POLITECNICO
MILANO 1863

Requirement Analysis and Specification Document

Maccarrone Salvatore, Pellizzer Massimiliano, Prati Andrea

December 23, 2021

Contents

Chapter 1

Introduction

Agriculture plays a pivotal role in India's economy. Climate change continues to be a real and potent threat to the agriculture sector, which will impact everything from productivity to livelihoods across food and farm systems and is predicted to result in a 4%-26% loss in net farm income towards the end of the century. This calls for a revamp of the entire mechanism that brings food from farms to our plates. The COVID-19 pandemic has greatly highlighted the massive disruption caused in food supply chains exposing the vulnerabilities of marginalized communities, small holder farmers and the importance of building resilient food systems. It has become even more important now that we develop and adopt innovative methodologies and technologies that can help bolster countries against food supply shocks and challenges.

1.1 Purpose

Telangana's government wants to design, develop and demonstrate anticipatory governance models for food systems using digital public goods and community-centric approaches to strengthen data-driven policy making in the state. This will require the involvement of multiple stakeholders, from normal citizens to policy makers, farmers, market analysts, agronomists and so on.

In the first place, Telangana wants to partner with IT providers with the aim of acquiring and combining:
Data concerning meteorological short-term and long-term forecasts. Telangana already collects and makes available such data (see https://www.tsdps.telangana.gov.in/aws.jsp).

- Information provided by the farmers about their production (types of products, produced amount per product).
- Information obtained by the water irrigation system concerning the amount of water used by each farmer.
- Information obtained by sensors deployed on the territory and measuring the humidity of soil.
- Information obtained by the governmental agronomists who periodically visit the farms in their areas.

Acquiring and combining such data, DREAM will support the work of three types of actors: policy makers, farmers, and agronomists. In the following we describe the goals summed up.

1.1.1 Goals

G1 - Data-Driven approach

DREAM will be able to fetch data from different databases. Then, DREAM will be able to collect, analyze and perform inference on this data in order to extract new knowledge that will be at the service of different *DREAM*'s users.

G2 - Help policy makers to implement better policies which will make Telangana's agriculture better

DREAM will allow policy makers to see real time data regarding Telangana and Telangana's farms. In particular they will be able to see which farms are performing well and which farms are performing poorly. Moreover, they will be able to understand which steering initiatives are working and which not.

G3 - Help farmers to become weather resilient and increase their production

DREAM will allow farmers to see statistics regarding their farms. Moreover, *DREAM* will provide to them personalized suggestions based on the data regarding their farm and on the data regarding farms that are doing particularly well in similar environments.

G4 - Facilitate and make systematic the communication between farmers and agronomists

DREAM will make communications between agronomists and farmers easy and stable. In fact, *DREAM* will allow farmers to chat with agronomists in just a few clicks. Moreover they will be able to request agronomists to visit them when they are facing complex problems. Agronomists, instead, will be able to access real time data regarding farms they are responsible for, and therefore they will be able to see which farms are performing well and which farms are performing poorly. Based on this knowledge, agronomists will be able to decide how much and when to visit each farm, and to send periodical reports to policy makers regarding how the farms (they are responsible for) are doing.

G5 - Create a network of farmers

Through *DREAM*, farmers will be able to communicate between each other in order to exchange best practices and advice.

1.2 Scope

The problem that Telangana is facing concerns the world of agriculture, and it has a very important weight on the lives of many people who, due to various impediments mentioned in the previous section, need a platform that can support them and allow those who create support initiatives to do the work in the most scientific way possible, keeping track of how is all going on without losing information.

To reach this goal, called "data-driven approach" it was necessary to think at an entire platform. In particular, this platform can be seen as divided into three categories:

- Data: users must be able to easily interact with the data to which they have the right to access. They are offered different views and aggregations with respect to this data. The available data can include weather forecasts, the state and composition of the land of an area of Telangana, the water irrigation system (the amount of water used), and the initiatives taken by other farmers who have distinguished themselves for being

productive considering the conditions in which they were found (positive and negative deviance). Other types of data are described in the remainder of the document.

- Community: users are able to communicate via two communication channels within the system, one is the Forum and the other is the Chat System that will allow Farmers and Agronomists to talk to each other to exchange advice and clarifications.
- Management: users can organize physical visits between Agronomists and Farmers directly in the app.

The processing of data and their analysis was not the focus for the drafting of this document. In fact, the first goal was to think about how to develop the platform as a whole. The choice on which criteria to calculate the Farmer's performance (i.e. their deviance, positive or negative) will be made thanks to the collaboration with expert agronomists who will provide all the knowledge to be able to create and study intelligent models using special tools such as, TensorFlow, a framework for processing data.
Moreover, the system will interface itself with different databases maintained and managed by third parties. How the data are inserted or managed in those databases is not one of the concern of the system described in this document.

1.2.1 Phenomena

World Phenomena

$W01$ The farmer weighs the crop

$W02$ Physical event that the farmer considers important to report happens

$W03$ The farmer faces a problem concerning their farm

$W04$ The agronomist misses an appointment

$W05$ The agronomist goes to the farmer

$W06$ The agronomist takes measures

$W07$ The agronomist assists the farmer while he is in visit to their farm

$W08$ The farmer open a chat

Shared Phenomena

$S01$ Registration of the user (World Controlled)

$S02$ Login of the user (World Controlled)

$S03$ The user queries the system (World Controlled)

$S04$ The system shows data to the user (Machine Controlled)

$S05$ The user queries the system to retrieve relevant data (World Controlled)

$S06$ The system shows agronomists the list of farmers they have to manage (Machine Controlled)

$S07$ The farmer inserts data concerning their production (World Controlled)

$S08$ The system confirms the success of an operation to the user (Machine Controlled)

$S09$ The farmer open a ticket in order to have a chat with an agronomist (World Controlled)

$S10$ The system shows an agronomist the list of their currently open chats (Machine Controlled)

$S11$ The system shows agronomists the list of chat requests to be managed (Machine Controlled)

$S12$ The agronomist accepts a chat request (World Controlled)

$S13$ The agronomist closes a chat (World Controlled)

$S14$ The farmer interacts with the system's forum (World Controlled)

$S15$ The agronomist schedules an appointment in order to visit a farm (World Controlled)

$S16$ The agronomist cancels an appointment from the schedule (World Controlled)

$S17$ The agronomist confirms the scheduled appointment done during the day (World Controlled)

$S18$ The agronomist inserts a report and other data regarding the appointment done during the day (World Controlled)

$S20$ The agronomist get notified

$S21$ The farmer get notified

Machine Phenomena

$M01$ The system periodically fetches data from its sources

$M02$ The system aggregates and filters data

$M03$ The system validates data and stores it

$M04$ The system updates the agronomist's schedule

$M05$ The system check the references of the agronomist

$M06$ The system checks credentials of the user

$M07$ The system generates the list of farmer associated to each agronomist

$M08$ The system orders the list of farmers associated to each agronomist by priority

$M09$ The system keeps updated the priority of farmers inside their list

$M10$ The system removes a chat request from the list and adds a new entry in the list of open chats managed by an agronomists

$M11$ The system removes the chat from the list of open chats of the agronomist that taken it into account

$M12$ The system synchronizes lists shared between agronomists working on the same area

$M13$ The system keeps track of the appointment missed by agronomists

$M14$ The system generates personalised suggestions based on historical data to be delivered to the farmer

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

Agronomist's schedule → The schedule of an agronomist is a list of planned activities or things to be done showing the times or dates when they are intended to happen or be done. In particular, in this case, the activities are the planned visits to the various farms.

Area → The word "area" indicates a disjoint partition of Telangana. In fact, Telengana is partitioned in several disjoint areas, and to each area are associated both farmers and agronomists, in such a way that:

- Each area can have one or more farmers and one or more agronomists
- Each farmer is associated to one and only one area
- Each agronomist is associated to one and only one area

Chat → A chat is a virtual space that, after it has opened, allows one agronomist and one farmer to discuss with each other.

Data Summary → A data summary is an aggregation of data fetched by various databases, which gives a certain data view of a subject (e.g. a farm or an area).

Data View → A data view is a selection of data fetched by the various databases, regarding a specific aspect (e.g. humidity of soil, water irrigation data) of the subject (e.g. a farm or an area of Telangana).

Priority → The term "priority", when used referring to the list of farms, indicates that the list of farms is sorted taking into account:

- The number of times the farm has already been visited during the year
- How well or bad the farm is performing
- Possible requests of help done by the farmer

These metrics define what is referred to as priority.

Relevant data → The term "relevant data" indicates data regarding:

- Data concerning meteorological short-term and long-term forecasts
- Information provided by the farmers about their production
- Information obtained by the water irrigation system concerning the amount of water used by each farmer
- Information obtained by sensors deployed on the territory and measuring the humidity of soil
- Information obtained by the governmental agronomists who periodically visit the farms in their areas

Report → When an agronomist visits a farm they must write (and insert into the system) a summary of what they have suggested to the farmer, which problems they have detected, and everything they think could be relevant in order to understand why the farmer is performing well or why the farmer is performing poorly. This summary is indicated by the word "report". The purpose of the report is to keep track of the history of every farm.

Ticket (or Chat Request) → Whenever a farmer wants to have a chat with an agronomist they have to open a ticket. A ticket, thus, is a request to have a chat.

Open a ticket → Open a ticket indicates the action, performed by the farmer, of asking an agronomist to have a chat.

Managed ticket → A managed ticket is a chat request accepted by an agronomist.

Not manged ticket → A ticket that is not managed is a chat request not accepted by any agronomist yet.

Users → The users of the system are: farmers, agronomists and policy makers. Whenever the word "users" is used, it indicates both agronomists, farmers and policy makers.

Visit → In order to help farmers, agronomists can reach the farm of the farmer physically. A visit, therefore, is the action, performed by an agronomist, of physically reaching a farmer's farm.

Booking a visit → To book a visit means reaching an agreement between a farmer and an agronomist on when the agronomist should visit the farm.

Booked visit → A booked visit is a visit that has been scheduled by the agronomist in accordance with the farmer

1.3.2 Acronyms

API → Application Programming Interface, it indicates on demand procedure which supply a specific task.

BPMN → Business Process Model and Notation, it is a graphical representation for specifying business processes in a business process model.

DBMS → Data Base Management System, it is an interface between the end user and the database, simultaneously managing the data, the database engine, and the database schema in order to facilitate the organization and manipulation of data.

DREAM → Data-Drive Predictive Farming in Telengana, is the name software to be developed, described in this document.

1.4 Revision History

- December 23, 2021: version 1.0 (first release)

1.5 Reference Documents

Reference book → Hans Van Vilet - Software Engineering: Principles and Practice

Alloy official documentation → https://alloytools.org/documentation.html

UML official specification → https://www.omg.org/spec/UML/

BPMN official specification → https://www.omg.org/spec/BPMN/2.0/

Paper → Jackson and Zave: the world and the machine

Github → https://github.com/UNDP-India/Data4Policy
(All the reference written on this repository are reference that we considered as well)

1.6 Document Structure

- **Section 1: Introduction**
  This section offers a brief overview of the problem, required functionalities and the reference used in order to write this document.
  It also contains the list of definitions, acronyms and abbreviations that could be found in this document.

- **Section 2: Overall Description**
  This section offers the overall organization of the system, the possible scenarios that this application covers and the description of the most important functions that needs to be present in the application.
  Here is also possible to find the domain assumptions under which the system works. The actors who use the application are described as well, this way it's easier to identify the type of user the app is designed for.

- **Section 3: Specific Requirements**
  This section is the most technical, there is a more precise description of each single important use case. It also explains, through the mapping *goal-assumptions-requirements*, how each functionalities is necessary to reach each goal requested by the client.
  It's also possible to understand the hardware and constraints and the attribute needed in the system (reliability, availability, security and so on).

- **Section 4: Formal Analysis through Alloy**
  In section number 4 of this document a formal description of the system is found. The Alloy language is used to reach the description (it has been referenced in the section 1) and each simulation done is commented.

- **Section 5: Effort spent**
  In the last section of this document is presented the number of hours spent per each person of the group. We decided to include also a "Thought process" section, in order to describe the steps we went trough in order to write this document and why we focused our energies into some details instead of others.

Chapter 2

Overall Description

2.1 Product Perspective

The following section contains the UML class diagram of the application and a list of meaningful scenarios where our system can be used.
Notice that a further discussion on the implementation details will be covered in the design document.
Furthermore, in the list of meaningful scenarios, we highlighted the correlation with the associated use cases, that are thoroughly described in section 3.3.

2.1.1 Class diagram

Here we include the class diagram, in order to show the structure of the system.
We structured the system with three type of users, as requested by the assignment, they are all an extension of the class **User**, which is the one that can interact with the *Data layer*, thus every user has the possibility to access at some data, the type of access depends on the type of instance the user is.
The so called *Data layer* has a general handler called **Data handler** that's the one that include a **Data viewer**, a **Data collector** and list of **Data**, here a short description of these classes:

- **Data viewer**: it's the class which has the responsibility of providing all the methods to proper visualize the data. Here there is the access with the Machine Learning component, in order to see this it's better to look at the Design Document.
- **Data collector**: it's the class that allow to collect data and make all the processing needed to store them properly into the DB.
- **Data**: it's the class used to consider the Data entity, thus every data which comes from or goes to the DB it's a **Data** object.

The **Data Handler** interact also with the external persistent storage, which is better specified in the Design Document, and the **Community Handler**, which is the class that has the responsibility of managing the **Chat** and the **Forum**, in order to store info about the *chats* and the forum *threads*.

Each **Farmer** and each **Agronomist** has a **Calendar** and a **Community Handler**. The big difference we can notice between the two type of **User**, the **Agronomist** and the **Farmer**, is that the **Farmer** is composed of a **Farm** and each **Farm** is contained in just one **Area** and an **Area** can contains many *farms*, while an **Agronomist** is bound to just

one **Area** while each **Area** can have more Agronomist bound to itself.
Notice that the **Area** class is an extension of a **GeoSpatialPosition**.

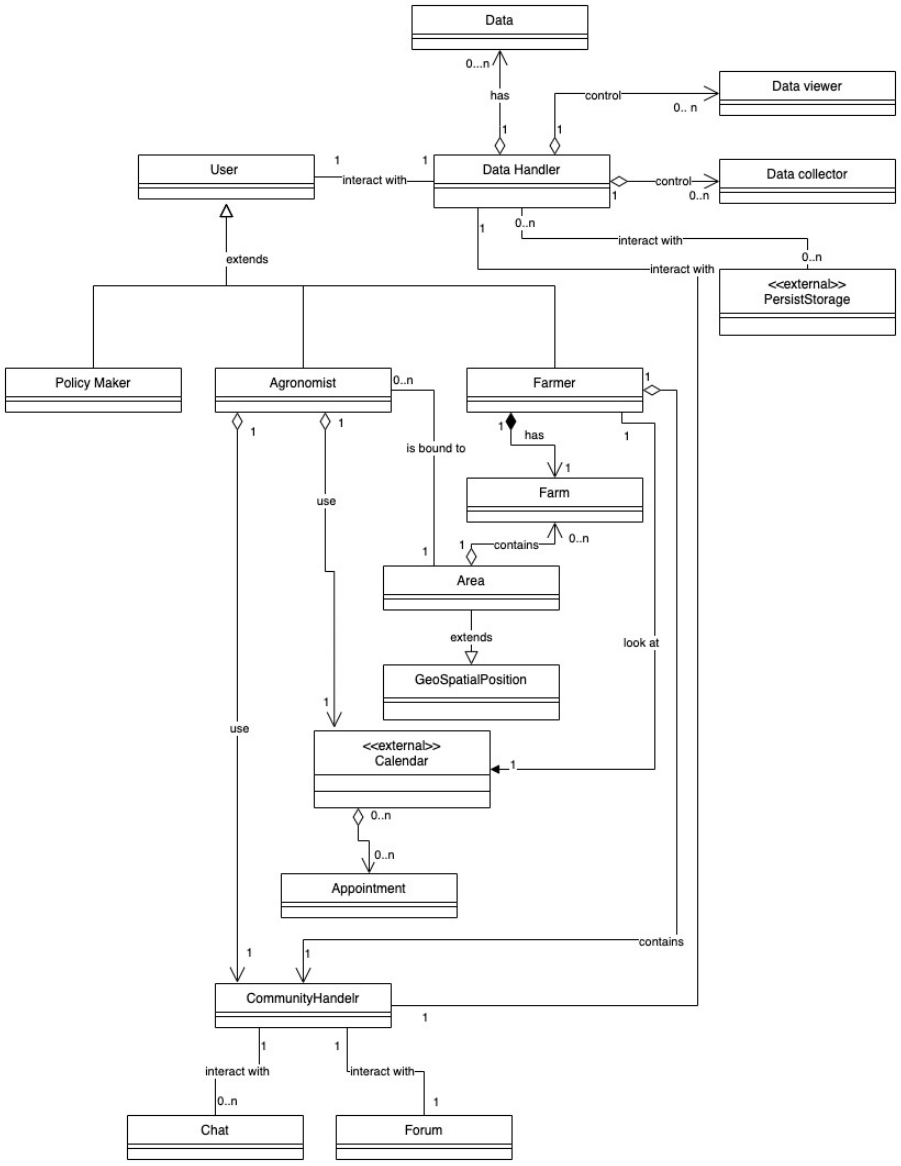

Figure 2.1: Class Diagram

2.1.2 Scenario 1 (Use case: 1.1, 2.1, 3.1)

Adahy is a Farmer that lives in telangana, he has some problems with his farm because of India's climate and the little amount of water available, as do a lot of his colleagues. Adahy heard about a new application that proposes to help Farmers with the decision making process using new informatic technologies and the help of professional agronomists.
Adahy picks up his phone, goes on his app store and downloads the DREAM application. He clicks on "Sign up" and the sign up interface shows up, he inserts the email and chooses a password. He confirms his email by clicking on the link on his personal inbox email. The system shows him a list of fields to be filled with Adahy's role, after he expresses his role

as farmer, the system opens a form to insert his Farm information fiel, so Adahy does so. Another confirmation mail is received by Adahy and the sign up is completed.

Please note

The dual scenario for the agronomist's registration has not been written in order to avoid redundancy. It would have been the same as the Farmer's, the only thing would have changed is the fact that the agronomist does not insert data about the farm, instead, they insert data about which area they make themself available to be assigned to. So, scenario 1 can also map the use case 2.1. Also the Policy Maker's registration use case, the number 3.1, can be mapped into this scenario and, also here, there would be not the insertion of farm's data, but the insertion of the Policy Maker's personal data only.

2.1.3 Scenario 2 (Use case: 1.2, 1.3, 1.4, 1.6)

Ravi is a Farmer that is already registered on the DREAM application. He is facing some problems with his plantation due to the drought, so decide to open a ticket so that he can chat with an Agronomist. Within two hours Yamir, one of the agronomists, accepts the chat request and starts the chat with Ravi. After a few days of chatting, Ravi decides it's better to ask the agronomist to come to Ravi's farm for a visit and write to Yamir about this. The agronomist advises Ravi to use the proper button to request for a visit while the agronomist sends him a proposal to close the chat that Ravi accepts because the chat is no longer needed. Ravi moves to the home screen where clicks on the "Request for a Visit" button, he fills up all the fields needed describing his problem and submit the request. After a few days Ravi receives a date proposal for the visit he requested and decides to accept it, after accepting it the confirmation frame is shown.

2.1.4 Scenario 3 (Use case: 1.5)

Hemang is a farmer registered into the DREAM system, he has a scheduled visit for tomorrow. Unfortunately, Hemang is not at home tomorrow because he has to work at one of his colleague's farm, so he goes to the scheduled visit screen and clicks on the "Reject" button related to the visit of tomorrow. By doing so, the agronomist receives a notification about the canceled appointment.

2.1.5 Scenario 4 (Use case: 1.7, 1.8, 2.6, 2.7)

Ganaraj wants to share his thoughts about the new seed he wants to plant, that's a perfect topic for the DREAM forum. Ganaraj clicks on the forum button, then on "open a thread", decides the title and the body of the post and then publishes it, receiving within 6 hours lots of comments from some colleagues from other areas. Satisfied about the participation on his new thread, Ganaraj decides to look for other interesting topics on the Form and he finds a thread about a machine that he used, so he opens the thread, clicks on "comment" and writes about what was his experience, publishing the comment.

Please note

As also in *Scenario 1*, here we map also scenario 2.6 and 2.7 which are the agronomist's ones, because the scenario would have been very similar to this one.

2.1.6 Scenario 5 (Use case: 1.9)

Dinpal is a farmer registered into the DREAM system, he is happy about the fact he signed up to it, because the agronomist of his area helped him during the production process and now he is proud about the results. In order to keep track of his results, so that he can

remember how he achieved this goal of production, Dinpal opens the DREAM application and clicks on " production data " button. A list of fields shows up on the interface, Dinple fills them up and clicks on "confirm". Now the data about Dinpal's production are stored.

2.1.7 Scenario 6 (Use case: 0.2)

Esh is a farmer already registered in the DREAM application, he is curious to know which crop is planted in land like his. So, he clicks on "visualize data", the system shows a dashboard with data about Esh's farm, like the soil composition, meteo forecasts and agronomist suggestions and so on. Esh sees that there is also the possibility to filter and fetch other types of data and decides to look for types of crop that have been planted in soil like his. The system shows him an interface where data are visualized.

2.1.8 Scenario 7 (Use case: 2.2, 2.3, 2.4)

Yamir is an agronomist already registered in DREAM. He receives a notification about a new ticket opened into the system, so he opens the "chat interface" where there are the pending chats that are visible to all the agronomists of the same area. He decides to take charge of the new chat request, so he clicks on it, and then "process request". After some days of chatting with Ravi, who is the Farmer who opened the ticket, it is proposed to Yamir to go to visit Ravi's farm to solve the problem, so Yamir invite Ravi to go on his interface and look for the proper button to request for a visit and in the meanwhile Yamir sends a proposal to close the chat. Ravi does as Yamir said, he closes the chat. Yamir goes on the screen where the farms assigned to his area are listed. Yamir notices that Ravi's farm went up in rank, due to the chat request which has just been done, he clicks on it and chooses a date to propose and send the proposal. After a while Yamir receives the confirmation to visit Ravi's farm on the date Yamir proposed.

2.1.9 Scenario 8 (Use case: 2.5)

Chandra is an agronomist registered in DREAM. It's noon when he finishes visiting Dharesh's farm. He is happy about the visit because Dharesh was having doubts about when to harvest but now they are solved. Chandra opens DREAM, goes in the schedule screen, where there is the list of the scheduled visits of the day, and clicks on "Done" where Dharesh's farm is written. Now compile the report of the visit, skipping the measures screen, and clicks on "Confirm". Now the report is sent and it's all done.
On tomorrow Chandra has Gopan's farm in schedule, but he thinks that's not more necessary because Gopan called him on the phone and they already solved Gopan's problem. So, Chandra goes on tomorrow's schedule and clicks on "Cancel", by doing so the appointment for tomorrow has been deleted and a notification has been sent to Gopan.

2.1.10 Scenario 9 (Use case: 3.2)

Harish, a Policy Maker registered in DREAM, wants to control the effect of DREAM on Telangana, he opens the application and goes to the performance screen, the system shows him a rank of farms ordered by performance score. Harish is very happy about this system because it simplifies his work a lot and allows him to see all the history of each farm. Therefore, now Harish can analyze the history of the data of the first farm of the rank in order to understand which kind of steering initiatives the government can propose.

2.2 Product Functions

2.2.1 Sign Up and Login

These functions will be available to all the users.

The sign up functionality allows users to register themselves to the website. In particular, each user will be asked to provide an email (the email will be the username of the user) and a password. If the email is not registered yet (a user can not register themselves twice) a verification email is sent to the user. Once the user has confirmed the email, the system will ask them which actor of the system they represent (farmer agronomist or policy maker), and depending on this choice the system will ask the user to insert role specific data, for example:

- A farmer will be asked to insert data regarding their farm
- An agronomist will be asked to insert the area in which she/he works and their professional code (this code will be used by the system's administrators in order to control if the agronomist is a real agronomist)
- A policy maker will be asked to identify themselves

The login functionality will allow registered users to login in the system by using the correct email and password tuple provided during the registration phase.

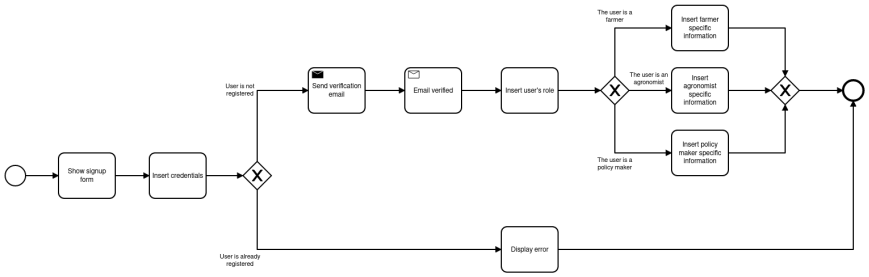

Figure 2.2: BPMN for the "Sign Up" functionality

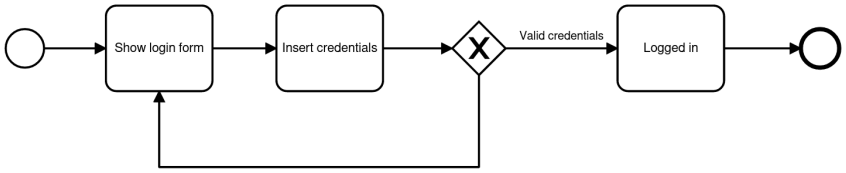

Figure 2.3: BPMN for the "Login" functionality

2.2.2 Consult data

This function will be available to all users.

Each user will be able to query the system in order to retrieve data. In particular, each user will be able to access different type of data:

- Each farmer will be able to see data regarding their farm and personalized suggestions on how to be more productive and weather resilient.
- Each agronomist will be able to see data regarding each farm in their area and data regarding the area they are responsible for as a whole. Moreover each agronomist will be able to see which farmers belongs to their area.
- Each policy maker will be able to see data about all the farms in Telangana and about each area in which Telangana is split in.

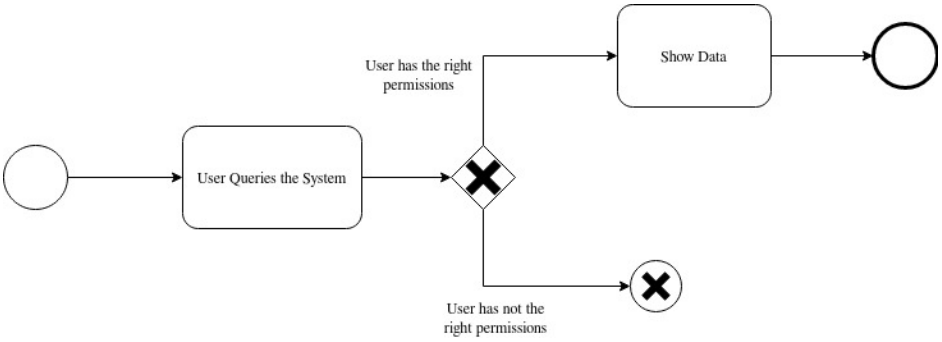

Figure 2.4: BPMN for the "Consult data" functionality

2.2.3 Insert data

This function will be available to agronomists and farmers.

This functionality will allow agronomists and farmers to insert data into the system. In particular, the system will allow farmers to insert data about the production of the farm, while the agronomist will be able to insert data regarding measurements taken during their visits. Moreover, agronomists will be able to insert a report of each farm they have visited. After the insertion of data the system must validate them, making sure that the inserted data makes sense.

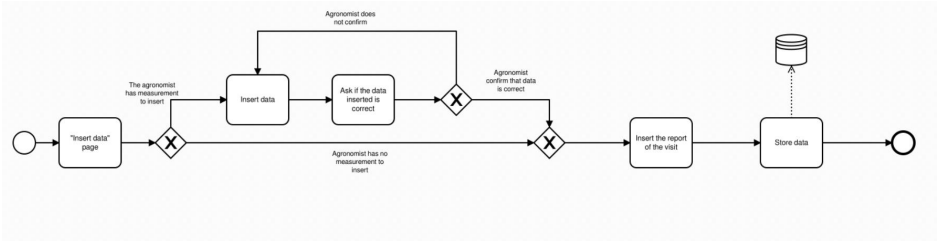

Figure 2.5: BPMN for the "Insert data" functionality, agronomist's side

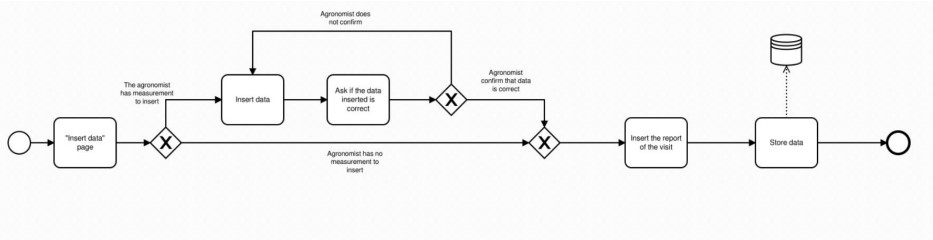

Figure 2.6: BPMN for the "Insert data" functionality, farmer's side

2.2.4 Chat between farmers and agronomists

This function will be available to farmers only.

The system will allow farmers to request the help of an agronomist by using a chat. In particular a farmer can open a ticket asking for the help of an agronomist responsible for their area. An agronomist then can handle the ticket, thus opening a chat. The system therefore will allow the two to communicate through the chat. Once the problem is solved, the agronomist will be able to close the chat.

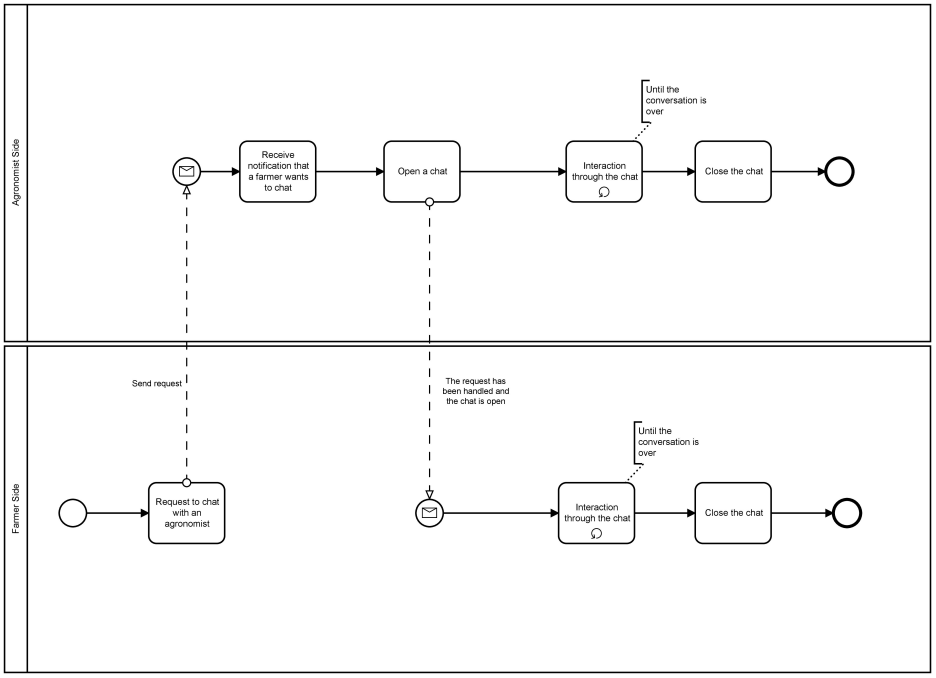

Figure 2.7: BPMN for the "Chat between farmers and agronomists" functionality

2.2.5 Request a visit by an agronomist

This function will be available to farmers.

The system will allow farmers to request an agronomist to visit their farm in order to help them. In particular, the farmer will be able to request a visit through the system. Once the request has been sent, it will be received by all the agronomists responsible for the area the farmer belongs to, and it can be handled by one and only one among the agronomists. In order to handle the request, an agronomist will schedule a visit in their

schedule. This will cause the system to send a notification to the farmer informing them that a visit has been booked, showing them both the date and the time of the visit.

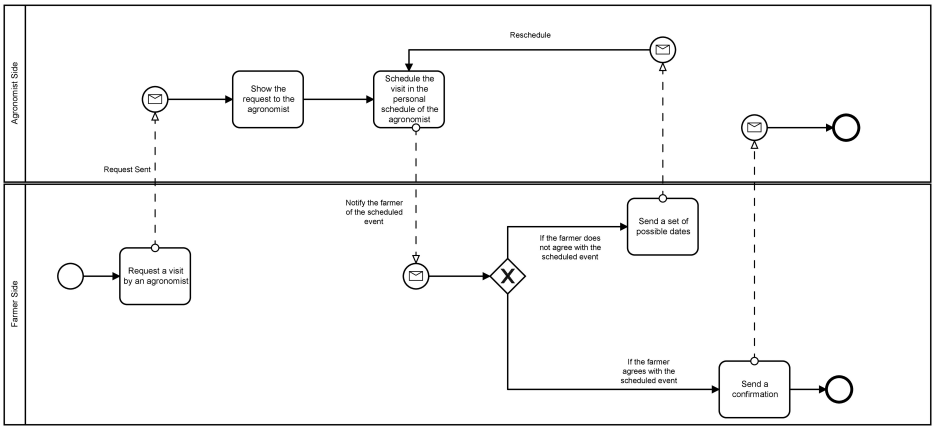

Figure 2.8: BPMN for the "Request a visit by an agronomist" functionality

2.2.6 Interaction between farmers and agronomists through a forum

This function will be available to farmers and agronomists.

The system will allow farmers and agronomists to interact through a forum. In particular, each of them will be able to open a new thread or to write into an existing one.

2.2.7 Schedule a visit to a farm

This function will be available to agronomists.

Because all farms must be visited at least twice a year, and those that are under-performing should be visited more often, this functionality will allow an agronomist to see the list of farms they are responsible of, ordered by priority (first the ones who are performing poorly and the ones who requested for a visit). By consulting this list the agronomist will be able to schedule, in their schedule, one or more visits to the various farms who need help. By scheduling a visit the priority of the farm will decrease and a notification will be sent to each scheduled farmer.

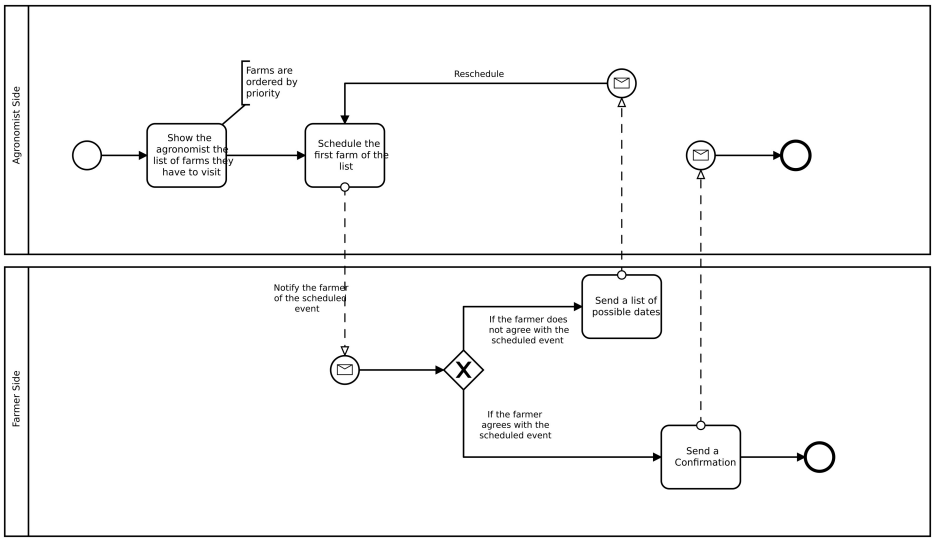

Figure 2.9: BPMN for the "Schedule a visit to a farm" functionality

2.2.8 Confirm the daily schedule

This function will be available to agronomists.

At the end of each day each agronomist will be able to confirm the visits done during the day. In particular, each agronomist will be able to insert data measurements taken during the daily visits, and they will be forced to write a report for each farm they have visited. If an agronomist did not visit a scheduled farm, for whatever reason, no report will be inserted and the system will increase the priority of the farm in order to make it appear on top of the list of all the farms of the area.

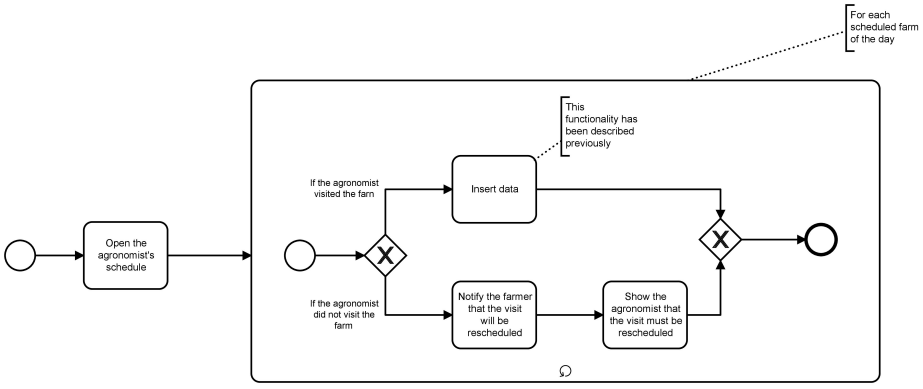

Figure 2.10: BPMN for the "Confirm the daily schedule" functionality

2.2.9 Cancel a scheduled visit

This function will be available to agronomists.

In every moment an agronomist will be able to cancel a scheduled event in their schedule. The system then will notify the farmer that the scheduled event has been canceled. Moreover, the system will increment the priority of the farm in order to make the agronomist keep in mind that the visit must be rescheduled as soon as possible.

Figure 2.11: BPMN for the "Cancel a scheduled visit" functionality

2.2.10 Fetch data

The system will periodically fetch data from different available databases. Once fetched, data will be cleaned, transformed in order to make them homogeneous, and analyzed in order to extract new knowledge from them. In particular, the system will be able to infer which farms are performing particularly well and the ones which are performing poorly (e.g. by leveraging machine learning and in particular clustering).

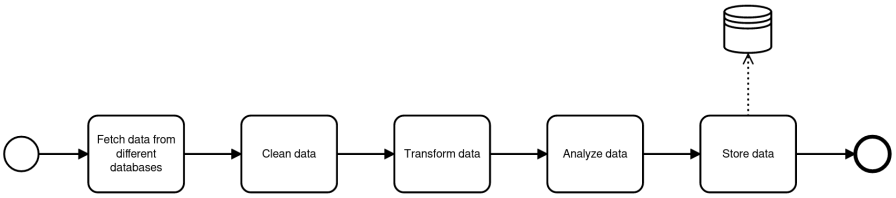

Figure 2.12: BPMN for the "Fetch data" functionality

2.3 User Characteristics

2.3.1 Policy Makers

They are the government users, they are interested in knowing how Telangana's Farmers are performing and they want to know why and how the performances are going so. Using their dashboard they will be able to see, filter and compare all the data stored in the system matching also with the report written by the agronomist. Studying the most effective strategies against meteorological adverse events in order to help spread out the best practice.

2.3.2 Farmers

A farmer is a person who has one or more pieces of land and wants an external, possibly professional, support in order to choose the better strategy to maximize the harvest, deals with problems that can derive from the climate and also which product is better to use considering the land he is cultivating. They will be in contact with other farmers through a forum so that they can exchange advice on the cultures. They will be able to contact both agronomists using the forum and/or one of the Agronomists who are assigned to the same area the Farmer is assigned as well, in order to get any help.

2.3.3 Agronomists

An Agronomist is a professional figure that decides to sign up into the DREAM project as a user who will take into account an area to which more farmers are associated. Being a professional figure, the Agronomist will be able to solve Farmer's problems by giving them advice such as which fertilizer they should use, when they should do the harvest and so on. They will use the application to see how Farmers all over Telangana are performing, in

terms of culture performance and will do visits to the farms that need it. Every Agronomist will be asked to produce a report so that the strategies applied to each farm are tracked. They can use a dashboard in order to filter the available data so that they will be able to compare different farms and so.

2.4 Domain Assumptions

(D1) There are enough agronomists for each area

(D2) Every user has an internet connection in order to interact with the system

(D3) Every farmer has the capability to interact with the system

(D4) Every agronomist has the capability to interact with the system

(D5) Every policy maker has the capability to interact with the system

(D6) Every farmer has an electronic device in order to interact with the system

(D7) Every agronomist has an electronic device in order to interact with the system

(D8) Every policy maker has an electronic device in order to interact with the system

(D9) The farmers insert true and correct data about their production

(D10) The agronomists insert true and correct data about measurements

(D11) The reports inserted by agronomists are sufficiently detailed

(D12) The agronomists always confirms all the visits that they have done, sooner or later

(D13) A visit in an agronomist schedule is confirmed only if it is actually done by the agronomist

(D14) All the agronomists have at least decent capabilities of managing and organizing their own schedule

(D15) All those agronomist that take measurements do so in a consistent way

(D16) The farmers follows agronomist's advice

(D17) Only real and capable agronomists are allowed to subscribe to the system

(D18) Only real and capable policy maker are allowed to subscribe to the system

(D19) The agronomist that wants to visit a farm will reach an agreement with the owner farmer on the day and date of the appointment within about 14 days from the first appointment request

(D20) Each farmer's account owns one and only one farm

(D21) Each farm is owned by one and only one farmer

(D22) Every user has its own credentials

(D23) The credentials used to access the system are not shareable

(D24) External databases are correctly and periodically updated

Chapter 3

Specific Requirements

3.1 Requirements

Login

(R1) The system allows users to sign up

(R2) Every user can access the system only if they are registered

(R3) The system must allow to access the system only if the credentials used are correct

(R4) The system must not allow to register two different users with the same username

(R5) The system must not allow the same user to register themselves more than once

Preliminary requirements

(R6) The system shall associate to each area one or more agronomists

(R7) The system shall associate each farmer to one and only one area

(R8) Each agronomist is associated to one and only one area

Show data

(R9) The system allows the agronomist to see only the list of farms for which they are responsible of

(R10) The system allows the farmer to see data about their farm

(R11) The system must not allow the farmer to see data about farms other than theirs

(R12) The system allows the agronomist to see data about a specific farm

(R13) The system allows the agronomist to see data about the area for which they are responsible for

(R14) The system must not allow the agronomist to see data about farms for which they are not responsible for

(R15) The system must not allow the agronomist to see data about an area for which they are not responsible for

(R16) The system allows the policy maker to see data about a specific farm

(R17) The system allows the policy maker to see data about an area

(R18) The system allows the policy maker to see data about all the areas in Telangana

(R19) The system must fetch data from different sources containing relevant information for the scope of the application

(R20) The system must elaborate and aggregate data fetched from different sources

Chat between farmer and agronomist

(R21) The system allows farmers to request to have a chat with an agronomist

(R22) The system allows agronomists to accept chat requests

(R23) The system allows farmers and agronomists to interact through a chat

(R24) A farmer, belonging to an area, can chat only with an agronomist belonging to the same area

(R25) Each farmer can have at most one open chat at any given time

(R26) Each agronomist can have multiple chats open at any given time

(R27) The system allows only agronomists to close chats

Booking a visit

(R28) The system allows farmers to request a visit from an agronomist

(R29) The system must show an agronomist a list of farms they have to visit, ordering them by priority

(R30) The system must prioritize the farms that need help

(R31) The agronomist must prioritize the farms that the system prioritize

(R32) The system allows agronomists to schedule a visit at any farm of their competence

(R33) The system allows agronomist to set up a daily schedule

(R34) The system must notify a farmer when an agronomist schedules a visit to their farm

(R35) The system must notify an agronomist when a farmers cancels a scheduled visit to their farm

(R36) The system must notify a farmer when an agronomist cancels a scheduled visit to their farm

(R37) The system allows agronomists to cancel a scheduled visits within one day before

Forum between farmers and agronomists

(R38) The system shall allow farmers to interact through a forum with other farmers and agronomists

(R39) The system shall allow agronomists to interact through a forum with other agronomists and farmers

(R40) The forum allow to each farmer or agronomist with each other no matter the area they belong to

Visit

(R41) The system shall allow agronomists to confirm what they have done during the day

(R42) The system shall allow agronomists to insert a report regarding the visits they have done during the day

(R43) The system shall allow to remove a visit from the agronomist's screen only if they fill out the report

(R44) The system shall allow agronomists to insert data regarding measurements taken during the visits done during the day

3.2 External Interface Requirements

3.2.1 User Interfaces

The system should interface with users through devices which must be connected to the Internet.
Everyone that wants to use the system must connect to it through a Web Interface (from an existent domain, like www.dream.gov) or through a mobile application that can be installed on smartphones and tablets.

3.2.2 Software Interfaces

The system will use some important external interfaces in order to accomplish its functionalities.
The system, in fact, must fetch data from various databases, therefore it must implement different interfaces in order to fetch data correctly.
The system, then, must elaborate the fetched data. In order to do this it could use an open-source machine learning platform such as *TensorFlow*.
For obvious reasons, the system should use an API provided by a map service owner, like *OpenStreetMaps*, which will show where the various farms in Telangana are.
In order to handle agronomists' schedules, the system must use a calendar API such as the one offered by *Google Calendar*. This will allow agronomists to schedule visits to farmers and to keep track of their appointment.
Moreover, in order to handle the chat between farmers and agronomists, the system may use an open source chat API such as *Chat SDK*

3.2.3 Hardware Interface

The system must be able to interface itself with various sensors, spread across Telangana, which measure the humidity of the soil.

3.3 Functional Requirements

In this section are listed all the use case that we think are important to understand properly the whole system. At first we present the *use case diagrams*, which can be used to get the scope of each *Actor*. Then we inserted the *use case tables*, in order to give a more detailed description of each use case in a summarized view. On *Figure 3.2* we used a generalization of some use cases: as "Chat use", "Ask for help" and "Forum use". Those use cases haven't a proper table, but we think that generalizing them helps the reader to understand the overall diagram. While "Update priority" is considerable as a sub- use cases, due to the fact that can be seen into some use-cases such as the owns that are connected to it with the arrow labeled with "include".

3.3.1 Use case diagrams

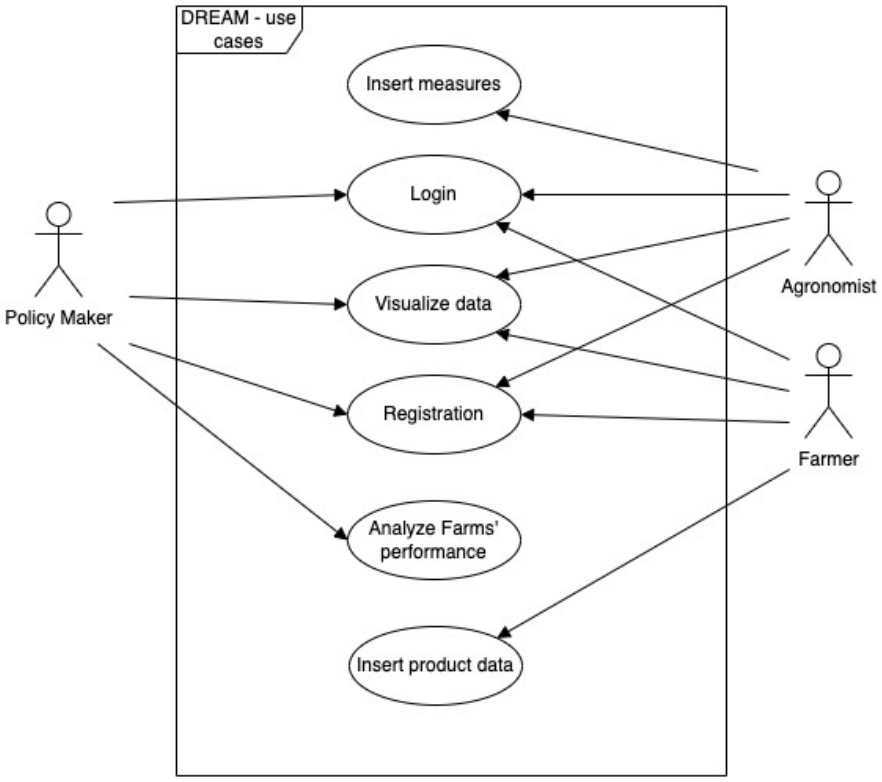

Figure 3.1: Use Case Diagram 1

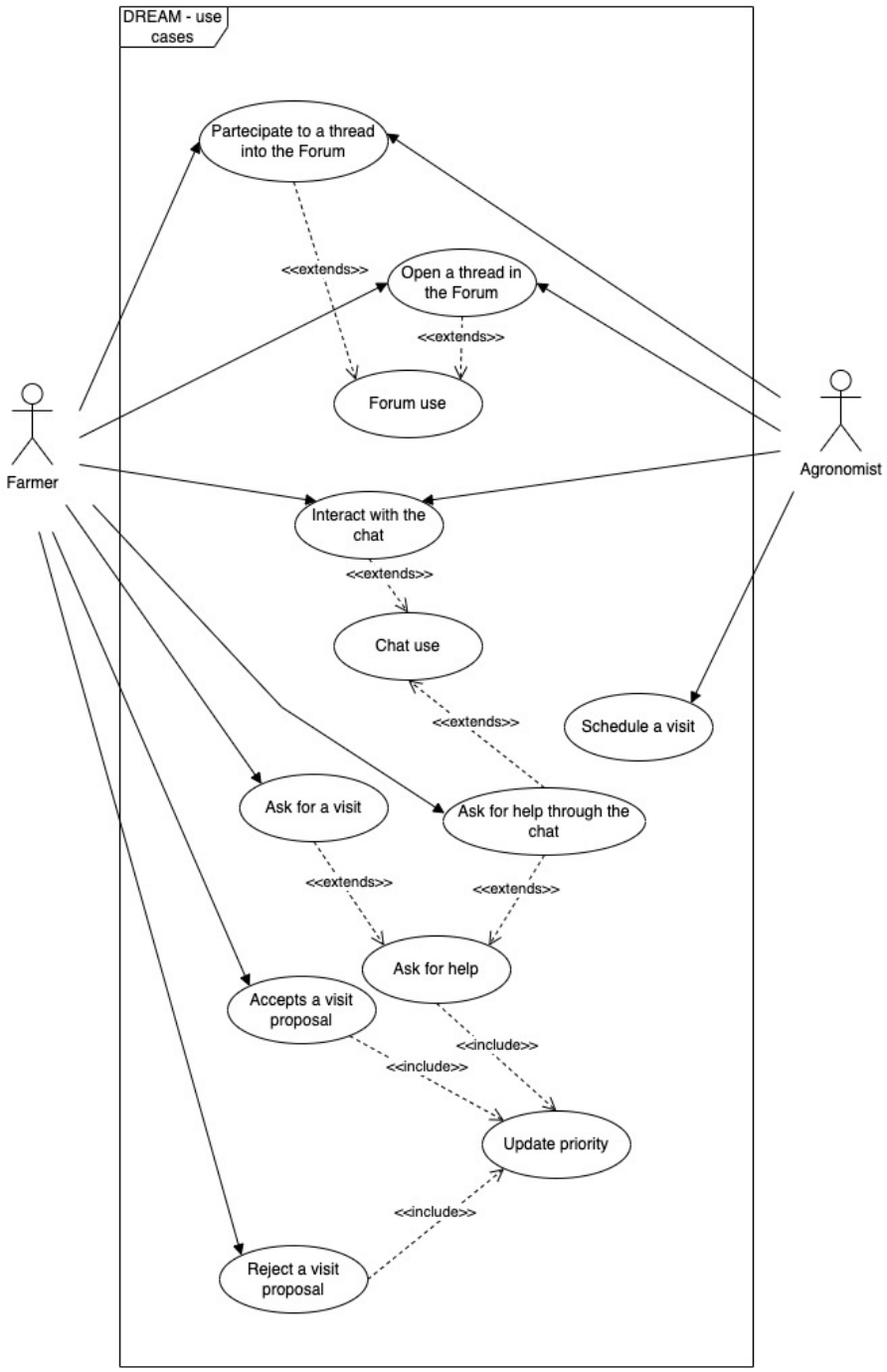

Figure 3.2: Use Case Diagram 2

3.3.2 User's use cases

Use case 0.1: User Login

ID	0.1

Name	User Login
Actors	User
Entry conditions	• The user is already registered
Input	User's credentials
Event flows	• The user opens the application • The user inserts their credentials • The system checks the credentials • The system lets the user login
Exit Condition	The credentials are correct
Output	The user is logged in
Exceptions	• The credentials are wrong, therefore the system prevents the access

Table 3.1: Use case 0.1 - User login

Use case 0.2: User visualizes data summary

ID	0.2
Name	User visualizes data summary
Actors	User
Entry conditions	• The user has logged in the system • The data are stored in the database
Input	NULL
Event flows	• The user opens the application • The user clicks on the button "Visualize data" • The system shows the available data views • The user selects the data views they are interested in • The system checks if the user has the right permission to visualize data they are requiring • The system fetches the required data • The system displays the required data

Exit Condition	Data are fetched and properly displayed
Output	Data summary is displayed
Exceptions	• The system fails to fetch data • The user has not the right permissions to see the requested data

Table 3.2: Use case 0.2 - User visualizes data summary

3.3.3 Farmer's use cases

Use case 1.1: Registration of a farmer

ID	1.1
Name	Registration of the farmer
Actors	Farmer
Entry conditions	• The farmer is not already registered in the system
Input	Farmer's credentials

Event flows	• The farmer opens the application • The farmer clicks on the button "Sign Up" • The system shows the registration form • The farmer inserts their email and a password • The system sends a confirmation email to the farmer • The farmer confirms their email by clicking in the link received by email • The system shows a form asking for the role of the farmer • The farmer selects the option "Farmer" • The system displays a form asking for data regarding the farmer's farm • The farmer inserts data about their farm • The system calculate the priority of the farm w.r.t. the list of farms belonging to the same area • The system displays information about the area the farmer belongs to • The system displays the list of agronomists the farmer will be able to reach • The system sends a confirmation email to the farmer
Exit Condition	Registration has been successful. The farmer's data is stored into the system's database. The farmer then will be able to login in the system by using their credentials.
Output	• A verification email is sent to the farmer • A confirmation email is sent to the farmer
Exceptions	• The farmer is already registered into the system. The system then will prevent the user to register themselves twice. • The farmer provides an invalid email. The farmer then will not be able to confirm their email.

Table 3.3: Use case 1.1 - Registration of the farmer

Use case 1.2: Farmer asks help through the chat

ID	1.2

Name	Farmer asks help through the chat
Actors	Farmer
Entry conditions	• The farmer needs help • The farmer is already registered in the system
Input	The farmer inserts a description of the problem they are facing
Event flows	• The farmer has a problem • The farmer opens the application • The farmer clicks on "Chat with an Agronomist" • The system shows a form asking for information about the problem the farmer is facing • The farmer inserts the required information • The farmer clicks on "Send Request" • The system acquires the data sent by the farmer • The system confirms that the request has been successful • The system shows, in the farmer interface, the request as "pending"
Exit Condition	The request has been successfully sent to all the agronomists responsible of the area the farmer belongs to.
Output	• The request is marked as "pending"
Exceptions	• The farmer misses to fill one or more mandatory fields during the chat request. The system then will show an error such as "in order to open the chat request you need to fill up all the mandatory fields". • The farmer closes the chat without clicking on "Send Request". By doing so the request is never sent but remains in the Farmer's chat interface as a draft.

Table 3.4: Use case 1.2 - Farmer asks help through the chat

Use case 1.3: Farmer asks for a visit

ID	1.3
Name	Farmer asks for a visit
Actors	Farmer

Entry conditions	• The farmer needs help • The farmer is already registered in the system
Input	The farmer inserts a description of the problem they are facing
Event flows	• The farmer has a problem • The farmer opens the application • The farmer clicks on "Request a Visit" • The system shows a form asking for information about the problem the farmer is facing • The farmer inserts the required information • The farmer clicks on "Send Request" • The system acquires the data sent by the farmer • The system updates the priority of the farm • The system confirms that the request has been successful • The system shows, in the farmer interface, the request as "pending"
Exit Condition	The priority of the farm is updated. putting it on top of the list of farms belonging to the same area. Moreover the farm is marked in such a way that agronomists will be able to notice that the farmer has requested a visit.
Output	The priority of the farm is updated putting it on top of the list of farms belonging to the same area. Moreover the farm is marked in such a way that agronomists will be able to notice that the farmer has requested a visit.
Exceptions	• The farmer misses to fill one or more mandatory fields during the chat request. The system then will show an error such as ""in order to open the chat request you need to fill up all the mandatory fields. • The farmer closes the chat without clicking on "Send Request". By doing so the request is never sent but remains in the Farmer's chat interface as a draft.

Table 3.5: Use case 1.3 - Farmer asks for a visit

Use case 1.4: Farmer accepts a visit proposal

ID	1.4

Name	Farmer accepts a visit proposal
Actors	Farmer
Entry conditions	• An agronomist scheduled a visit to the farmer's farm • The farmer is already registered in the system
Input	Visit proposal by an agronomist
Event flows	• The system notifies the farmer that there is a visit proposal • The farmer taps on the notification • The farmer clicks on "Request a Visit" • The system shows details about the proposed visit, such as the date and the time of the visit • The farmer accepts the visit by clicking on the "Confirm" button • The system shows both to the farmer and to the agronomist that the visit has been scheduled
Exit Condition	The farmer accepts the proposed visit.
Output	• A notification is sent on the agronomist's device. • A notification is sent on the farmer's device.
Exceptions	The farmer does not confirm nor rejects the proposed visit. In this case the system should continue pushing the notification containing the proposed visit.

Table 3.6: Use case 1.4 - Farmer accepts a visit proposal

Use case 1.5: Farmer rejects a visit proposal

ID	1.5
Name	Farmer rejects a visit proposal
Actors	Farmer
Entry conditions	• An agronomist scheduled a visit to the farmer's farm • The farmer is already registered in the system
Input	Visit proposal by an agronomist

Event flows	• The system notifies the farmer that there is a visit proposal • The farmer taps on the notification • The farmer clicks on "Request a Visit" • The system shows details about the proposed visit, such as the date and the time of the visit • The farmer rejects the visit by clicking on the "Reject" button • The system shows the farmer a form asking to propose new dates for the visit • The farmer selects new dates and clicks on "Propose Dates" • The system confirms the farmer that the dates have been forwarded to the agronomist • The system notifies the agronomist about the rejection and shows them the new proposed dates
Exit Condition	The agronomist is notified that the farmer has rejected the visit
Output	• A notification is sent on the agronomist's device. • A notification is sent on the farmer's device.
Exceptions	• The farmer tries to reject the upcoming visit on the same day of the visit which is forbidden by the system • The farmer does not confirm nor rejects the proposed visit. In this case the system should continue pushing the notification containing the proposed visit.

Table 3.7: Use case 1.5 - Farmer rejects a visit proposal

Use case 1.6: Farmer and Agronomist interact through the chat

ID	1.6
Name	Farmer and Agronomist interact through the chat
Actors	Farmer and Agronomist
Entry conditions	• The farmer has an open chat with an agronomist • The farmer is already registered in the system
Input	Messages in the chat

Event flows	• The notification "A chat has been opened" is received by the farmer • The farmer taps on the notification • The system shows the farmer the first message written by the agronomist • The farmer reads the answer and in case there are more doubt they can keep asking questions using the chat • The system notifies the agronomist that one ore more messages have been written in the chat • The agronomist taps on the notification • The system shows the agronomist the new messages • The agronomist reads the new messages and replies to them • ... • When the conversation is over the agronomist send a request to close the chat • The system notifies the farmer that the agronomist wants to close the chat • The farmer approves the action of the agronomist by clicking on "Confirm" • The system closes the chat • The system saves the chat history in its databases
Exit Condition	The farmer gets all the answer they needed from the agronomist, which, therefore, proposes to close the chat
Output	Chat history
Exceptions	The agronomist proposes to close the chat, but the farmer ignores the notification. In this case the system closes the chat automatically in 24h.

Table 3.8: Use case 1.6 - Farmer and Agronomist interact through the chat

Use case 1.7: Farmer opens a thread in the forum

ID	1.7
Name	Farmer opens a thread in the forum
Actors	Farmer

Entry conditions	• The farmer wants to share some thoughts • The farmer is already registered in the system
Input	Title of the thread and description of the thread
Event flows	• The farmer opens the application • The farmer clicks on the "Forum" button • The system display the screen relative to the forum • The farmer taps on the button "Open a Thread" • The system displays a forum asking for a title for the thread and the description of the thread • The farmer fills up the fields • The farmer clicks on the "Confirm" button • The system opens the required thread • The system shows the new thread to all the farmer which navigate through the forum
Exit Condition	The farmer successfully publish their new thread on the forum
Output	The system have a new thread
Exceptions	• The title of the thread provided by the farmer has already been used • The farmer closes the application before clicking the "Confirm" button. By doing so all the content are saved as draft for the next time the farmer wants to open a new thread

Table 3.9: Use case 1.7 - Farmer opens a thread in the forum

Use case 1.8: Farmer participates to a thread in a forum

ID	1.8
Name	Farmer participates to a thread in a forum
Actors	Farmer
Entry conditions	The farmer is already registered in the system
Input	A message sent by the farmer

Event flows	• The farmer opens the application • The farmer clicks on the "Forum" button • The system display the screen relative to the forum, and in particular the list of all the thread order by the most recent to the least recent • The farmer selects a thread • The system displays all the messages belonging to the thread and the form used to append new messages to the thread itself • The farmer appends a message to the thread • The farmer clicks on the "Comment" button • The system appends the new message to the thread
Exit Condition	The farmer successfully publish their message into the thread
Output	The thread has a new message in it
Exceptions	The Farmer closes the application before clicking the "Comment" button. By doing so all the content is lost.

Table 3.10: Use case 1.8 - Farmer participates to a thread in a forum

Use case 1.9: Farmer inserts data about production

ID	1.9
Name	Farmer inserts data about production
Actors	Farmer
Entry conditions	• The farmer is already registered in the system • The farmer has already done the harvest
Input	Farm's production data

Event flows	• The farmer opens the application • The farmer clicks on the "Production Data" button • The system display a form asking for information about the latest harvest • The farmer fills up the form with data relative to their production • The farmer clicks on the "Confirm" button • The system stores the data inside a database • The system updates the priority of the farm
Exit Condition	The farmer successfully inserts the data regarding their production
Output	NULL
Exceptions	The Farmer closes the application before clicking the "Confirm" button. By doing so all the content is lost.

Table 3.11: Use case 1.9 - Farmer inserts data about production

3.3.4 Agronomist's use cases

Use case 2.1: Registration of the agronomist

ID	2.1
Name	Registration of the agronomist
Actors	Agronomist
Entry conditions	• The agronomist is not already registered in the system
Input	Agronomist's credentials

Event flows	• The agronomist opens the application • The agronomist clicks on the button "Sign Up" • The system shows the registration form • The agronomist inserts their email and a password • The system sends a confirmation email to the agronomist • The agronomist confirms their email by clicking in the link received by email • The system shows a form asking for the role of the agronomist • The farmer selects the option "Agronomist" • The system displays a form asking for data regarding the agronomist; in particular, among the others, the are in which they work in • The agronomist fills the form with the required data • The system displays information about the area the agronomist belongs to • The system displays the list of farm the agronomist will be responsible of • The system sends a confirmation email to the farmer
Exit Condition	Registration has been successful. The agronomist's data is stored into the system's database. The agronomist then will be able to login in the system by using their credentials.
Output	• A verification email is sent to the agronomist • A confirmation email is sent to the agronomist
Exceptions	• The agronomist is already registered into the system. The system then will prevent the user to register themselves twice. • The agronomist provides an invalid email. The agronomist then will not be able to confirm their email.

Table 3.12: Use case 2.1 - Registration of the agronomist

Use case 2.2: Agronomist takes charge of a chat request

ID	2.2
Name	Agronomist takes charge of a chat request
Actors	Agronomist
Entry conditions	• The agronomist has a chat request pending • The request has been issued by a farmer that belongs to the same area of the agronomist • The agronomist is already registered in the system • The agronomist has logged in the system
Input	A message from the agronomist
Event flows	• The system notifies the agronomist that a farmer has requested to chat with them • The agronomist taps on the notification • The system shows the agronomist the list of pending chat requests • The agronomist selects a chat request and clicks on "Process request" • The system opens a chat between the farmer who requested it and the agronomist • The notifies shows both to the farmer and to the agronomist that the chat is open • The agronomist writes the first message in the chat
Exit Condition	The chat is opened.
Output	There exists a chat between the agronomist and the farmer
Exceptions	• The agronomist closes the chat before clicking on "Process request", so the chat request is still "pending" • The agronomist clicks on "Process request" but in the meanwhile the request was opened by another agronomist in the same area. The system shows them an error message: "This chat request is already opened by another agronomist"

Table 3.13: Use case 2.2 - Agronomist takes charge of a chat request

Use case 2.3: Agronomist schedules a visit to a farm

ID	2.3
Name	Agronomist schedules a visit to a farm
Actors	Agronomist

Entry conditions	• The agronomist has is registered in the system • The agronomist has logged in • The agronomist wants to add a visit to their schedule
Input	NULL
Event flows	• The agronomist opens the application • The system shows the agronomist their dashboard • The agronomist taps on the "Farms List" button • The system shows the list of the farms, that belong to the area of the agronomist, ordered by priority and possibly marked if the farmer requested a visit • The agronomist selects the top farm and clicks on "Schedule a visit" • The system opens the schedule of the agronomist and asks them when they want to visit the selected farm • The agronomist selects the date and the time in which they want to visit the farm • The agronomist clicks on "Confirm" • The system updates the agronomist's schedule adding this visit • The system sends the information about the upcoming visit to the farmer
Exit Condition	The visit has been successfully inserted in the agronomist's schedule, and the details have been successfully sent to the farmer.
Output	The relative farmer is notified that there is an upcoming visit scheduled
Exceptions	The agronomist closes the chat before clicking on "Confirm". In this case the visit is not scheduled and the notification to the farmer is not sent.

Table 3.14: Use case 2.3 - Agronomist schedules a visit to a farm

Use case 2.4: Agronomist confirms a scheduled visit

ID	2.4
Name	Agronomist confirms a scheduled visit
Actors	Agronomist

Entry conditions	• The agronomist has logged in
Input	• The daily schedule of the agronomist • A visit that the agronomist has completed
Event flows	• The agronomist opens the application • The system shows the agronomist their dashboard • The agronomist taps on the "Schedule" button • The system shows the agronomist their daily schedule • The agronomist selects a visit and confirms it by clicking on "Done" • The system shows a form that allows the agronomist to insert data about measurements taken during the visit and a report of the visit. The report of the visit is marked as mandatory. • The agronomist fills in the form • The agronomist clicks on "Confirm" • The system stores the data in the database
Exit Condition	The system has updated the schedule with the confirmed visit and it has stored the data concerning the confirmed visit.
Output	• The report of the visit • (Optional) Data about measurements taken by the agronomist
Exceptions	The agronomist does not fill the report field of the form, thus it is impossible to confirm the visit

Table 3.15: Use case 2.4 - Agronomist confirms a scheduled visit

Use case 2.5: Agronomist cancels a scheduled visit

ID	2.5
Name	Agronomist cancels a scheduled visit
Actors	Agronomist
Entry conditions	• The agronomist has logged in

Input	• The daily schedule of the agronomist • A visit that the agronomist can not attend
Event flows	• The agronomist opens the application • The system shows the agronomist their dashboard • The agronomist taps on the "Schedule" button • The system shows the agronomist their schedule • The agronomist selects a visit and cancels it by clicking on "Cancel" • The system shows a form that allows the agronomist to send a message to the farmer in order to explain why the visit can not take place as scheduled. • The agronomist fills in the form • The agronomist clicks on "Confirm" • The system notifies the farmer about the cancelled visit
Exit Condition	• The system has updated the schedule considering the canceled visit • The farmer is notified that the scheduled visit has been canceled
Output	A notification is sent to the farmer.
Exceptions	The agronomist tries to cancel a visit that is already confirmed. The system forbids the operation.

Table 3.16: Use case 2.5 - Agronomist cancels a scheduled visit

Use case 2.6: Agronomist opens a thread in the forum

ID	2.6
Name	Agronomist opens a thread in the forum
Actors	Agronomist
Entry conditions	• The agronomist wants to share some thoughts • The agronomist is already registered in the system
Input	Title of the thread and description of the thread

Event flows	• The agronomist opens the application • The agronomist clicks on the "Forum" button • The system display the screen relative to the forum • The agronomist taps on the button "Open a Thread" • The system displays a forum asking for a title for the thread and the description of the thread • The agronomist fills up the fields • The agronomist clicks on the "Confirm" button • The system opens the required thread • The system shows the new thread to all the agronomist which navigate through the forum
Exit Condition	The agronomist successfully publish their new thread on the forum
Output	The system have a new thread
Exceptions	• The title of the thread provided by the agronomist has already been used • The agronomist closes the application before clicking the "Confirm" button. By doing so all the content are saved as draft for the next time the agronomist wants to open a new thread

Table 3.17: Use case 2.6 - Agronomist opens a thread in the forum

Use case 2.7: Agronomist participates to a thread in a forum

ID	2.7
Name	Agronomist participates to a thread in a forum
Actors	Agronomist
Entry conditions	The agronomist is already registered in the system
Input	A message sent by the agronomist

Event flows	• The agronomist opens the application • The agronomist clicks on the "Forum" button • The system display the screen relative to the forum, and in particular the list of all the thread order by the most recent to the least recent • The agronomist selects a thread • The system displays all the messages belonging to the thread and the form used to append new messages to the thread itself • The agronomist appends a message to the thread • The agronomist clicks on the "Comment" button • The system appends the new message to the thread
Exit Condition	The agronomist successfully publish their message into the thread
Output	The thread has a new message in it
Exceptions	The Agronomist closes the application before clicking the "Comment" button. By doing so all the content is lost.

Table 3.18: Use case 2.7- Agronomist participates to a thread in a forum

3.3.5 Policy Maker's use cases

Use case 3.1: Registration of the Policy Maker

ID	3.1
Name	Registration of the Policy Maker
Actors	Policy Maker
Entry conditions	The policy maker is not already registered in the system
Input	Policy maker's credentials

Event flows	• The policy maker opens the application • The policy maker clicks on the button "Sign Up" • The system shows the registration form • The policy maker inserts their email and a password • The system sends a confirmation email to the policy maker • The policy maker confirms their email by clicking in the link received by email • The system shows a form asking for the role of the policy maker • The farmer selects the option "Policy maker" • The system shows a form asking for the personal information used in order to identify the policy maker • The policy maker inserts the requested data • The system sends a confirmation email to the Policy Maker
Exit Condition	Registration has been successful. The farmer's data is stored into the system's database. The policy maker then will be able to login in the system by using their credentials.
Output	• A verification email is sent to the Policy Maker • A confirmation email is sent to the Policy Maker
Exceptions	• The Policy Maker is already registered into the system. The system then will prevent the user to register themselves twice. • The Policy Maker provides an invalid email. The Policy Maker then will not be able to confirm their email.

Table 3.19: Use case 3.1 - Registration of the Policy Maker

Use case 3.2: Policy Makers analyze farmer's performance

ID	3.2
Name	Policy Makers analyze farmer's performance
Actors	Policy Maker

Entry conditions	• The Policy Maker has already logged in the system • The data are stored in the database • The machine learning techniques have to be already defined
Input	NULL
Event flows	• The Policy Maker clicks on "Compute Performance" data summary • The system computes the rank of the best farms using machine learning techniques already defined • The system shows the ranking computed before
Exit Condition	The data are fetched and properly displayed
Output	The ranking built.
Exceptions	The system fails to fetch the data.

Table 3.20: Use case 3.2 - Policy Makers analyze farmer's performance

3.3.6 Activity Diagrams

Activity Diagram: User Registration

The following three activity diagrams maps the use case tables 1.1, 2.1, 3.1 respectively.

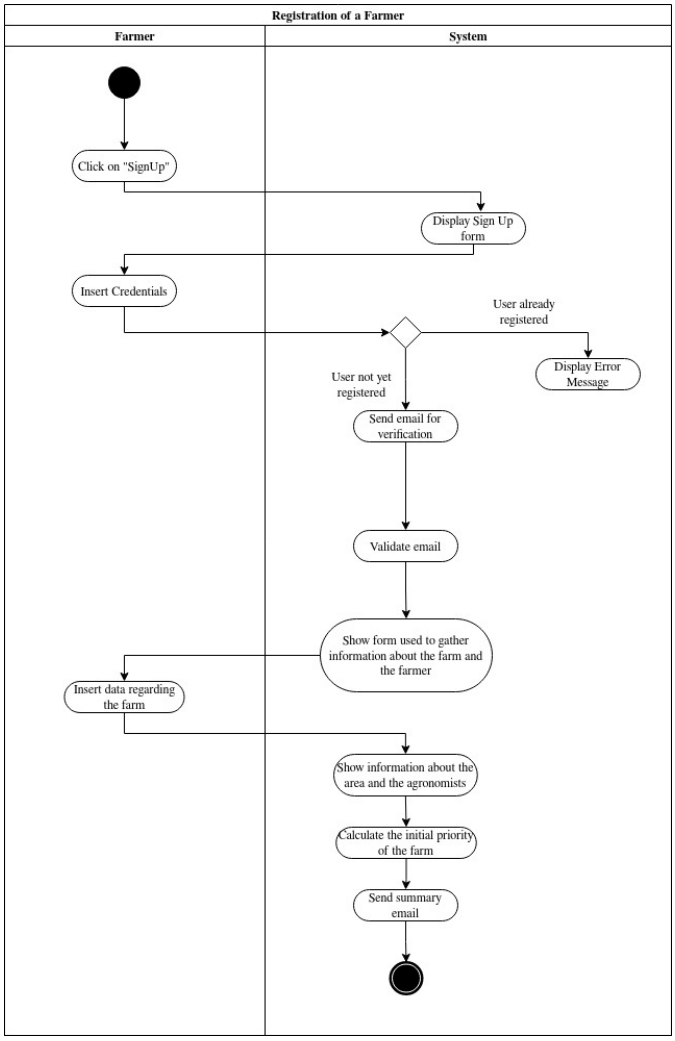

Figure 3.3: Activity diagram: Farmer Sign Up

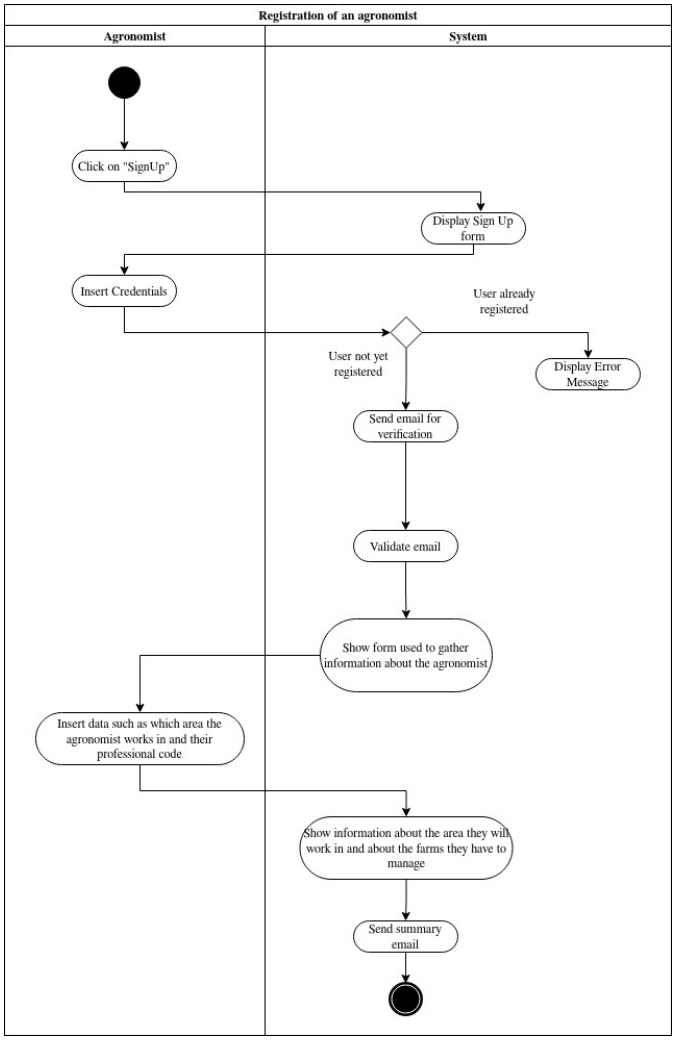

Figure 3.4: Activity diagram: Agronomist Sign Up

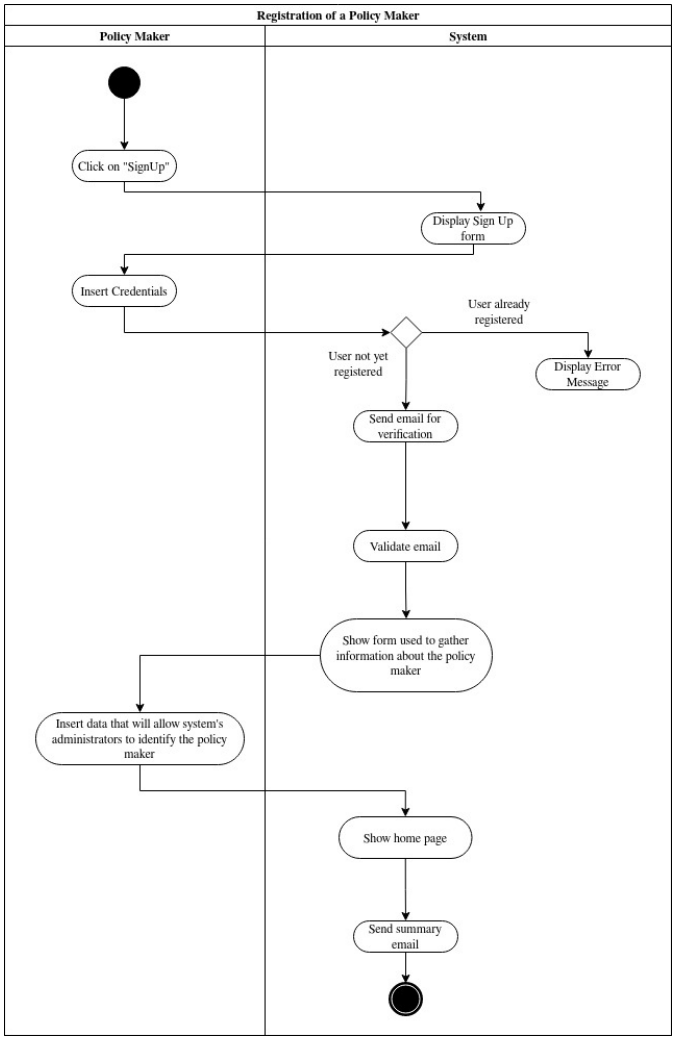

Figure 3.5: Activity diagram: Policy Maker Sign Up

Activity Diagram: User Login

The following activity diagram describes the login process of the system. In particular, it maps the use case 0.1.

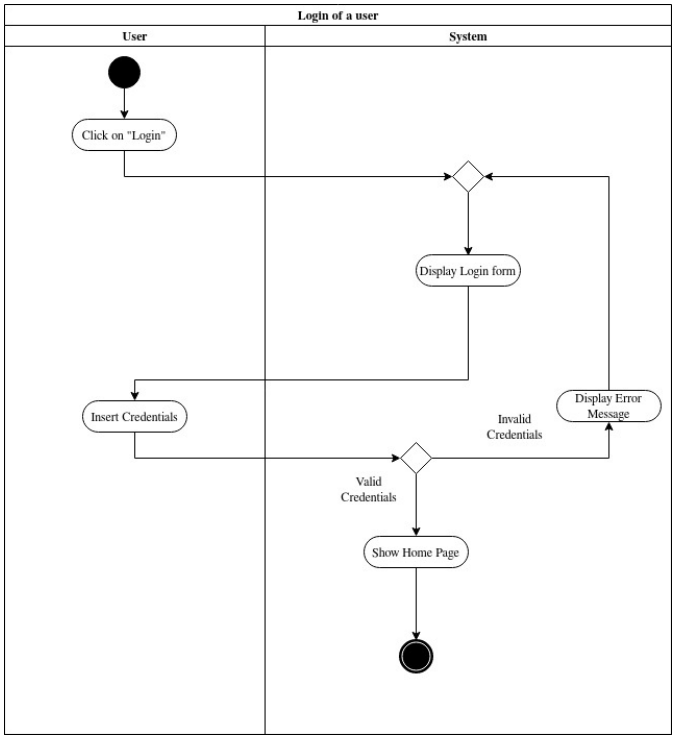

Figure 3.6: Activity diagram: User Login

Activity Diagram: Data Insertion

The following activity diagrams describe how farmers and agronomists insert data inside the system. In particular, the two diagrams map the use cases 1.9 and 2.4 respectively.

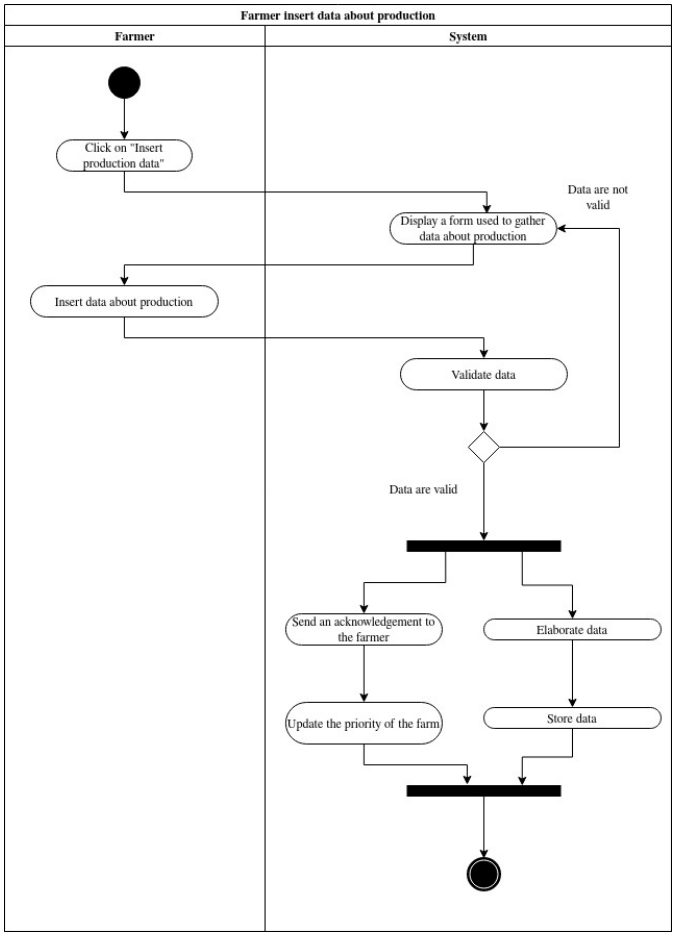

Figure 3.7: Activity diagram: Farmer inserts data about the production

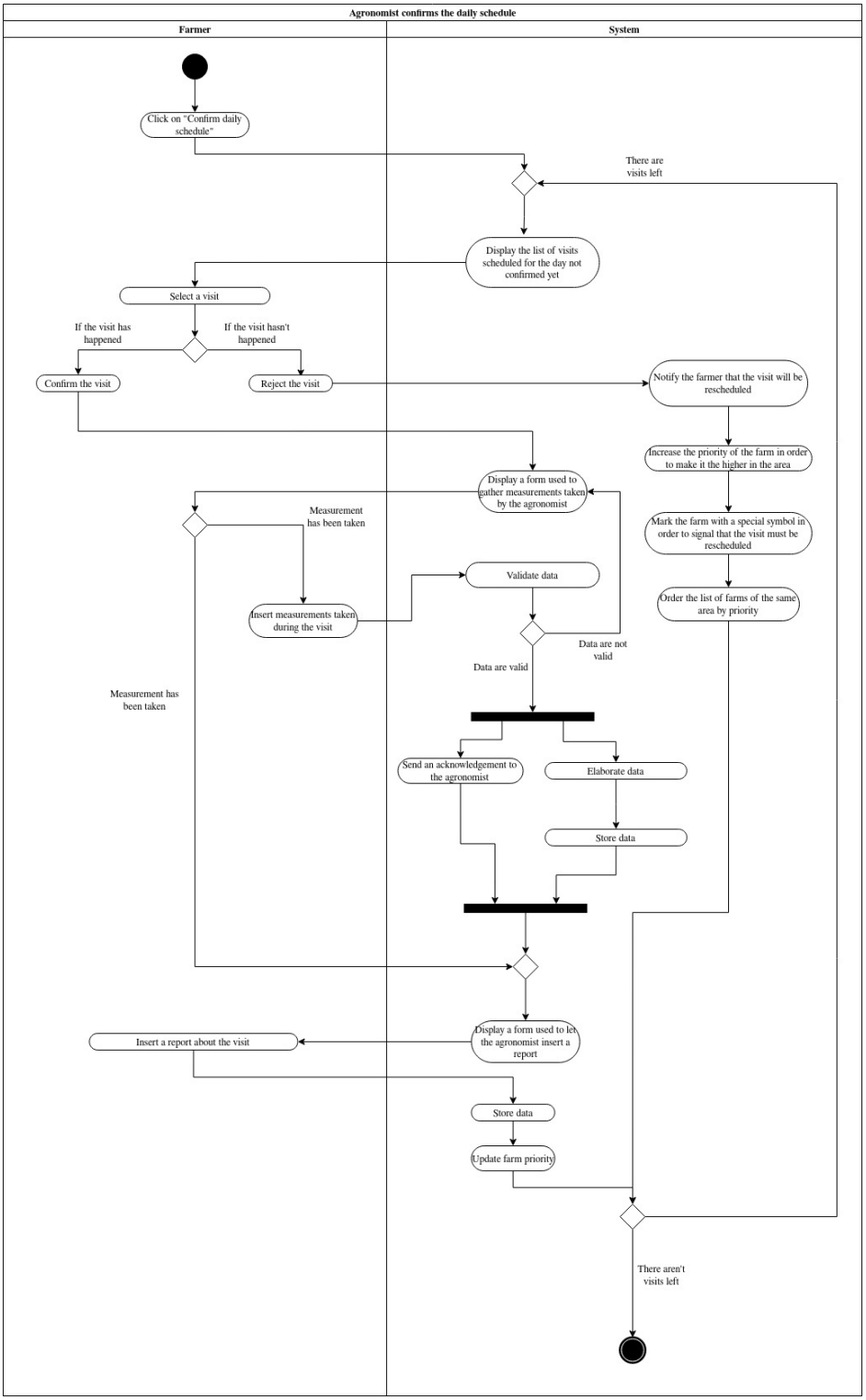

Figure 3.8: Activity diagram: Agronomist inserts data regarding a visit to a farm

Activity Diagram: Visualize Data

The following activity diagrams shows how users can visualize data of interest by using the system. In particular, the diagram maps the use case 0.2.

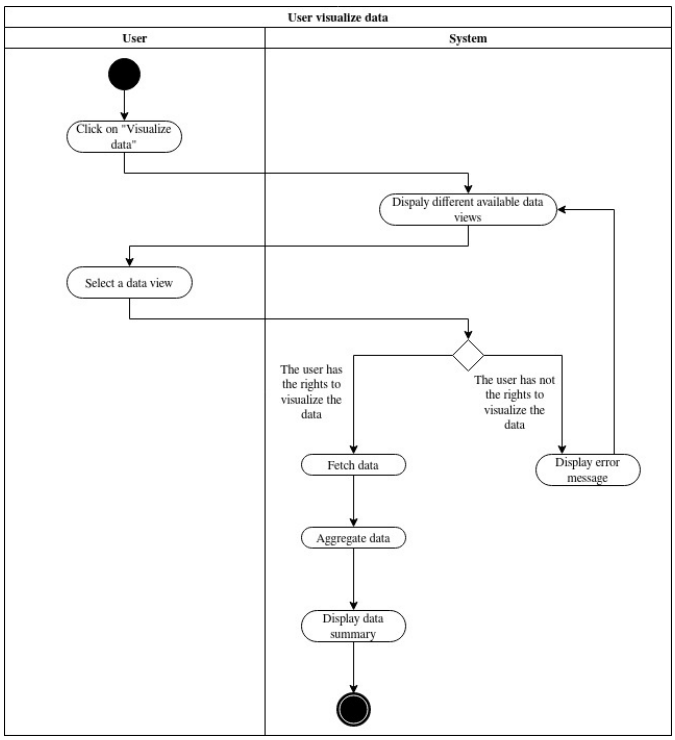

Figure 3.9: Activity diagram: User visualizes data summary

Activity Diagram: Chat between agronomists and farmers

The following activity diagram describes how agronomists and farmer can communicate by using the chat functionality offered by the system. In particular it maps the use cases 1.2, 2.2 and 1.6.

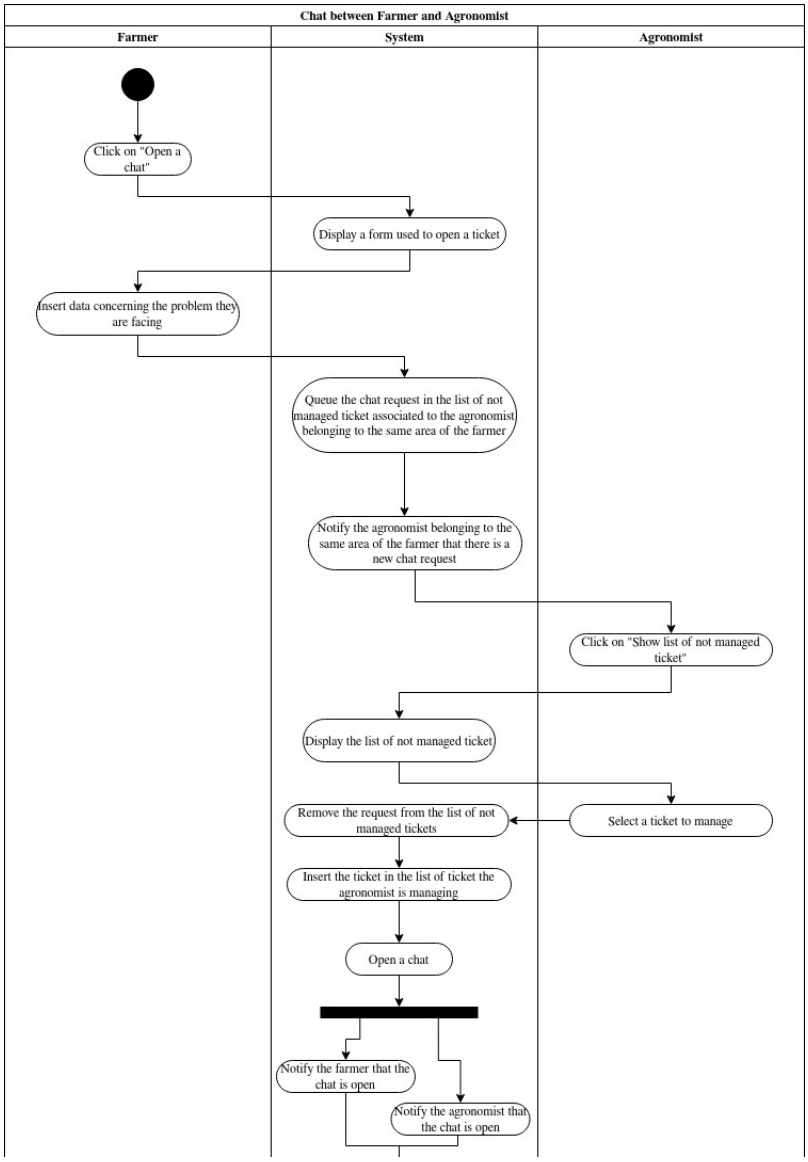

Figure 3.10: Activity diagram: Chat between agronomists and farmers (part 1)

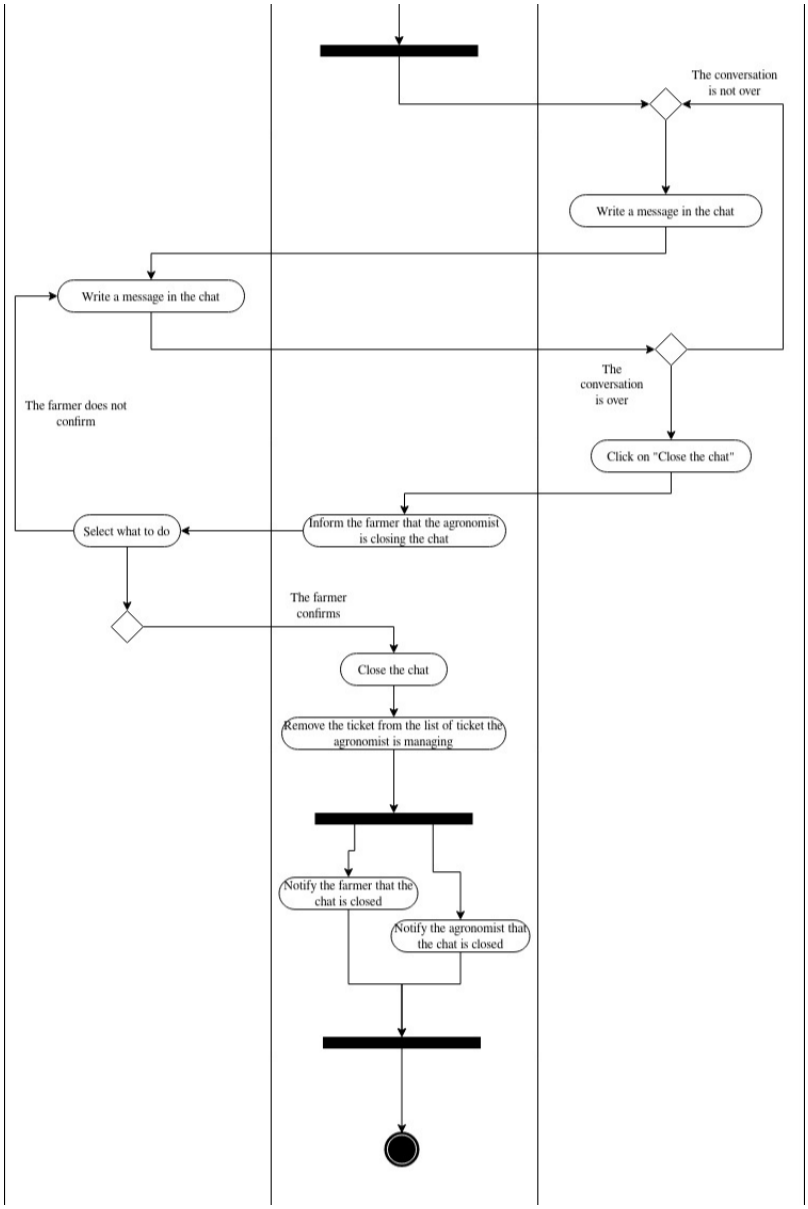

Figure 3.11: Activity diagram: Chat between agronomists and farmers (part 2)

Activity Diagram: Booking of visit

The following activity diagrams describes how a farmer can request a visit by an agronomist and how an agronomist can schedule a visit to a farm respectively. In particular the two diagrams map the use cases 1.3, 1.4, 1.5, 2.3 and 2.5.

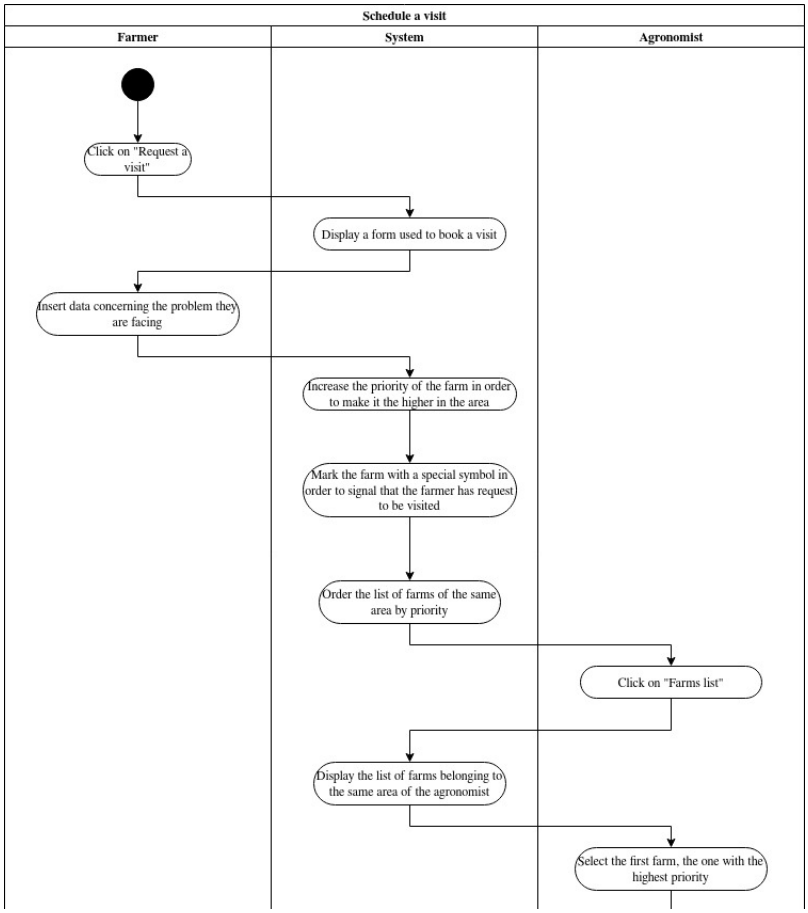

Figure 3.12: Activity diagram: Farmer requests a visit by an agronomist (Part 1)

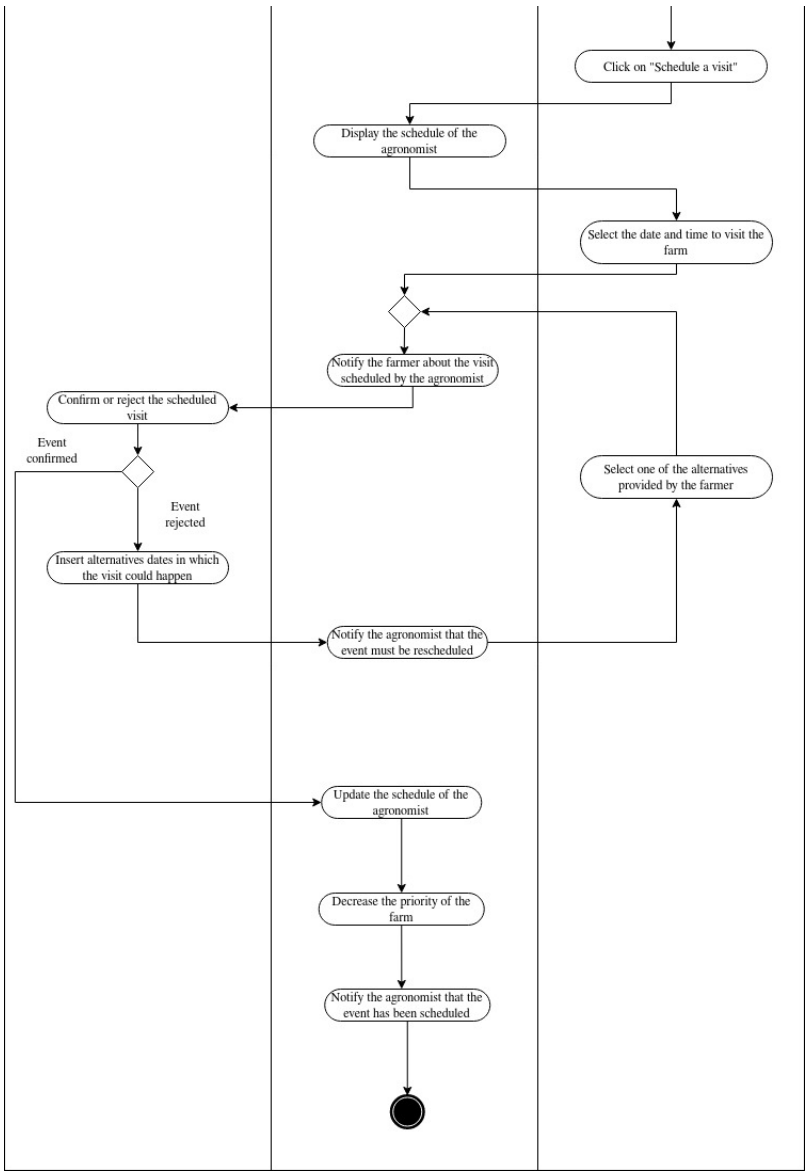

Figure 3.13: Activity diagram: Farmer requests a visit by an agronomist (Part 2)

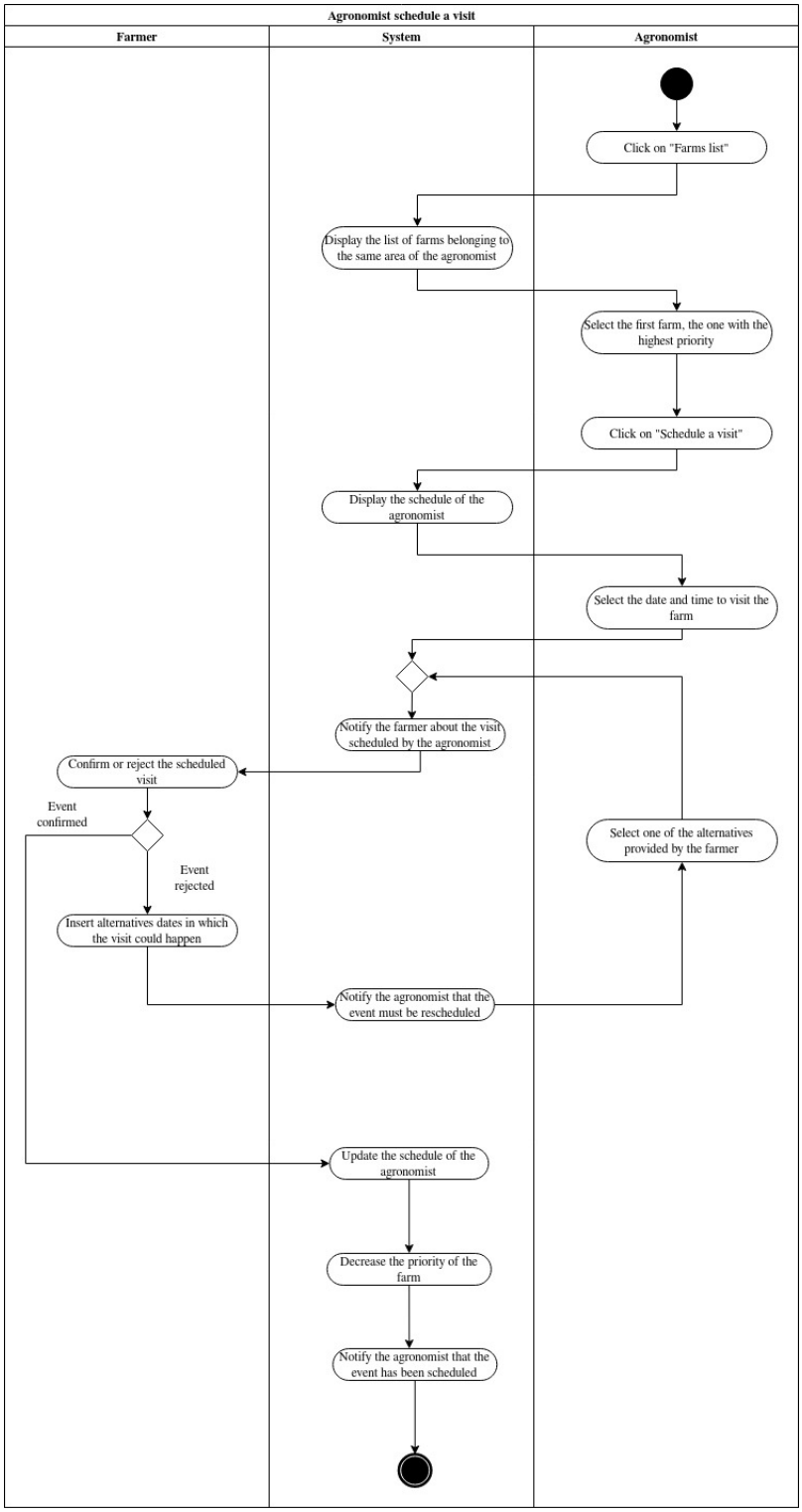

Figure 3.14: Activity diagram: Agronomist schedules a visit to a farm

3.3.7 Mapping between Goals, Domain Assumptions and Requirements

In the following will be shown which requirements and domain assumptions are needed ti fulfill each goals.

G1 - Data Driven approach

*R*01 The system allows users to sign up

*R*02 Every user can access the system only if they are registered

*R*03 The system must allow to access the system only if the credentials used are correct

*R*04 The system must not allow to register two different users with the same username

*R*05 The system must not allow the same user to register themselves more than once

*R*19 The system must fetch data from different sources containing relevant information for the scope of the application

*R*20 The system must elaborate and aggregate data fetched from different sources

*D*02 Every user has an internet connection in order to interact with the system

*D*03 Every farmer has the capability to interact with the system

*D*04 Every agronomist has the capability to interact with the system

*D*06 Every farmer has an electronic device in order to interact with the system

*D*07 Every agronomist has an electronic device in order to interact with the system

*D*09 The farmers insert true and correct data about their production

*D*10 The agronomists insert true and correct data about measurements

*D*15 All those agronomist that take measurements do so in a consistent way

*D*17 Only real and capable agronomists are allowed to subscribe to the system

*D*24 External databases are correctly and periodically updated

G2 - Help policy makers to implement better policies which will make Telangana's agriculture better

*R*01 The system allows users to sign up

*R*02 Every user can access the system only if they are registered

*R*03 The system must allow to access the system only if the credentials used are correct

*R*04 The system must not allow to register two different users with the same username

*R*05 The system must not allow the same user to register themselves more than once

*R*16 The system allows the policy maker to see data about a specific farm

*R*17 The system allows the policy maker to see data about an area

*R*18 The system allows the policy maker to see data about all the areas in Telangana

*R*19 The system must fetch data from different sources containing relevant information for the scope of the application

$R20$ The system must elaborate and aggregate data fetched from different sources

$D02$ Every user has an internet connection in order to interact with the system

$D03$ Every farmer has the capability to interact with the system

$D04$ Every agronomist has the capability to interact with the system

$D05$ Every policy maker has the capability to interact with the system

$D06$ Every farmer has an electronic device in order to interact with the system

$D07$ Every agronomist has an electronic device in order to interact with the system

$D08$ Every policy maker has an electronic device in order to interact with the system

$D09$ The farmers insert true and correct data about their production

$D10$ The agronomists insert true and correct data about measurements

$D11$ The reports inserted by agronomists are sufficiently detailed

$D15$ All those agronomist that take measurements do so in a consistent way

$D18$ Only real and capable policy maker are allowed to subscribe to the system

$D22$ Every user has its own credentials

$D23$ The credentials used to access the system are not shareable

$D24$ External databases are correctly and periodically updated

G3 - Help farmers to become weather resilient and increase their production

$R01$ The system allows users to sign up

$R02$ Every user can access the system only if they are registered

$R03$ The system must allow to access the system only if the credentials used are correct

$R04$ The system must not allow to register two different users with the same username

$R05$ The system must not allow the same user to register themselves more than once

$R10$ The system allows the farmer to see data about their farm

$R11$ The system must not allow the farmer to see data about farms other than theirs

$R19$ The system must fetch data from different sources containing relevant information for the scope of the application

$R20$ The system must elaborate and aggregate data fetched from different sources

$D02$ Every user has an internet connection in order to interact with the system

$D03$ Every farmer has the capability to interact with the system

$D04$ Every agronomist has the capability to interact with the system

$D06$ Every farmer has an electronic device in order to interact with the system

$D07$ Every agronomist has an electronic device in order to interact with the system

$D09$ The farmers insert true and correct data about their production

$D10$ The agronomists insert true and correct data about measurements

$D15$ All those agronomist that take measurements do so in a consistent way

$D16$ The farmers follows agronomist's advice

$D17$ Only real and capable agronomists are allowed to subscribe to the system

$D20$ Each farmer's account owns one and only one farm

$D21$ Each farm is owned by one and only one farmer

$D22$ Every user has its own credentials

$D23$ The credentials used to access the system are not shareable

$D24$ External databases are correctly and periodically updated

G4 - Facilitate and make systematic the communication between farmers and agronomists

$R01$ The system allows users to sign up

$R02$ Every user can access the system only if they are registered

$R03$ The system must allow to access the system only if the credentials used are correct

$R04$ The system must not allow to register two different users with the same username

$R05$ The system must not allow the same user to register themselves more than once

$R06$ The system shall associate to each area one or more agronomists

$R07$ The system shall associate each farmer to one and only one area

$R08$ Each agronomist is associated to one and only one area

$R09$ The system allows the agronomist to see only the list of farms for which they are responsible of

$R10$ The system allows the farmer to see data about their farm

$R11$ The system must not allow the farmer to see data about farms other than theirs

$R12$ The system allows the agronomist to see data about a specific farm

$R13$ The system allows the agronomist to see data about the area for which they are responsible for

$R14$ The system must not allow the agronomist to see data about farms for which they are not responsible for

$R15$ The system must not allow the agronomist to see data about an area for which they are not responsible for

$R19$ The system must fetch data from different sources containing relevant information for the scope of the application

$R20$ The system must elaborate and aggregate data fetched from different sources

$R21$ The system allows farmers to request to have a chat with an agronomist

$R22$ The system allows agronomists to accept chat requests

$R23$ The system allows farmers and agronomists to interact through a chat

$R24$ A farmer, belonging to an area, can chat only with an agronomist belonging to the same area

$R25$ Each farmer can have at most one open chat at any given time

$R26$ Each agronomist can have multiple chats open at any given time

$R27$ The system allows only agronomists to close chats

$R28$ The system allows farmers to request a visit from an agronomist

$R29$ The system must show an agronomist a list of farms they have to visit, ordering them by priority

$R30$ The system must prioritize the farms that need help

$R31$ The agronomist must prioritize the farms that the system prioritize

$R32$ The system allows agronomists to schedule a visit at any farm of their competence

$R33$ The system allows agronomist to set up a daily schedule

$R34$ The system must notify a farmer when an agronomist schedules a visit to their farm

$R35$ The system must notify an agronomist when a farmers cancels a scheduled visit to their farm

$R36$ The system must notify a farmer when an agronomist cancels a scheduled visit to their farm

$R37$ The system allows agronomists to cancel a scheduled visits within one day before

$D01$ There are enough agronomists for each area

$D02$ Every user has an internet connection in order to interact with the system

$D03$ Every farmer has the capability to interact with the system

$D04$ Every agronomist has the capability to interact with the system

$D06$ Every farmer has an electronic device in order to interact with the system

$D07$ Every agronomist has an electronic device in order to interact with the system

$D09$ The farmers insert true and correct data about their production

$D10$ The agronomists insert true and correct data about measurements

$D12$ The agronomists always confirms all the visits that they have done, sooner or later

$D13$ A visit in an agronomist schedule is confirmed only if it is actually done by the agronomist

$D14$ All the agronomists have at least decent capabilities of managing and organizing their own schedule

$D15$ All those agronomist that take measurements do so in a consistent way

$D16$ The farmers follows agronomist's advice

$D17$ Only real and capable agronomists are allowed to subscribe to the system

$D19$ The agronomist that wants to visit a farm will reach an agreement with the owner farmer on the day and date of the appointment within about 14 days from the first appointment request

$D20$ Each farmer's account owns one and only one farm

$D21$ Each farm is owned by one and only one farmer

$D22$ Every user has its own credentials

$D23$ The credentials used to access the system are not shareable

$D24$ External databases are correctly and periodically updated

G5 - Create a network of farmers

$R01$ The system allows users to sign up

$R02$ Every user can access the system only if they are registered

$R03$ The system must allow to access the system only if the credentials used are correct

$R04$ The system must not allow to register two different users with the same username

$R05$ The system must not allow the same user to register themselves more than once

$R38$ The system shall allow farmers to interact through a forum with other farmers and agronomists

$R39$ The system shall allow agronomists to interact through a forum with other agronomists and farmers

$R40$ The forum allow to each farmer or agronomist with each other no matter the area they belong to

$D02$ Every user has an internet connection in order to interact with the system

$D03$ Every farmer has the capability to interact with the system

$D04$ Every agronomist has the capability to interact with the system

$D06$ Every farmer has an electronic device in order to interact with the system

$D07$ Every agronomist has an electronic device in order to interact with the system

$D22$ Every user has its own credentials

$D23$ The credentials used to access the system are not shareable

3.4 Performance requirements

The response time is not the main focus of our system. In fact, scalability surely has more impact in the system: we need to support a lot of different areas in Telangana, which include several farmers and agronomists. Furthermore, in the future the system may be adopted by more regions of India, and even by different countries. The system must be able to process a large amount of data, including data fetched periodically from different external databases that will be elaborated in real-time, allowing users to access it.

3.5 Design constraints

3.5.1 Standard compliance

The system described will be used by the government, therefore it must be compliant with the Digital Public Goods (DPG) standard.
The Digital Public Goods Standard is a set of specifications and guidelines designed to maximise consensus about whether a digital solution conforms to the definition of a digital public good: open-source software, open data, open AI models, open standards, and open content that adhere to privacy and other applicable best practices, do no harm by design and are of high relevance for attainment of the United Nations 2030 Sustainable Development Goals (SDGs).

More information can be found here: https://digitalpublicgoods.net/standard/

3.5.2 Hardware Constraints

Each person interested in using the system described in this document must have a device connected to the Internet, which could be a smartphone, a tablet or a personal computer. In particular, the basic requirement is to have an Internet browser installed on the device. There is then the possibility of downloading and installing the official DREAM application from the most important stores, such as *Apple's App Store* or *Google's Play Store* or *Huawei's AppGallery*.

3.6 Software system attributes

3.6.1 Reliability

The system must prevent downtime. In fact, agronomists' work depend on the functioning of the system, and any downtime could make the schedule of each agronomist shift or could prevent users to insert useful data in the system.

3.6.2 Availability

The system must be available as much as possible. In fact, because the system could be adopted worldwide, and therefore could be used in different time zones, the system must be available 24 hours per day every day of the year.

3.6.3 Security

Communication between parties are encrypted and goes on a secure channel (through SSL protocol). Furthermore, the database guarantees that performed operations are always authorized.

3.6.4 Maintainability

The system must be designed in such a way that permits future addition of functionalities with minimum effort. For example, new machine learning modules could be needed in the future as long as agriculture techniques evolve. Moreover, the rising of 5G could allow the adoption of more complex sensors systems that could be included in the system.

3.6.5 Portability

The system must be supported on the principal operating systems, either mobile and computers. It is in any case usable on every web browser, from every device.

3.6.6 Scalability

The scalability of the system must be an absolute priority. In fact, in the future the system may be adopted by more regions of India other than Telangana, and even by more states spread around the world.

3.6.7 Usability

The system should be very easy to use and intuitive. Indeed, potentially, it could be used by a large spectrum of people (e.g. from farmers to policy makers) having different backgrounds and educations.

Chapter 4

Formal analysis using Alloy

We want to formalize our application by creating a model with Alloy. For this reason, we describe what are the main topics that we want to address in the formal analysis:

- Farmers and Agronomists are organized into geographical areas. We want to prove that they can interact with each other, through different mechanisms, only if they belong to the same area
- Each Agronomist has a schedule of visits to the farms: when a schedule is confirmed and inserted into the system they must insert a report about the visits they have done
- The policy makers must see everything that allows them to understand how both the farmers and the agronomists are performing. We will show that they are able to see data about production of the farms along with report generated by the agronomists

4.1 Alloy Code

```
abstract sig User {}

sig PolicyMaker extends User{}

sig Agronomist extends User{
    agronomistArea: one Area,
    agronomistSchedule: seq DailySchedule
}


sig Farmer extends User{
    ownedFarm: Farm,
}

sig Date{
    day: Int,
    month: Int,
    year: Int,
}{
    day     > 0 and day ≤ 31
    month   > 0 and month ≤ 12
    year    > 0
}

sig Time{
    hour: Int,
    minute: Int
}{
    hour    > 0
    minute  > 0
}

abstract sig Boolean{}
one sig True extends Boolean{}
```

```
one sig False extends Boolean{}

sig Data{}

sig GeoPosition{}

sig Farm{
    farmOwner:Farmer,
    farmArea:Area,
    farmScope: some User,
    productionData: some Data
}

sig Area{
    areaFarms: some Farm,
    areaAgronomists: some Agronomist,
    location:GeoPosition
}

sig Chat{
    chatFarmer: Farmer,
    chatAgronomist: lone Agronomist,
    chatArea: Area,
    chatScope: some User,
    chatState: ChatState
}

sig Report{
    reportAgronomist: Agronomist,
    reportVisit: Visit,
    reportData: set Data,
    reportScope: some User
}

sig Visit{
    visitAgronomist: Agronomist,
    visitFarm: Farm,
    visitDate: Date,
    visitTime: Time,
    visitScope: some User,
    visitCompleted: Boolean
}

sig DailySchedule{
    schedVisit: some Visit,
    schedDate: one Date,
    schedExamined: Boolean
}

one sig Forum{
    forumThreads: set Thread
}

sig Thread{
    threadCreator: one User,
    threadScope: some User
}

abstract sig ChatState{}
one sig Pending extends ChatState {}
one sig Opened extends ChatState {}
one sig ClosedProposed extends ChatState {}
one sig Closed extends ChatState {}

//-----------------------------------------------------------------
// FACTS

fact AreasAreDisjoint{
    no disj a1,a2:Area |
        one f:Farm |
            f in a1.areaFarms
            and
            f in a2.areaFarms
}

fact AreasHaveDifferentLocations{
    no disj a1,a2:Area |
        a1.location = a2.location
}
```

```
fact{ //Two-way relation
    all o:Farmer, f:Farm |
        f in o.ownedFarm iff f.farmOwner = o
}

fact{ //Two-way relation
    all ag:Agronomist, ar:Area |
        ag in ar.areaAgronomists iff ag.agronomistArea = ar
}

fact{ //Two-way relation
    all f:Farm, ar:Area |
        f in ar.areaFarms iff f.farmArea = ar
}

fact { //Agronomists can visit only farms that are in their area
    all v:Visit |
        v.visitAgronomist.agronomistArea = v.visitFarm.farmArea
}

fact {  //Agronomists can deal with a chat request only if the request is coming
        //from a farm that is in their area
    all c:Chat |
        c.chatState ≠ Pending
        implies
        c.chatFarmer.ownedFarm.farmArea = c.chatAgronomist.agronomistArea
}

fact { //report data and production data are different
    all r:Report, f:Farm |
        #(r.reportData & f.productionData) = 0

}

fact ChatArea{
    all c:Chat |
        c.chatFarmer.ownedFarm.farmArea = c.chatArea
}

fact reportAgronomist{
    all r:Report |
        r.reportAgronomist = r.reportVisit.visitAgronomist
}

fact farmScope{
    all f:Farm, pm:PolicyMaker |
        pm in f.farmScope

    all f:Farm, ag:Agronomist |
        ag in f.farmScope
        iff
        f.farmArea = ag.agronomistArea

    all f:Farm, f1:Farmer |
        f1 in f.farmScope
        iff
        f1 = f.farmOwner
}

fact threadScope{
    all t:Thread, u:User | u in t.threadScope
}

fact reportScope{
    all r:Report, pm:PolicyMaker |
        pm in r.reportScope

    all r:Report, a:Agronomist |
        a in r.reportScope
        iff
        r.reportAgronomist = a

    all r:Report |
        no f:Farmer |
            f in r.reportScope
}

fact chatScope{
    all c:Chat |
        c.chatState = Pending
```

```
        implies
        #c.chatAgronomist = 0

    all c:Chat |
        c.chatState = Pending
        implies
        c.chatScope = c.chatFarmer + c.chatArea.areaAgronomists

    all c:Chat |
        c.chatState = Opened
        implies
        c.chatScope = c.chatFarmer + c.chatAgronomist

    all c:Chat |
        c.chatState = Closed
        implies
        c.chatScope = c.chatFarmer + c.chatAgronomist

    all c:Chat |
        c.chatState = ClosedProposed
        implies
        c.chatScope = c.chatFarmer + c.chatAgronomist
}

fact visitScope{
    all v:Visit, pm:PolicyMaker |
        pm in v.visitScope

    all v:Visit |
        v.visitAgronomist in v.visitScope

    all v:Visit |
        v.visitFarm.farmOwner in v.visitScope
}

fact dailySchedule{
    //Visits in the daily schedule have the same date of the daily schedule
    all ds:DailySchedule, v:Visit |
        v in ds.schedVisit
        implies
        v.visitDate = ds.schedDate

    //schedExamined means that the agronomist has "confirmed"
    //the daily schedule in the system: so a report must be inserted
    all ds:DailySchedule, v:Visit | ds.schedExamined = True
        implies
            (v in ds.schedVisit and v.visitCompleted = True
            implies
            one r:Report | r.reportVisit = v)

    // There cannot be two unexamined schedules:
    // the agronomist has to confirm their schedule
    // at most at the end of the day!
    all ag:Agronomist, ds:DailySchedule |
        (ds in ag.agronomistSchedule.elems and ds.schedExamined = False )
        implies
        ds = ag.agronomistSchedule.last

    // There are no daily schedules belonging to nobody
    all ds:DailySchedule | some ag:Agronomist | ds in ag.agronomistSchedule.elems

    // Each daily schedule is personal
    no disj ag1, ag2: Agronomist | ag1.agronomistSchedule = ag2.agronomistSchedule

    all ds:DailySchedule|
        no disj ag1, ag2: Agronomist|
            ( ds in ag1.agronomistSchedule.elems)
            and
            ( ds in ag2.agronomistSchedule.elems)


    all  ag:Agronomist | !ag.agronomistSchedule.hasDups
}

fact visitAgronomist{
    // If there is a visit, it is inserted in one of the agronomists'schedule
    all v:Visit, ag:Agronomist |
        ag = v.visitAgronomist
        iff
        v in ag.agronomistSchedule.elems.schedVisit
```

```
}

fact{
    //There cannot be two different visit that happened in the same instant
    all ag:Agronomist |
        no disj v1,v2 :Visit |
            (v1 in ag.agronomistSchedule.elems.schedVisit
            and v1.visitCompleted = True)
            and
            (v2 in ag.agronomistSchedule.elems.schedVisit
            and v2.visitCompleted = True)
            and
            (v1.visitDate = v2.visitDate
            and v1.visitTime = v2.visitTime)
}

fact{
    //Each daily schedule is in fact unique for each day
    //Since the schedule of the agronomist is a sequence of daily schedules
    //the visits in the daily schedules reflect the same order
    all ag: Agronomist |
        no disj ds1, ds2 : DailySchedule |
            ds1 in ag.agronomistSchedule.elems
            and
            ds2 in ag.agronomistSchedule.elems
            and
            ds1.schedDate = ds2.schedDate

    all ag:Agronomist, ds1, ds2:DailySchedule |
        (ds1 in ag.agronomistSchedule.elems
        and
        ds2 in ag.agronomistSchedule.elems
        and
        ag.agronomistSchedule.idxOf[ds2] > ag.agronomistSchedule.idxOf[ds1])
        implies
        isSuccDate[ds1.schedDate, ds2.schedDate]

}

fact {
    // A report can be generated only if a visit has been completed
    all r:Report, v:Visit | v in r.reportVisit iff v.visitCompleted = True
}

//----------------------------------------------------------------
// PREDICATES

pred isSuccDate[d1:Date, d2:Date]{
    d2.year > d1.year
    or
    (
        d2.year = d1.year
        and
        d2.month > d1.month
    )
    or
    (
        d2.year = d1.year
        and
        d2.month = d1.month
        and
        d2.day > d1.day
    )
}

pred show {
    #User > 2
    #Chat = 1
    #Agronomist = 1
    #PolicyMaker = 1
    #Farmer = 2
    #Visit  = 1
    #Thread = 0
    #DailySchedule = 1

}
run show for 5 but 6 int

pred scheduleExamined[ds:DailySchedule, v:Visit]{
    //This predicate shows that if a schedule has been examined
```

```
    //visits that have been completed have an associated report
    ds.schedExamined = True
    v in ds.schedVisit
    v.visitCompleted = True
    #Visit > 2
}

run scheduleExamined for 3 but 6 int

pred showScope
    [disj a1,a2:Area,ds:DailySchedule, v:Visit, disj f1,f2:Farm,
    disj agr1, agr2:Agronomist, pm1:PolicyMaker]
{
    //This predicate shows what can be seen by the different Users
    f1.farmArea = a1
    f2.farmArea = a2
    agr1.agronomistArea = a1
    agr2.agronomistArea = a2
    ds.schedExamined = True
    v in ds.schedVisit
    v.visitCompleted = True
    ds in agr1.agronomistSchedule.elems

}

run showScope for 5 but 6 int

pred chatPending[c:Chat, f:Farmer, disj a1,a2:Area,
disj ag1,ag2,ag3:Agronomist,state:Pending]{
    c.chatFarmer = f
    c.chatState = state
    c.chatArea = a1
    f.ownedFarm.farmArea = a1
    ag1.agronomistArea = a1
    ag2.agronomistArea = a1
    ag3.agronomistArea = a2

}
run chatPending for 5 but 6 int


pred newVisit[v:Visit, ag1:Agronomist, ds:DailySchedule]{
    #Chat = 0
    ds in ag1.agronomistSchedule.elems
    v.visitAgronomist = ag1
    ds.schedVisit = ds.schedVisit + v
}

run newVisit for 5 but 6 int


//---------------------------------------------------------------
// ASSERTIONS

assert OneOwner {
   no disj f1,f2: Farmer | f1.ownedFarm = f2.ownedFarm
}
check OneOwner

assert AgronomistsRespectArea{
    no v:Visit |
        v.visitAgronomist.agronomistArea ≠ v.visitFarm.farmArea
    and
    no c:Chat |
        c.chatAgronomist.agronomistArea ≠ c.chatFarmer.ownedFarm.farmArea
}
check AgronomistsRespectArea for 5 but 6 int

assert AgronomistOperatingArea{
    all ag:Agronomist, ds1, ds2 : DailySchedule|
        no disj v1, v2: Visit |
            ds1 in ag.agronomistSchedule.elems
            and
            ds2 in ag.agronomistSchedule.elems
            and
            v1 in ds1.schedVisit
            and
            v2 in ds2.schedVisit
            and
            v1.visitFarm.farmArea ≠ v2.visitFarm.farmArea
```

```
    no disj ag1,ag2:Agronomist, ds1:DailySchedule|
        all v:Visit |
            ds1 in ag1.agronomistSchedule.elems
            and
            v in ds1.schedVisit
            and
            v.visitAgronomist = ag2

}

check AgronomistOperatingArea for 5 but 6 int


assert PolicyMakerSeeEveryFarm{
    all pm:PolicyMaker, f:Farm |
        pm in f.farmScope
    all pm:PolicyMaker, r:Report |
        pm in r.reportScope
    all pm:PolicyMaker, v:Visit |
        pm in v.visitScope
    all pm:PolicyMaker, t:Thread |
        pm in t.threadScope

}
check PolicyMakerSeeEveryFarm for 5 but 6 int
```

4.2 Alloy Worlds

4.2.1 Farmer requests a chat with an Agronomist

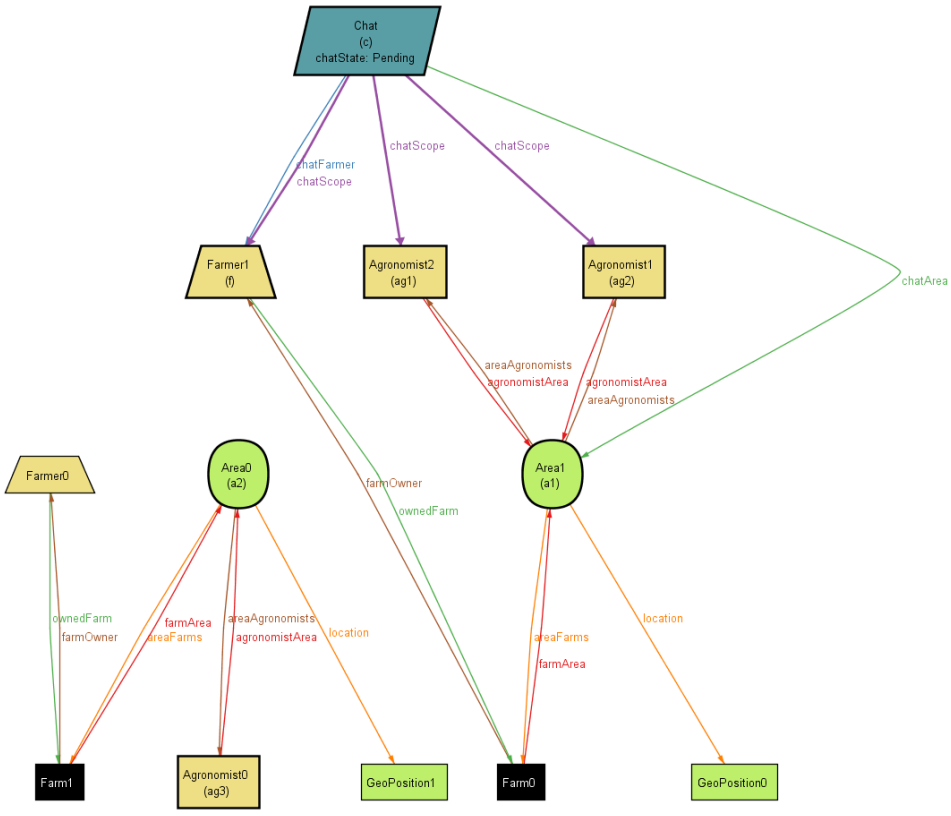

Figure 4.1: pred chatPending

This diagram shows a Farmer (f) that has opened a chat (c). The just opened chat is in state "pending" meaning that no Agronomists took it yet. There are two Agronomists (ag1,ag2) that belong to the same area (a1) of the Farmer's farm. Furthermore, there is another area (a2) and another Agronomist (ag3) in this model. As we can see the chat scope is correctly related only to ag1 and ag2. On the other hand, ag3 that belongs to a different area, a2, cannot see the pending chat.

4.2.2 Agronomist schedule a visit to a Farm

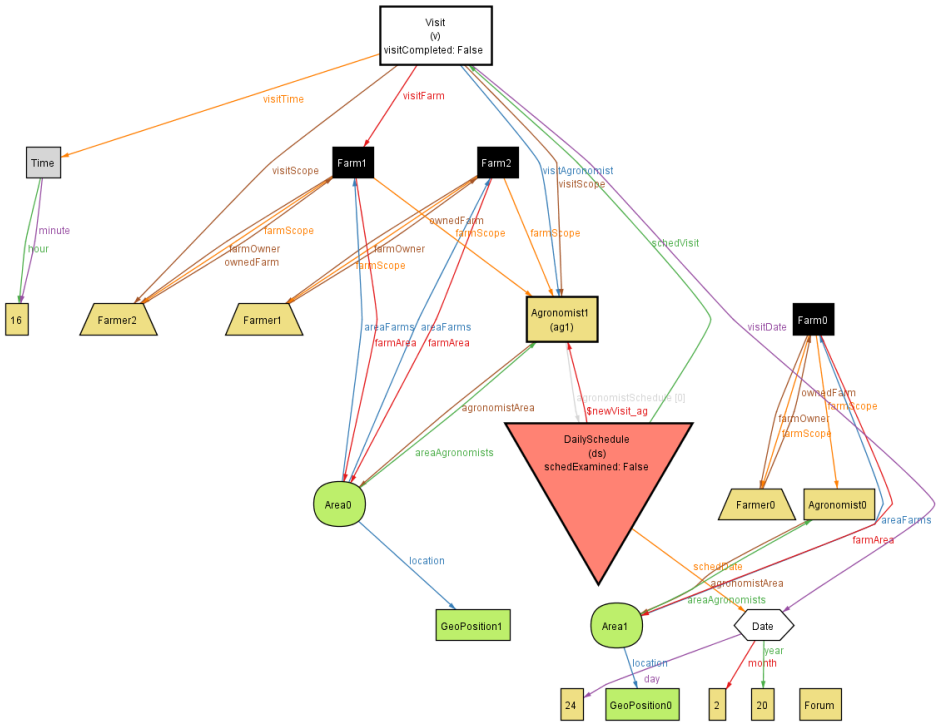

Figure 4.2: pred newVisit

The diagram shows that the Agronomist (ag1) has updated one of their daily schedules to take into account a new visit (v). The visit is set to be to Farm1, that belongs to Area0, the same of the Agronomist (ag1). We can notice that in this instance of our model, there is another area, Area1, that is completely separated from Area0. In fact, ag1 can see Farm1 and Farm2, not Farm0. Furthermore, we can notice how Farmer2, which is the owner of the Farm1, the one that is addressed in the new visit (v), can see it; Farmer0 and Farmer1 instead, do not have the appropriate visitScope relation.

4.2.3 Agronomist confirms and inserts a daily schedule into the system

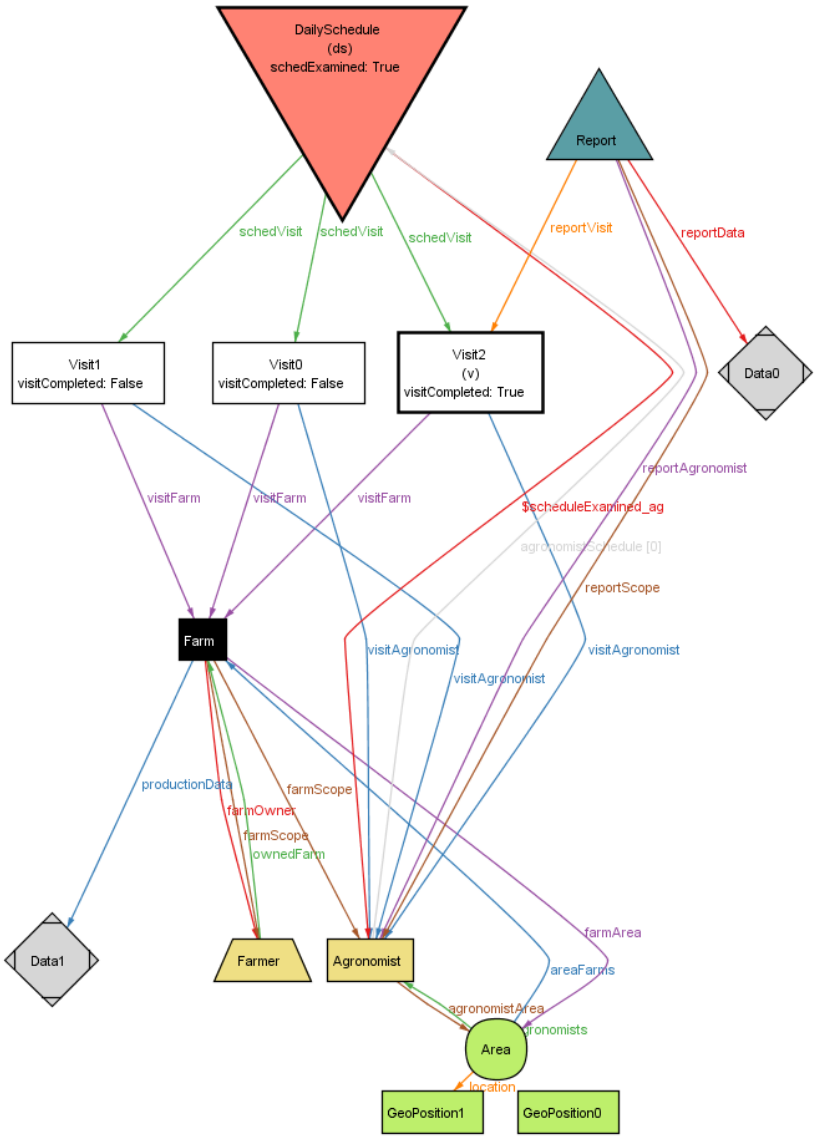

Figure 4.3: pred scheduleExamined

The diagram shows that the Agronomist present has a DailySchedule (ds) which has been "examined". It means that the Agronomist has confirmed and inserted the daily schedule into the system. As we can see, since the daily schedule (ds) has a visit that was completed, the Agronomist has also produced a Report of such visit (v). Notice that, since Visit0 and Visit1 were not completed, even if the schedule was confirmed no associated reports have been produced.

4.2.4 An interesting and general instance of the model

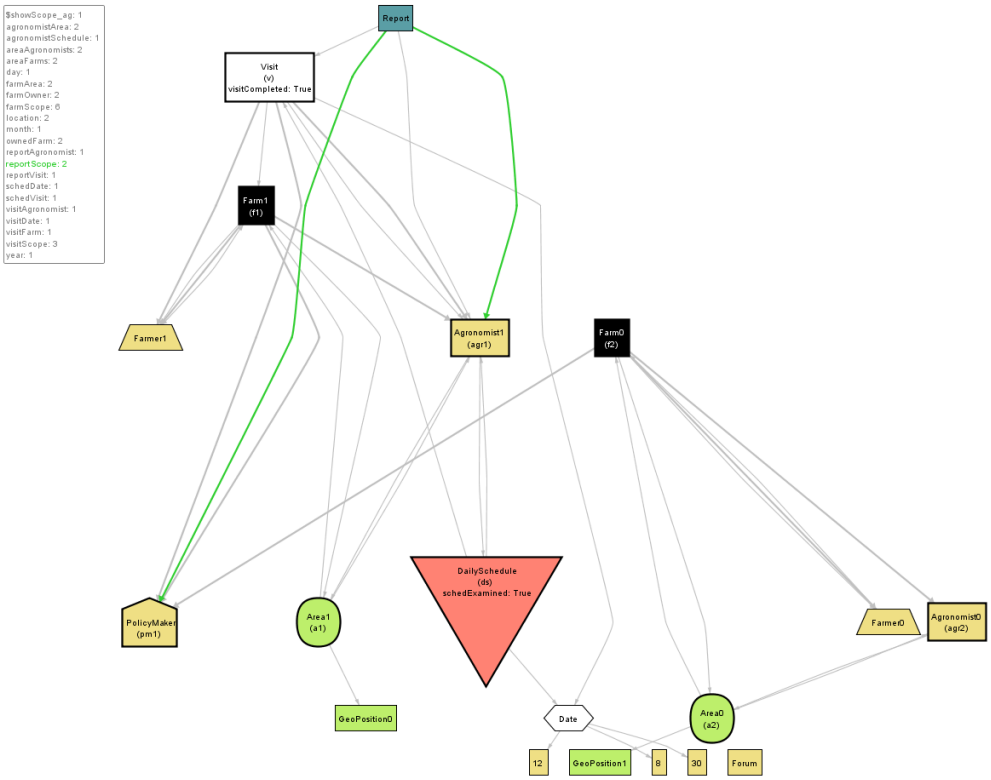

Figure 4.4: pred showScope Report

In this instance we have an Agronomist (ag1) that has produced a report of a visit (v). As we can see from the green arrows, the report is visible to (ag1) that produced it, and to the policy maker that is present in this instance (pm1).

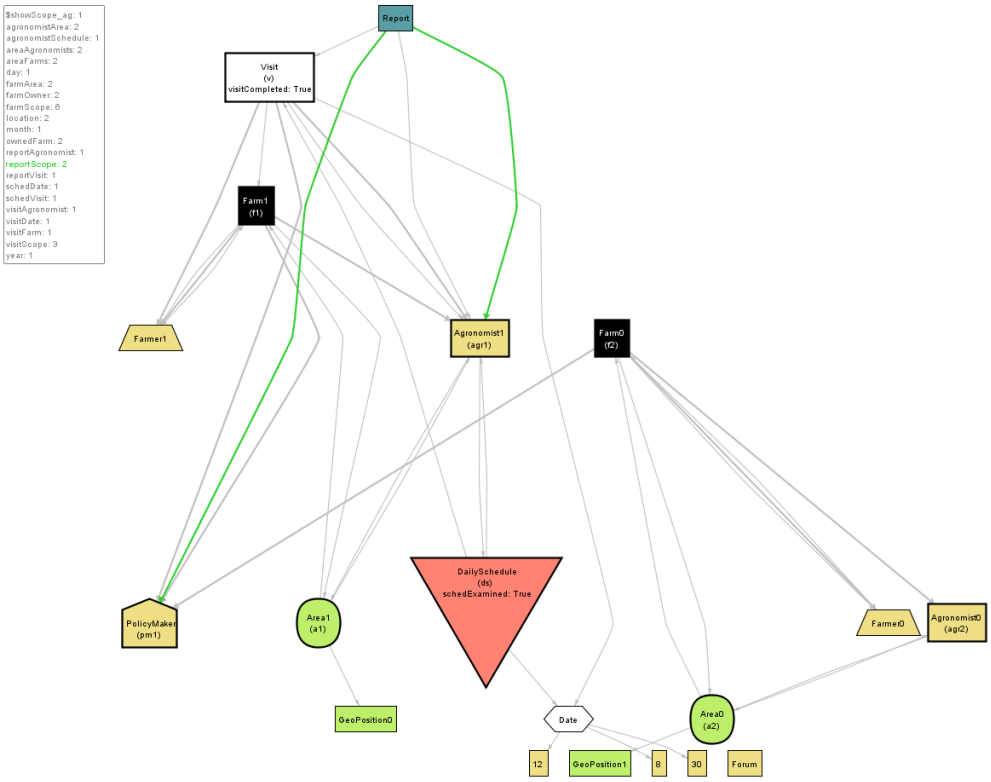

Figure 4.5: pred showScope Farm

The light red arrows instead specify the visibility of the Farms: while for the farmers and the agronomist the visibility is limited with respect to the area of belonging, the Policy Maker (pm1) can see both of the farms present, independently from their area.

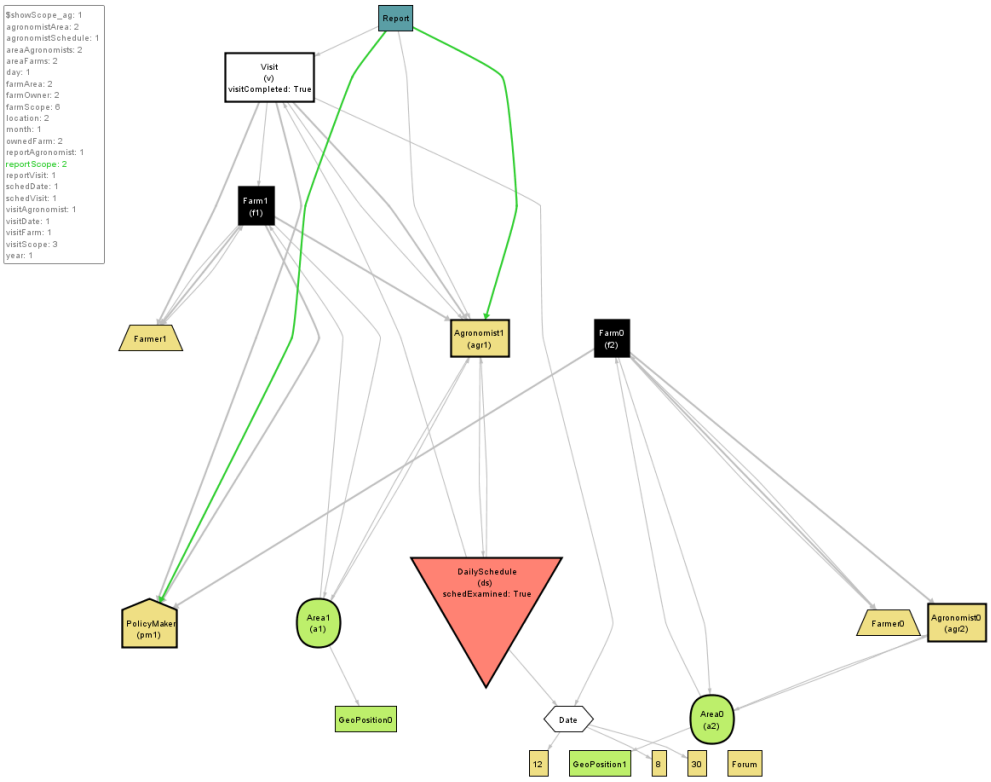

Figure 4.6: pred showScope Visit

The yellow arrows specify the visibility of the visit, completed, (v). Thus, we can see, differently from the previous diagram (4.2.2) that here there is also a Policy Maker (pm1) that can see the visit (v).

Chapter 5

Effort spent

Student	Effort
Salvatore Maccarrone	55h
Massimiliano Pellizzer	55h
Andrea Prati	55h

5.1 Thought process

The drafting process of this document started on October 20, we had a few meetings where we read and tried to figure out the most important goals of the assignment.
We started writing down the phenomena and the Domain Assumption, then we wrote the requirements.
At this point we divided the sections of the document in order to work in parallel and try to avoid any kinds of ambiguity, actually we used an iterative writing process, everyone put something of their personal thoughts in each section, always keeping the whole document consistent as much as possible.
The sections have not been written in the order they appear in the last version of the document.

5.1.1 Comments about the assignment

The assignment was described with a certain level of ambiguity, thus we had to make some decisions that weren't specified in it.
We interpreted the assignment as a request to design a platform that should be a reference for all the Telangana's Farmers, Agronomists and Policy Makers, always considering that the system needs to have a strong Data Driven nature.
Therefore, we wrote a first version of DREAM, a platform that can be divided into three categories: Data, Community and Management. More details about these categories are written in the Section 1.2 of the document.

We started thinking about which data would have been produced through DREAM and which data would have been taken from external DBs.
Then we started designing the communication channels which would have allowed the users to talk with each other. Here we started to think about the Forum and the chat mechanism.
After having a first idea about the community, we thought about how the Agronomist and the Policy Maker would have been able to use the data stored into the system, and properly processed and aggregated, in order to acquire information about the live status of the farms on the territory.

As last thing we decided the method to use to keep track of the steering initiatives given to the farmers by the agronomists.

The interview with the stakeholders (the Q&A) was useful in order to have a better idea about the domain. Indeed, during this event the farmer's average digital competence was clarified, we discovered that in India most of the farmers have a basic digital competence, that is why we decided to assign them just the insertion about their harvests and the community features.

Ultimately, as requested by the stakeholders, we designed DREAM first and foremost as a platform, to which all the desired analytics can then be associated.

Looking at the GitHub repository that we listed among the references, we find in the README.md in the root of the repository an image that describes the data flow of the system: link.
We have used that image as a reference, but we have only considered the application aspect of that schema.
In addition, in the top left corner we find the merge into a single cloud data storage of some data, this aspect has not been described in this document because, from the interview with the stakeholders we considered more important to stress the nature of the platform itself, in particular, we note that we were told this:
"Please assume all data to be as the datasets that are given and available to you.
The data processing and analytics should not be the focus of the work; rather, the DREAM platform and its elaboration should be the major focus, perhaps designing in sight of specifying further analytics around what we discussed together and what is documented in the specifications given."

5.1.2 Trackability of the writing process

We worked approximately at least half of the hours reported in section 5 remotely, using the Google suite. The document named "RASD_ALPHA" is the one that we considered to be almost as the official document: that is why we will report a link to it in the github repo. We expect it to be useful to understand how the various sections of the final document have been produced and organized together. In fact, often the different sections of the document were written using different google documents, so that we could focus better on individual sections and facilitate the work of comparing the cohesion and coherence of the document.