's tab menu

5.2.3. Create Task

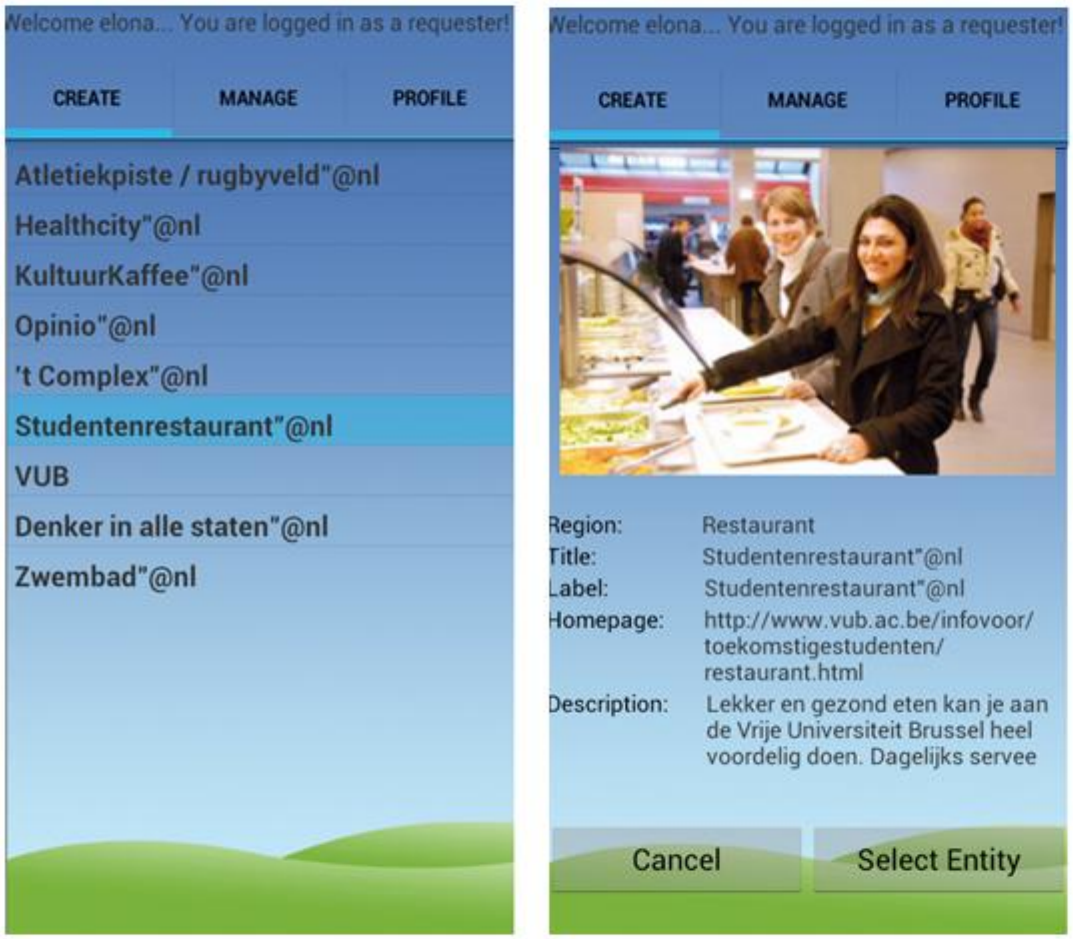
The tasks are separated into five different categories. For simplicity reasons we will demonstrate only one task category at this stage. The task category that could decently illustrate the interaction between the system and the users as well as the use of context-awareness is the "vote" category. For the crowdsourcer the first screen that will be displayed is the following.


Welcome elona... You are logged in as a requester!
CREATE
MANAGE
PROFILE
STEP 1: Please choose the location of your task..
Nearby Entities
Explicit Location

Listing 6: Select task location

5.2.3.1. Task with Nearby Entities

In the scenario of choosing to search for nearby entities, the crowdsourcer will be displayed with a list containing all entities around him. These entities are being queried from the SCOUT framework and they change accordingly to the user's location. Once the user selects one of the entities, the next screen will display all the rest of the details that are related to this entity.



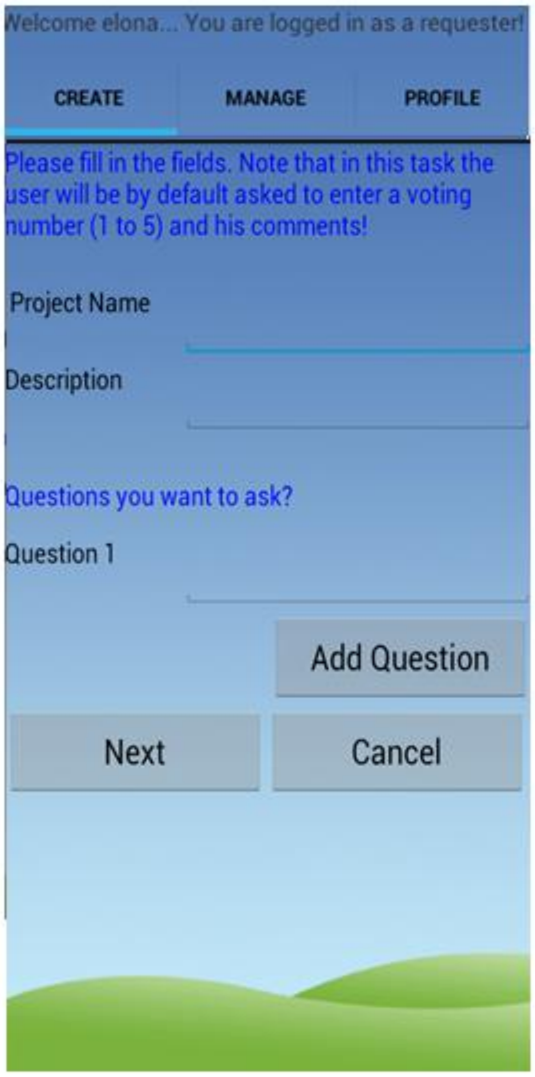
Atletiekpiste / rugbyveld"@nl
Healthcity"@nl
KultuurKaffee"@nl
Opinio"@nl
't Complex"@nl
Studentenrestaurant"@nl
VUB
Denker in alle staten"@nl
Zwembad"@nl

Region:	Restaurant
Title:	Studentenrestaurant"@nl
Label:	Studentenrestaurant"@nl
Homepage:	http://www.vub.ac.be/infovoor/toekomstigestudenten/restaurant.html
Description:	Lekker en gezond eten kan je aan de Vrije Universiteit Brussel heel voordelig doen. Dagelijks servee

Listing 7: Context Entity

After selecting the entity, the next screen illustrates the task information that has to be entered. This information includes the project name, the project description and the questions that will asked to the crowd.

The next screen is about selecting the proper workers to whom this task will be displayed. The user has to scroll down in order to view the whole screen.


Welcome elona... You are logged in as a requester!

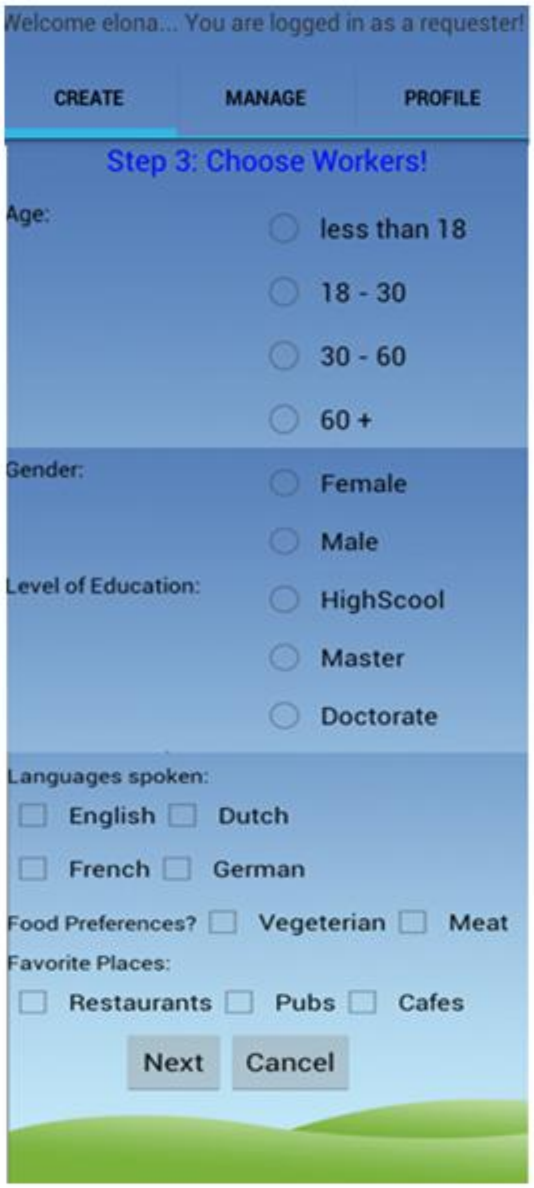
Please fill in the fields. Note that in this task the user will be by default asked to enter a voting number (1 to 5) and his comments!

Project Name

Description

Questions you want to ask?

Question 1


Welcome elona... You are logged in as a requester!

Step 3: Choose Workers!

Age: ○ less than 18 ○ 18 - 30 ○ 30 - 60 ○ 60 +

Gender: ○ Female ○ Male

Level of Education: ○ HighScool ○ Master ○ Doctorate

Languages spoken: ☐ English ☐ Dutch ☐ French ☐ German

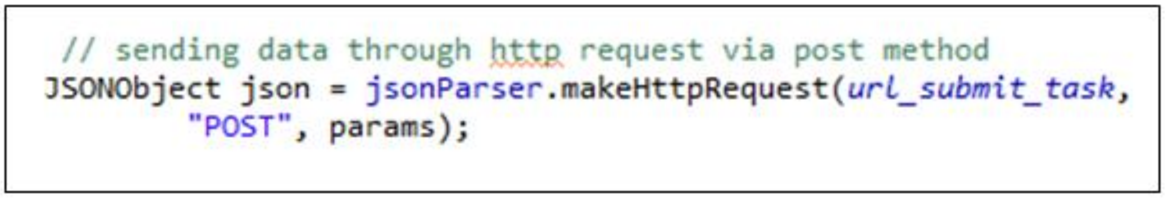
Food Preferences? ☐ Vegeterian ☐ Meat

Favorite Places: ☐ Restaurants ☐ Pubs ☐ Cafes

Listing 8: Selecting the proper crowd

All the above information will be stored in the database allowing the system to present this task to the proper workers. This implementation includes:

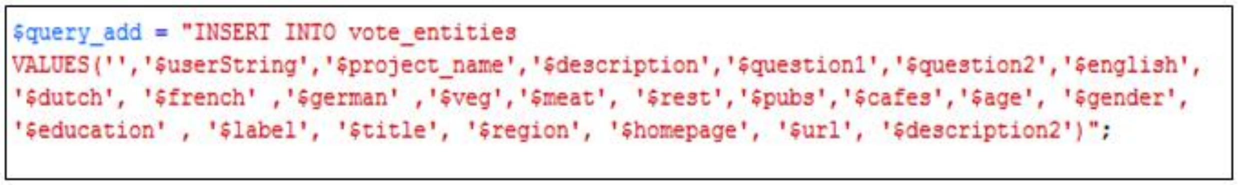
- Error messages in case the crowdsourcer does not fill in one of the fields.
- Sending the data to the PHP server via a JSON request



```
// sending data through http request via post method
JSONObject json = jsonParser.makeHttpRequest(url_submit_task,
        "POST", params);
```

Listing 9: JSON request

- Storing the data in the database by using MySQL queries. Although the user goes through several android screens, the data is being added at once; thus there is a one-time communication with the server.



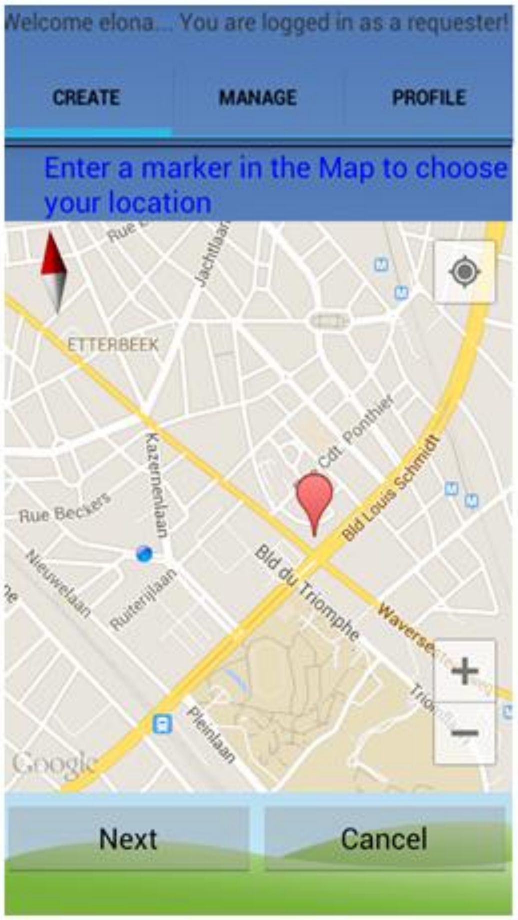
```
$query_add = "INSERT INTO vote_entities
VALUES('','$userString','$project_name','$description','$question1','$question2','$english',
'$dutch', '$french' ,'$german' ,'$veg','$meat', '$rest','$pubs','$cafes','$age', '$gender',
'$education' , '$label', '$title', '$region', '$homepage', '$url', '$description2')";
```

Listing 10: Storing data

- Displaying a confirmation message to the crowdsourcer that his task was successfully created. He will then be redirected to the screen that includes all the task categories.

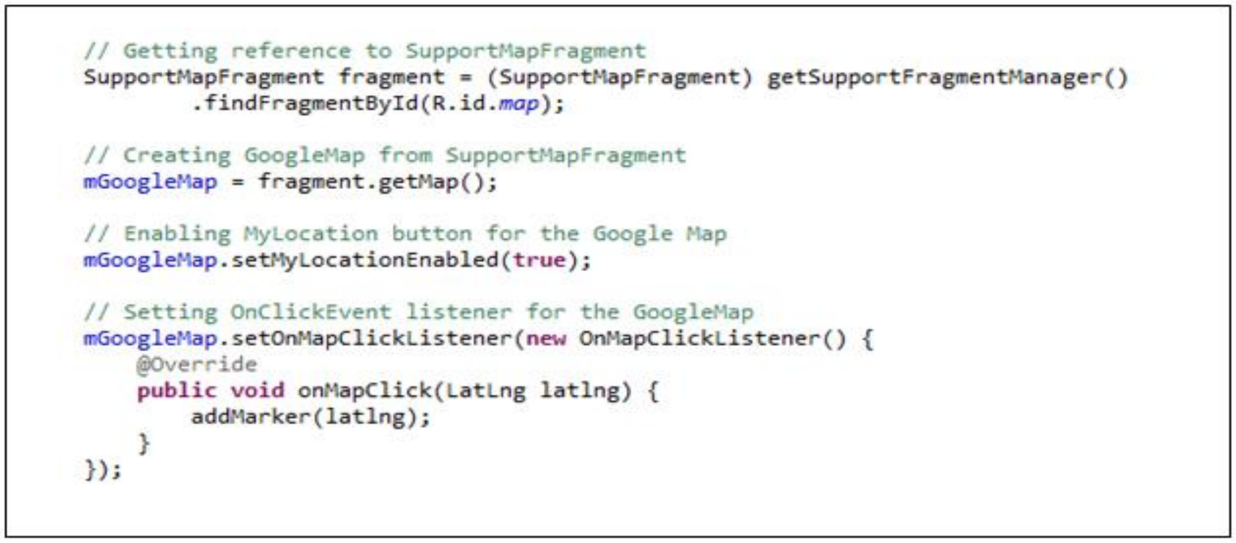
5.2.3.2. Task with Explicit Location

In case the crowdsourcer chooses to create a task in an explicitly selected location, he will be displayed with a map where he can easily mark the location of his task. As you can see from the picture below, the blue marker displays the current location of the crowdsourcer and the red marker which is added by the crowdsourcer represents the location of his task. The location of the task does not necessarily have to be nearby the user.



Listing 11: Google maps location

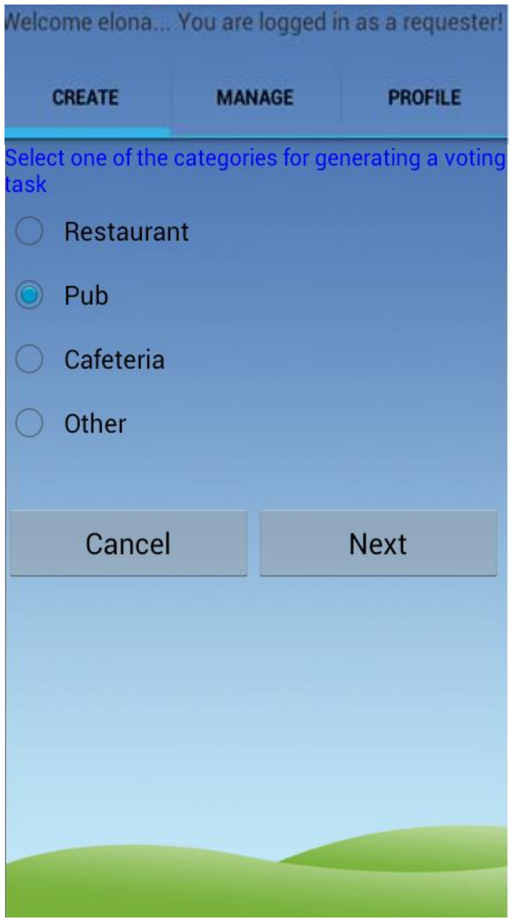
This implementation was made by using the Google Maps API.



```
// Getting reference to SupportMapFragment
SupportMapFragment fragment = (SupportMapFragment) getSupportFragmentManager()
        .findFragmentById(R.id.map);

// Creating GoogleMap from SupportMapFragment
mGoogleMap = fragment.getMap();

// Enabling MyLocation button for the Google Map
mGoogleMap.setMyLocationEnabled(true);

// Setting OnClickEvent listener for the GoogleMap
mGoogleMap.setOnMapClickListener(new OnMapClickListener() {
    @Override
    public void onMapClick(LatLng latlng) {
        addMarker(latlng);
    }
});
```

Listing 12: Google Maps implementation

The next screen is requesting the crowdsourcer to select one category of places or others for which the voting will be about. The options include restaurant, pub, cafeteria or other. This is especially related to the personalization aspect of our system, as the crowdsourcer can choose for example only people whose favourite place is pub. In case the crowdsourcer wishes to receive votes for a restaurant he can also choose the food habits of the users (vegetarians or meat-fans).

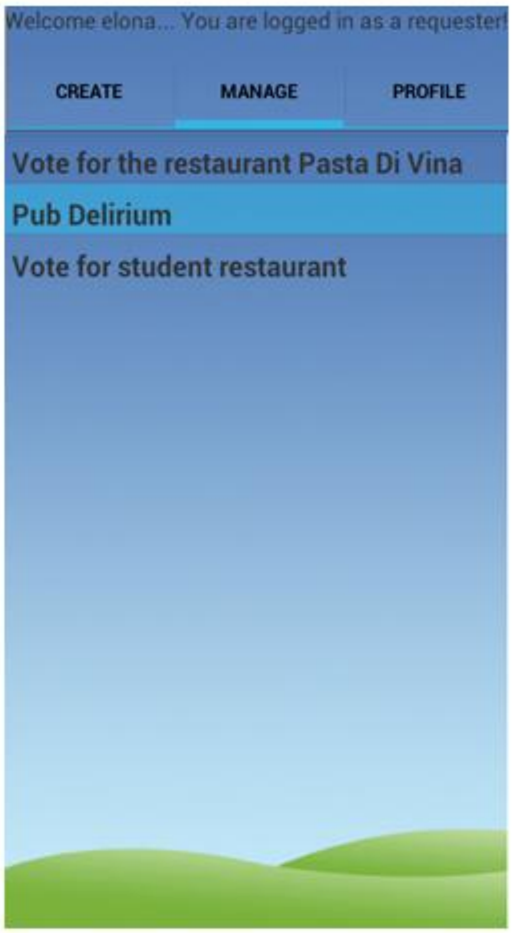

Welcome elona... You are logged in as a requester!
CREATE
MANAGE
PROFILE
Select one of the categories for generating a voting task
Restaurant
Pub
Cafeteria
Other
Cancel
Next

Listing 13: Choose type

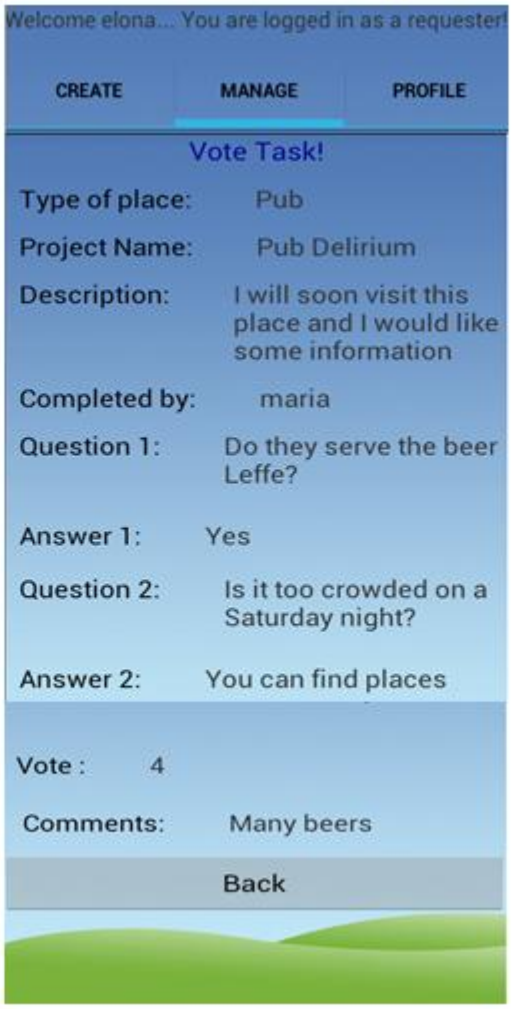
Same as before, the next screens that will be presented to the crowdourcer are the task details and the selection of the proper users (see Listing 8). The data are going to be sent to the PhP server and stored to the MySQL database.

5.2.4. Manage

In this area the crowdsourcer can view the tasks that he has created and the responses of the users.


Welcome elona... You are logged in as a requester!

Vote for the restaurant Pasta Di Vina
Pub Delirium
Vote for student restaurant


Welcome elona... You are logged in as a requester!

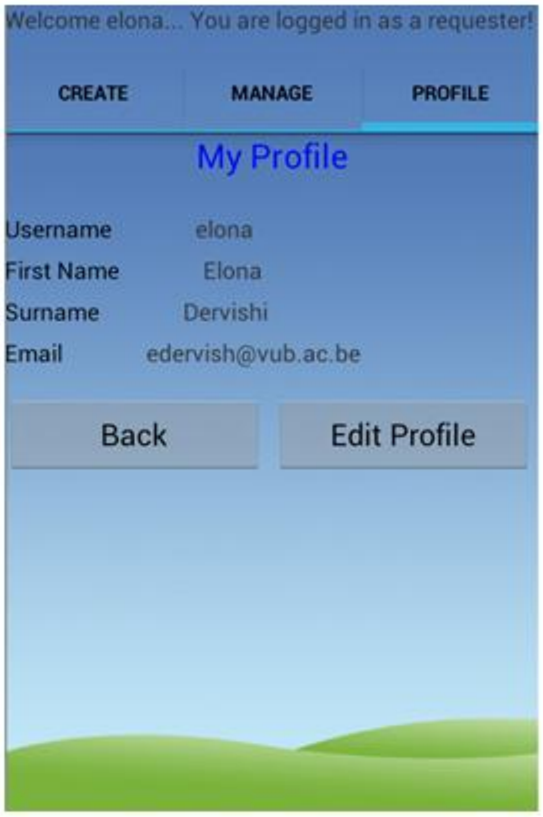
Vote Task!

Type of place:	Pub
Project Name:	Pub Delirium
Description:	I will soon visit this place and I would like some information
Completed by:	maria
Question 1:	Do they serve the beer Leffe?
Answer 1:	Yes
Question 2:	Is it too crowded on a Saturday night?
Answer 2:	You can find places
Vote :	4
Comments:	Many beers

Listing 14: View completed task

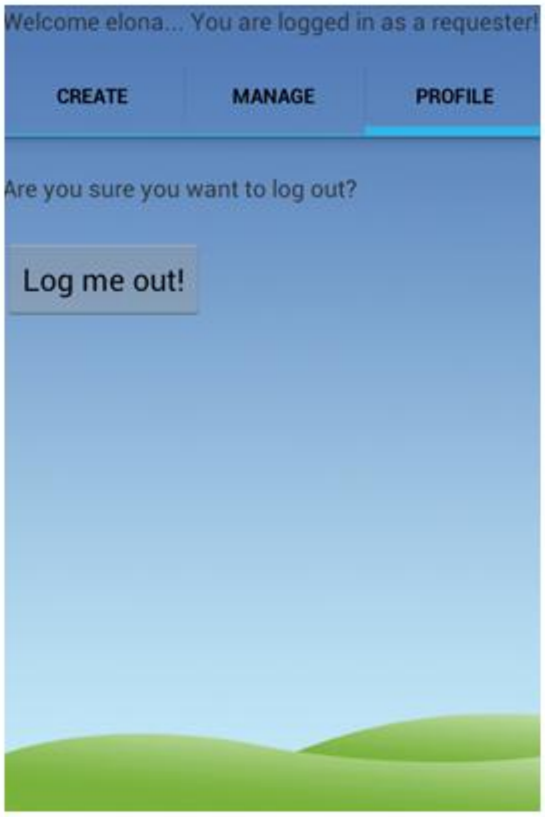
5.2.5. Profile

In the profile tab the user will be given the ability to view his profile and the data he has entered to our system, to edit his profile and log out. The data are being queried from the database and sent to the android client by JSON request. The response from the server is firstly converted to a string message and displayed to the user in the correct format.


Welcome elona... You are logged in as a requester!

My Profile

Username	elona
First Name	Elona
Surname	Dervishi
Email	edervish@vub.ac.be


Welcome elona... You are logged in as a requester!

Are you sure you want to log out?

Listing 15: Manage profile

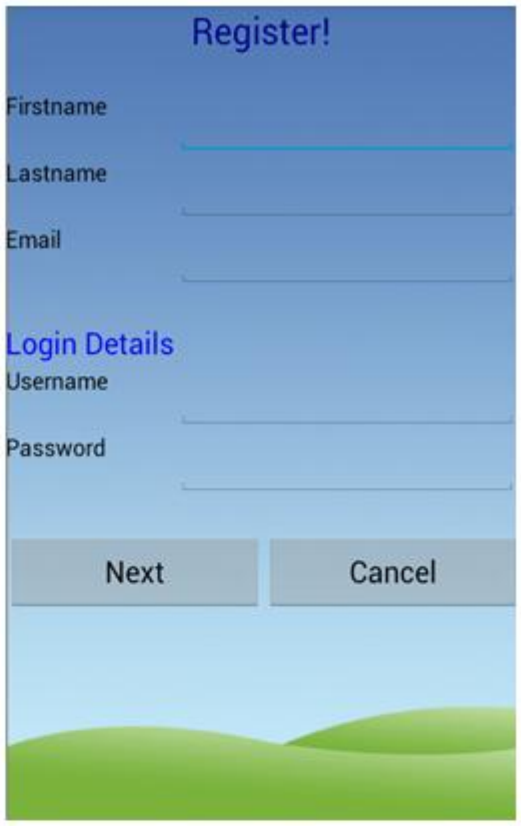
```
JSONArray jArray = new JSONArray(result);
for(int i=0;i<jArray.length();i++){
     JSONObject json_data = jArray.getJSONObject(i);

     display_firstname.setText(json_data.getString("c_firstname"));
     int id = json_data.getInt("id");
     display_lastname.setText(json_data.getString("c_lastname"));
     display_email.setText(json_data.getString("c_email"));
}
```

Listing 16: Server’s JSON response

5.3. Worker's View

5.3.1. Register / Login

Same as the crowdsourcer, the worker can register or log in to our application by filling in their personal data. The login screen remains the same, but the registration form is slightly different as the worker has to enter more data in order to get a personalized crowdsourcing experience. The extra fields have to be filled match exactly with the fields that the crowdsourcer is given in order to choose the workers.



Register!

Firstname

Lastname

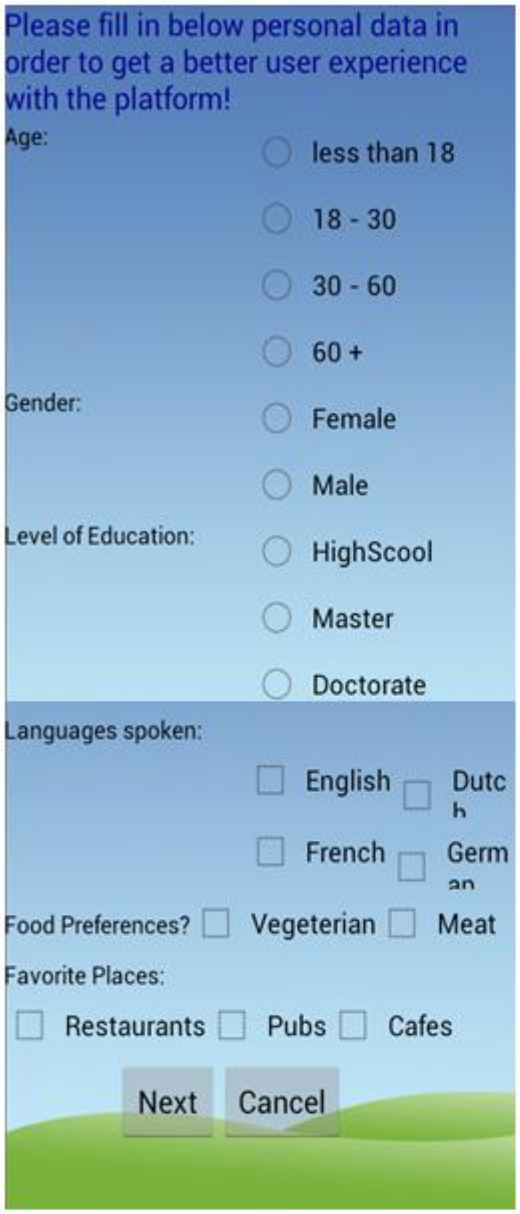
Email

Login Details

Username

Password

Next Cancel



Please fill in below personal data in order to get a better user experience with the platform!

Age: less than 18 / 18 - 30 / 30 - 60 / 60 +

Gender: Female / Male

Level of Education: HighScool / Master / Doctorate

Languages spoken: English / Dutch / French / German

Food Preferences? Vegeterian / Meat

Favorite Places: Restaurants / Pubs / Cafes

Next Cancel

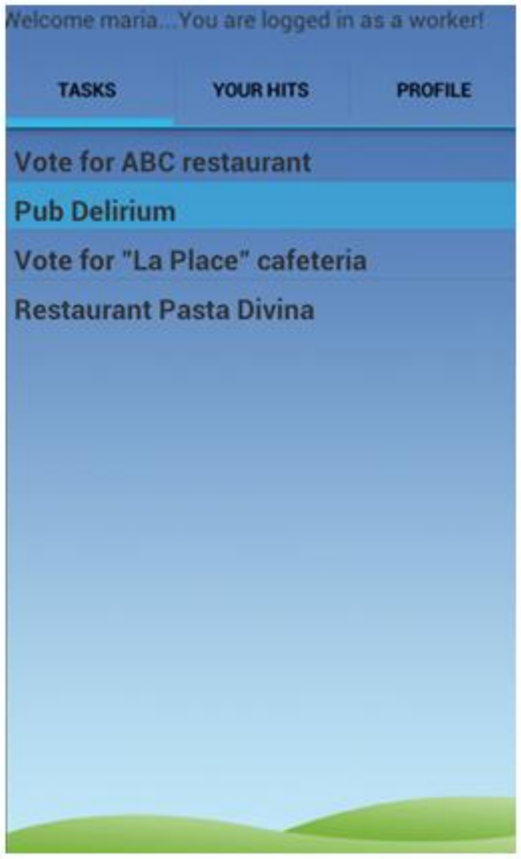
Listing 17: Worker's registration

5.3.2. Tab Menu

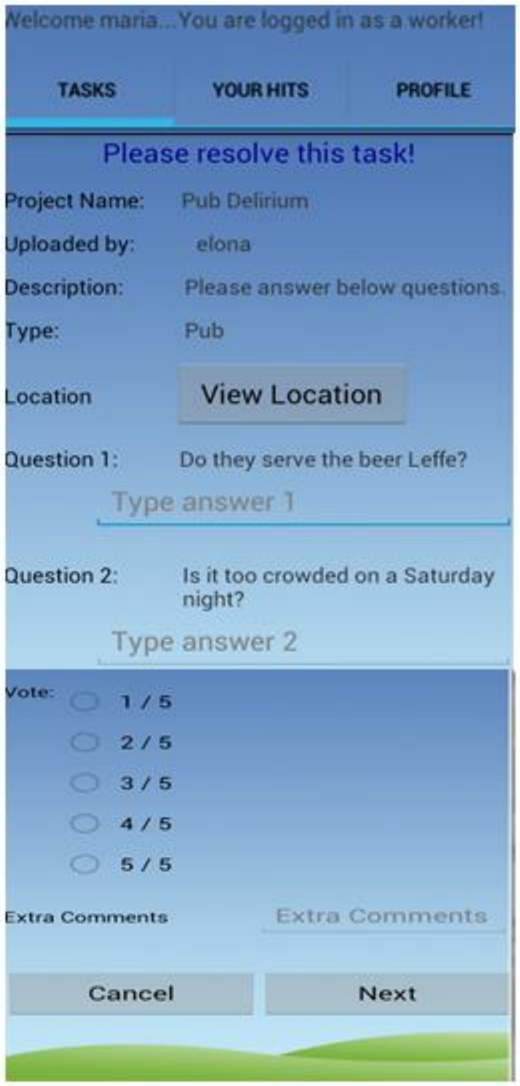
The worker's tab menu is similar with the crowdsourcer's menu. The worker can resolve a task, view the tasks he has completed and manage his profile.

5.3.3. Resolve Task

The system pushes the right tasks to the right users, by comparing the data entered by the crowdsourcer to the ones entered by the worker. In case there are no tasks matching the user, a relevant message will be displayed. Let's assume that a worker selects to resolve the task with project name "Pub Delirium. Firstly he will be displayed with a screen that includes all the information related to that task as added by the crowdsourcer and also the questions that he has to answer. Furthermore, the worker can view the location of the place he has to vote by clicking the button "View Location".



Vote for ABC restaurant
Pub Delirium
Vote for "La Place" cafeteria
Restaurant Pasta Divina



Please resolve this task!

Project Name: Pub Delirium
Uploaded by: elona
Description: Please answer below questions.
Type: Pub
Location View Location
Question 1: Do they serve the beer Leffe?
Type answer 1
Question 2: Is it too crowded on a Saturday night?
Type answer 2
Vote: 1 / 5 2 / 5 3 / 5 4 / 5 5 / 5
Extra Comments Extra Comments
Cancel Next

Listing 18: Worker solving task

5.3.4. Worker’s Hits

Same as with the crowdsourcer, the worker can view the tasks that he has resolved.

5.3.5. Worker’s Profile

The worker can also view or edit his profile as well as the personal data that will be taken into account for delivering tasks.

6. Conclusions and Future Work

In this thesis, we investigated how crowdsourcing can be extended to offer a new personalized and context-aware experience to users. Based on our discussions and research, we proposed an approach for context- and location- based crowdsourcing. We created a framework consisting of a mobile client for the crowdsourcer, a mobile client for the solver, and a web server for storing and disseminating tasks. To the best of our knowledge, our work is the first example of crowdsourced work allocation of *context and location sensitive* tasks to suitable groups of users. In summary, our contributions are as follows: (i) A novel system for mobile crowdsourcing, which pushes relevant tasks to suitable workers based on their *full context* (location, habits, preferences). (ii) A fully mobile and context-aware framework, since not only the worker but also the crowdsourcer can specify tasks in a context-aware way, using a mobile application. (iii) A generic system, as it supports multiple sort of tasks.

As future work, we plan to enhance our application with video and audio features as well as tagging local songs. Furthermore, it might be interesting to provide real-time responses by allowing posting of high priority tasks to the front of the task list. Users will have to follow the priority task list and provide a response within the time specified by the crowdsourcer. Additionally, future work will also explore the possibility of remuneration for each task entered in the system. Tasks with the highest priority will have the highest rewarding amount.

Furthermore, in order to achieve efficient and appropriate task allocation we will take into consideration the worker's availability to work on a task. Recruiting workers has the consequence of potential delays in the resolution time of a task. In a normal case scenario users have to wait several minutes, hours or even days to get their first answers. This can be an important drawback for crowdsourcers who are seeking a quick reply. In order to overcome this challenge, we will investigate the implementation of an approach where workers can be intelligently recruited in advance and based on their availability. This could be implemented in two ways: (i) manually, by allowing workers to specify the time that they are able to receive and solve tasks, (ii) using external libraries, as for example the TurKit Mechanical Turk [11], implemented by Amazon. TurKit primarily achieves low latency by querying workers before they are needed. These two approaches would help crowdsourcers get answers back even faster and reduce task's resolution time.

Lastly, we plan to work on providing a general mechanism for easily defining new, arbitrary tasks, based on Semantic Web technologies. More specifically, by defining ontologies for task types (e.g., sensor data collection) and types of data to be collected (e.g., noise level, votes), as well as task criteria (e.g., location, time), new types of tasks can be defined and plugged-in easily. This will result in making our framework truly generic. At the same time, it is clear that the range of supported tasks will still depend on the capabilities of the mobile devices (e.g.,

[11] http://groups.csail.mit.edu/uid/turkit/

light sensor). We also believe that Semantic Web descriptions of tasks will supply an added value to our framework, for instance data exchange with other systems.

7. Bibliography

[1] Brabham, Daren C. (2013), Crowdsourcing, MIT Press

[2] - Alt, F., Shirazi, A. S., Schmidt, A., Kramer, U., Nawaz, Z. 2010. Location-based crowdsourding – Extending crowdsourcing to the real world. In Proc. NordiCHI2010, 13-22.

[3] Eagle, N. (2009). txteagle : Mobile Crowdsourcing, Internationalization, Design and Global Development, 447-456

[4] Alt, F., Shirazi, A. S., Schmidt, A., Kramer, U., Nawaz, Z. 2010. Location-based crowdsourding – Extending crowdsourcing to the real world. In Proc. NordiCHI2010, 13-22.

[5]: Howe J., The rise of crowdsourcing, Wired 14(6) (2006)

[6] J. Howe. Crowdsourcing: Why the power of the crowd is driving the future of business, Three Rivers Press, 2009.

[7]: Blohm, I.; Fähling, J.; Leimeister, J. M.; Krcmar, H. & Fischer, J. (2010): Accelerating customer integration into innovation processes using Pico-Jobs. In: 22. ISPIM Conference 2010, Bilbao, Spain.

[8]: Brabham, D. C. (2010) Moving the crowd at Threadless: Motivations for participation in a crowdsourcing application, Information, Communication & Society, 13, 8, 1122-1145.

[9] Oomen, Johan and Lora Aroyo. 'Crowdsourcing in the Cultural Heritage Domain: Opportunities and Challenges.' Paper presented at the 5th International Conference on Communities and Technologies, Queensland University of Technology, Brisbane, Australia, 29 June-2 July 2011.

[10] Schilit, B., Theimer, M. Disseminating Active Map Information to Mobile Hosts. IEEE Network, 8(5) (1994) 22-32

[11] Brown, P.J., Bovey, J.D. Chen, X. Context-Aware Applications: From the Laboratory to the Marketplace. IEEE Personal Communications, 4(5) (1997) 58-64

[12] Dey, A.K., Abowd, G.D., Wood, A. CyberDesk: A Framework for Providing Self-Integrating Context-Aware Services. Knowledge-Based Systems, 11 (1999) 3-13

[13] Brown, P.J. The Stick-e Document: a Framework for Creating Context-Aware Applications. Electronic Publishing '96 (1996) 259-272

[14] Hull, R., Neaves, P., Bedford-Roberts, J., Towards Situated Computing, 1st International Symposium on Wearable Computers (1997) 146-153

[15] Dey, K., et al., A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. Human-Computer Interaction Journal 16, 24 (2001), pp. 97-166.

[16] Salber, D., Dey, A.K., Abowd, G.D. Ubiquitous Computing: Defining an HCI Research Agenda for an Emerging Interaction Paradigm. Georgia Tech GVU Technical Report GIT-GVU-98-01 (1998)

[17] Brown, M. Supporting User Mobility. International Federation for Information Processing (1996)

[18] Cooperstock, J., Tanikoshi, K., Beirne, G., Narine, T., Buxton, W. Evolution of a Reactive Environment CHI ’95 (1995) 170-177

[19] Elrod, S., Hall, G., Costanza, R., Dixon, M., des Rivieres, J. Responsive Office Environments. CACM 36(7) (1993) 84-85

[20] Rekimoto, J., Ayatsuka, Y., Hayashi, K. Augment-able Reality: Situated Communication through Physical and Digital Spaces. 2nd International Symposium on Wearable Computers (1998) 68-75

[21] Dey, A. K., et al., Towards a Better Understanding of Context and Context- Awareness. Technical Report 99-22, Georgia Institute of Technology, 1999

[22]: Gupta, A., Thies, W., Cutrell, E., Balakrishnan, R. mClerk: enabling mobile crowdsourcing in developing regions. In Proc. CHI '12, ACM (2012), 1843-1852

[23]: . Narula, P., Gutheim, P., Rolnitzky, D., Kulkarni, A., and Hartmann, B. MobileWorks: A Mobile Crowdsourcing Platform for Workers at the Bottom of the Pyramid. In Proc. HCOMP '11 (2011)

[24]: Bigham, J.P., Jayant, C., Ji, H., Little, G., Miller, A., Miller, R.C., Miller, R., Tatrowicz, A., White, B., White, S., and Yeh, T. VizWiz: Nearly Real-time Answers to Visual Questions. UIST 2010

[25] Yan et al-mCrowd: Yan, T.; Marzilli, M.; Holmes, R.; Ganesan, D.; and Corner, M. 2009. mCrowd: a platform for mobile crowdsourcing. In ACM Conference on Embedded Networked Sensor Systems, SenSys ’09, 347–348. ACM

[26] Alt, F., Shirazi, A. S., Schmidt, A., Kramer, U., Nawaz, Z. 2010. Location-based crowdsourding – Extending crowdsourcing to the real world. In Proc. NordiCHI2010, 13-22

[27] Konomi, S., Thepvilojana, N., Suzuki, R., Pirttikangas, S., Sezaki, K., Tobe, Y. Askus: Amplifying mobile actions. Proc. Pervasive 2009, Springer, (2009), 202-219

[28] A. Tamilin, I. Carreras, E. Ssebaggala, A. Opira, and N. Conci, “Context-aware mobile crowdsourcing,” in Proc. of UbiMi Workshop, Pittsburgh, US, Sept. 2012