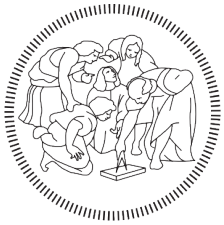


POLITECNICO
MILANO 1863

Computer Science and Engineering

Requirements Analysis and Specifications Document

DREAM - Data-Driven Predictive Farming in Telangana

Software Engineering 2 Project
Academic year 2021 - 2022

November 2021
Version 2.0

Authors:
Samuele Mariani [10622653]
Alessandro Molteni [10623928]
Matteo Monti [10622780]

Professor:
Matteo Giovanni ROSSI

Contents

Chapter 1

Introduction

1.1 Purpose

The purpose of this document is to serve as a detailed description of the 'Data-Driven Predictive farming system for the Telangana Indian region', named `DREAM` system. This document aims to provide all **functional** and **non-functional** requirements needed to develop the system.

A detailed list of use cases, interactions between the system and the user and system constraints will be included in the following sections.

This document is intended to the developers of the DREAM system, in order to provide them with all the useful requirements and information and to assist them along the implementation phase.

1.1.1 Goals

The overall ambition of the adoption and implementation of the DREAM system is to incentivize the development of a community-centric and data-driven approach to the creation of policies and regulatory models concerning farming, in order to significantly increase the farms production and the overall living conditions in the Telangana Indian region. Below are presented the goals of DREAM system:

G.1 Enable the policy makers to have a clear overview concerning the farmers situation.

- **G.1.1** Allow policy makers to incentivize those farmers who are performing well despite adversarial meteorological adverse events in order to reward them with special incentives and to ask them to provide useful best practices to the other farmers.
- **G.1.2** Allow policy makers to identify those farmers who are performing particularly badly and need to be helped by the government.

G.2 Provide policy makers with a feedback about the effectiveness of newly adopted policies and farming models provided by good farmers and agronomists.

G.3 Farmers should manage data relevant to them.

- **G.3.1** Allow farmers to visualize the following information: weather forecasts, personalized suggestions concerning specific crops to plant or specific fertilizers to use. This information should be based on their personal data such as location and type of plantation.
- **G.3.2** Allow farmers to insert all the information concerning their production and eventual problems they may face.

G.4 Farmers should be able to create and manage their own discussion forums, over a specific topic, where they can exchange ideas and suggestions with other farmers.

G.5 Farmers should be able to ask for help and suggestions to agronomists and other farmers. Each agronomist can answer to questions related to his own area of competence.

G.6 Provide each agronomist with an overview of their area which provides them with the best performing farmers and weather analytics

G.7 Agronomists should be able to access and edit their own daily schedule. In particular, they shall be able to:

- Visualize and update a daily plan to visit farms in the area
- Confirm the execution of the daily plan at the end of each day or specify the deviations from the plan.

G.8 Every Telangana's farmer, agronomist or policy maker should be able to create an account and eventually delete an already existing one.

1.2 Scope

The DREAM system allows the user to retrieve useful information based on the user role: *farmer*, *agronomist* or *policy maker*.

Each farmer has access to a *Personal page* containing every useful information about their farm and production such as weather forecasts and analytics, tips about the best crops to plant and fertilizers to use. The personal page also keeps track of the production of the farm which is regularly updated by the farmer.
Additional to the personal page farmers have access to a *Forums* section where they can create a specific forum tackling a particular topic or join conversations on already existing forums in order to exchange ideas and tips with other farmers.
Finally farmers have the possibility to ask for help on specific problems in the *Help!* section where all the different farmers questions are visible and can be answered by agronomists and other farmers.

Agronomists have a *Region overview* section which contains an overview of the area of their competence filled with a ranked list of the farmers of the area and section dedicated to the area weather analytics. Agronomists can visualize the personal page of each farmer contained in the list. They have access to the *Help!* section described above where they can answer to questions made by farmers of their area.
Finally agronomists can access their *Daily plan* section which displays an up-to-date daily schedule about farms that are yet to be visited. The schedule can be modified by the agronomist at every time.

Policy makers have access to a *State overview* section in which a ranked list of the farmers in the whole state is displayed. They can also visualize each specific ranking of every region.

1.2.1 World Phenomena

World Phenomena	Description
WP.1	The farmer decides what to plant
WP.2	The farmer harvests the crop
WP.3	The farmer faces a practical problem and needs help
WP.4	The agronomist visits a farm
WP.5	weather conditions change
WP.6	A new policy is introduced by a policy maker
WP.7	Policy maker rewards a farmer that performed particularly well

1.2.2 Shared Phenomena

Shared Phenomena	Description	Control
SP.1	The farmer registers a new harvest	World
SP.2	The farmer asks for help	World
SP.3	The farmer answer to an help request	World
SP.4	The system displays other people's suggestions	Machine
SP.5	The farmer creates a new forum	World
SP.6	The farmer answer on a specific forum	World
SP.7	The system displays the forum content	Machine
SP.8	The system displays some personal suggestions to the farmer	Machine
SP.9	The system displays weather forecasts	Machine
SP.10	The system shows the daily schedule to the agronomists	Machine
SP.11	The agronomist modifies his daily schedule	World
SP.12	The agronomist sets a farm as 'visited' in his daily schedule	World
SP.13	The system displays help requests	Machine
SP.14	The agronomist answer to help requests	World
SP.15	The system displays to the agronomist a ranked list of the best farmers in *his area*	Machine
SP.16	The system displays to the policy maker a ranked list of the best farmers in the *whole region*	Machine
SP.17	The system displays to the policy maker a ranked list of the best farmers of *each area*	Machine
SP.18	The policy maker want to check the best farmers	World

1.3 Glossary

1.3.1 Definitions

Term	Definition
Farmers	Identifies Telengana's farmers logged into the application.
Agronomists	Identifies the agronomists who have access to the system.
Policy makers	Identifies the policy makers who are involved in the project.
System	The set of hardware devices and software applications involved in the provided operations.
Personal page	A customized page for every farmer which includes personal data and weather forecast.
Forum	A web portal in which farmers can talk to each others and exchange advice.

1.3.2 Acronyms

Acronyms	Term
DREAM	Data-dRiven PrEdictive FArMing in Telengana
GPS	Global Positioning System
UI	User Interface

1.3.3 Abbreviations

Abbreviations	Term
G	Goal
WP	World Phenomena
SP	Shared Phenomena
R	Requirement

1.4 Revision history

- Version 1.0: Initial commit.
- Version 2.0: Fixed Typo errors; deleted incorrect requirements; fixed some minor use cases and mapping issues; fixed document layout issues; modified class diagram.

1.5 Reference documents

- Project assignment specification document.

1.6 Document Structure

This document is presented as it follows:

1. **Introduction**: contains a preamble of the given problem and proposes, in a simple way, the system and its goals as solution.

2. **Overall Description**: gives a general description of the system, focusing on its functions and constraints. Moreover, it provides the domain assumptions of the analysed world.

3. **Specific Requirements**: explains in detail the functional and non functional requirements. It lists the possible interactions with the system in the form of scenarios, use cases and sequence diagrams.

4. **Formal Analysis Using Alloy**: contains the Alloy model of some critical aspects of the system and an example of the generated world.

5. **Effort Spent**: keeps track of the time spent to complete this document. The first table defines the hours spent as a team for taking the most important decisions, the seconds contain the individual hours.

6. **References**: contains a list of the different documents, materials and sources used to complete this document.

Chapter 2

Overall Description

2.1 Product perspective

2.1.1 Scenarios

1. Creation of a forum

Kamal the farmer has just returned from a lunch spent with his friend Dharma. During this meal they talked a little bit about farming tools and how to use them. Since Kamal has learnt some new techniques from his friend, he is starting to wonder if he could learn even more from farmers all around Telengana. So, having opened the DREAM application, he navigates to the *Forum* section where he then proceeds to create a new forum with the title "best tools and how to use them" and select the topic from a drop down menu. Once the forum is created Kamal checks the application once in a while in order to see the responses he got and eventually answer to those responses to start a chat on the forum. When Kamal believes he has learnt enough he can close the forum and end his discussion. Alternatively, he can choose to keep the forum open in case some new farmer has something to share on that topic. Now Kamal has a deeper understanding of the tools and he can use his renewed knowledge in order to be more productive.

2. A farmer asks for help

Raj is a farmer who has recently decided to renew his farm and start to produce barely instead of grain, with the hope of increasing his productivity. Since this would be a big change for his farm, and his future, he decides not to ask for other farmer's opinions in the *Forum* section, but instead, ask directly the agronomist of his area. So Raj goes to the *Help!*, clicks on the *New help request* button and writes his answer. Here, initially, he receives answers from farmers all around Telengana, but he believes that they don't know enough about the terrain in his area and, most importantly, he wants the opinion of someone more relevant, the agronomist. After a while, the answer arrives and the agronomist suggests to Raj to not switch from grain to barely since the cost of fertilising the terrain for a new production would be too high to compensate the increasing production. Once the agronomist answer has arrived, the help request is set as "Answered", so it will appear just in farmer's *Help!* section and not in the agronomist's one.

3. A farmer wants to update his personal page

A farmer named Tarun has decided in the past to use the *Forum* section on the DREAM application in order to learn about the best fertilisers and increase his production. The suggestions he read were definitely effectively and he produced more then usual. So, he proceeds to update his personal to show the increased production. Once the DREAM application is opened it directly shows the farmer's personal page, so he just needs to click on the *Manage Crops*

where Tarun selects from a drop down menu what kind of production he has to update. After having selected the corrected crop he types in the new increased amount and confirms. Now the page will show to everyone what Tarun has achieved. Alternatively, if Tarun had decide to completely change the crops to produce, he could have select a different type from the drop down menu and the page there would be his new production.

4. An agronomist updates its schedule

Anand is an agronomist and he is closely monitoring Akash, a farmer, who unfortunately is performing particularly bad. Anand has planned to visit him both this Friday and the next one. As the first Friday passes, he realizes he can't visit Akash the next week since he has to attend his son's birthday party. So, Anand opens the DREAM application and navigates to the *Daily Schedule* schedule, he presses the *Modify* button, selects the event on the next Friday and he presses the *Postpone Event* button. Now he just need to call the farmer and asks if it is fine to re schedule the visit on Thursday. Since Akash agrees on the new day, Anand selects Thursday as the new day of the appointment and then happily closes the application knowing he will be able to both visit Akash and attend his son's birthday party.

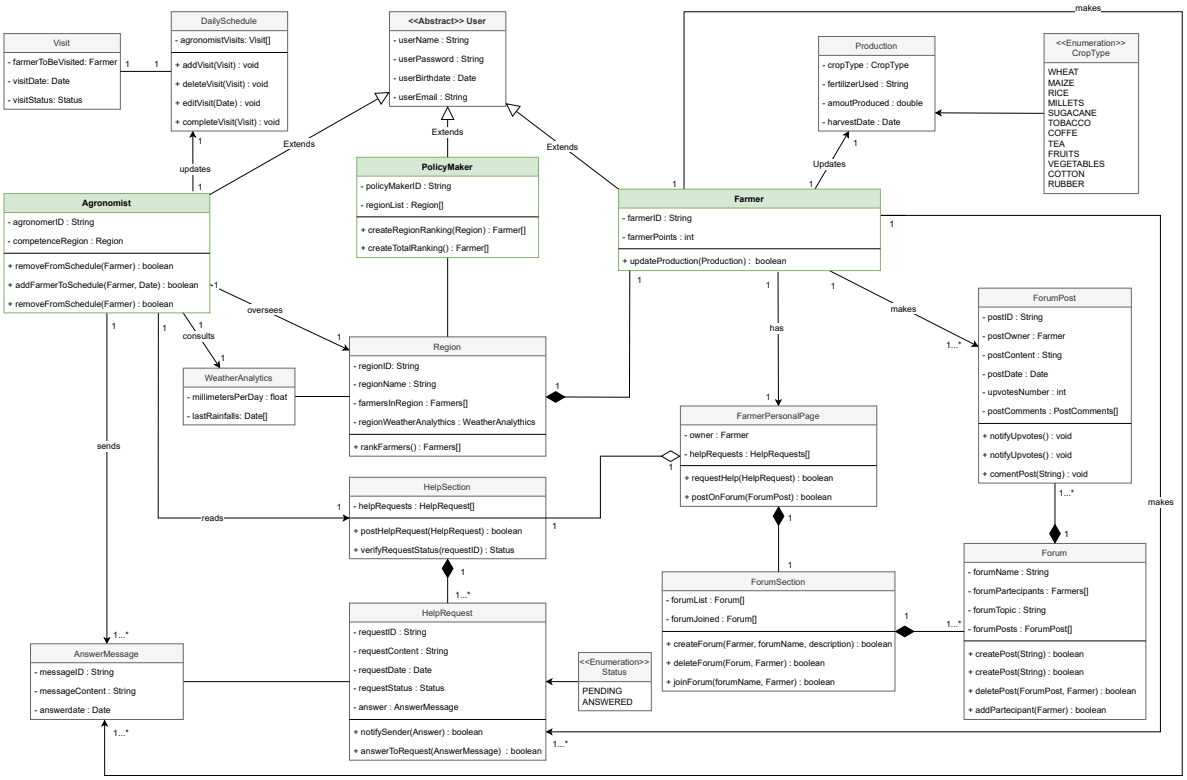
5. An agronomist answers a question of a farmer

The agronomist Dharesh has just finished his dinner with his family. After having watched some TV with his family, he puts his two daughters to bed and picks his phone to see if some farmer needs his help. He goes to the *Help!* section and notices a question asked by a farmer named Raj who is asking if it is convenient to switch his production from grain to barely. Dharesh knows that, even if he knows the farmer's production will increase, he would have to spend a lot to fertilise the terrain in order to prepare it for barely. So he responds to Raj suggesting him not to switch his production and then closes the application to go to sleep. Dharesh also had the possibility to highlight the answer of one of the other farmer if he believed that answer was sufficient to solve Raj's problem, however, no answer was sufficient enough.

6. A policy maker checks the productivity of the Siddipet Area

Piyush is a policy maker who wants to introduce a new policy to regulate the production of crops. He understand that the opinion of the agronomists that successfully managed their region is extremely important. Piyush has heard from one of his friends that the Siddipet Area has recently improved a lot in his production, so he wants to check personally. He takes his smartphone and opens the DREAM application, he goes to the *Regional Ranking* section and a map of the Telengana state, divide by it's areas, appears. Piyush clicks on the Siddipet area and the ranking is displayed, showing each farmer with his production in each crop he has plant. As expected, the production has truly increased since the last time Piyush checked that area, considering the harsh weather conditions that region has faced. So, he closes the application and immediately calls the agronomist responsible of the Siddipet area to see what strategy he used to increase so quickly the production.

2.1.2 Class diagram



2.2 Product functions

Every product function described below requires the concerned stakeholder to be authenticated. Therefore, there will be an initial login screen to continue.

2.2.1 Crop management

Farmers will have the possibility to handle their plantation data. Every crop is characterized by a state which can have one of the following values: **Planted** or **Harvested**.

New crop planted

A farmer which wants to plant a new crop, inserts the following data:

- What crop has to be planted
- Market cost of the crop seeds or plants
- Plantation date. The default plantation date will be the current system's date, and will be editable since the farmer may not have had the possibility to insert the data when he actually planted the crop.
- The fertilizer that will be used on the field
- The approximate plantation area

When all the data has been inserted, the farmer can save the new crop, that will be initialized with **Planted** state.

Harvest

When a farmer harvests a field, he updates the respective crop as follows:

- He selects a **Planted** crop from the ones displayed
- He inserts the harvest date (the date is managed as above)
- He inserts the harvested quantity. The unit of measurement of the quantity can be **Units** or **KG**

When all the data has been inserted, the farmer can update the crop, and its state will be changed in **Harvested**.

2.2.2 Forum interaction

Farmers can also create discussion forums and create new posts on them: here are described these two main functionalities.

Forum creation

A farmer that wants to create a forum will navigate to the appropriate section and will insert the forum name and a main topic from a drop-down menu. Before the creation, the DREAM system will check if this operation can be done: if there are more than 3 forums with the same topic in the same region or there is a forum with the same topic and the same name, an error message will be displayed. If not, the forum will be created and the farmer will be its admin.

Post creation

First of all, the farmer who wants to post something will look for the forum in one of the two following ways:

- Via selecting region and topic from two drop-down menus
- Via search by title

Once the forum has been selected, it will be possible to write a new post or to answer to a previous one. A post is characterized by an author, a date, a creation time and a state, which can be **Pending**, **Approved**, **Not approved**. Every new post will be in **Pending** state and will pass through:

- An automatic check for non-appropriate language
- Manual check and validation by the admin (only if its author is different from the forum admin)

In case these checks have a positive outcome, the state of the post will be set as **Approved** and will be visible to everyone, if not its state will be set as **Not approved** and the post will not be displayed.

2.2.3 Farmer ranking

This functionality assigns a score to every farmer in a specific region and creates a ranking sorted by best to worst farmer. The score is given holding into account:

- The ratio between produced quantity and the seeds price
- The ratio between the sown area and the total sowable area
- A multiplicative factor based on the weather condition that the farmer faced during the year

2.2.4 Daily schedule management

Part of the agronomist's job consists in visiting the farm in his area of competence to check the farmers and give them some advice. The DREAM system supports the agronomist by providing a daily schedule of the farms to be visited, holding into account the minimum visits indicated in the goal G.7.2. Visits are characterized by the farmer to be visited, the date and a state, that can be **Scheduled** or **Completed**. In particular, the actions that can be done are the following:

Visit modify

The agronomist can change the visit date due to unexpected events.

Visit completion

The visit state is set as **Completed**, and the agronomist will indicate the visit result (positive or negative). In case of a negative result, a new visit will be scheduled after a certain amount of time.

2.2.5 Farmer overview

One of the main functionalities that will be available for a policy maker is the possibility to have an overview of the best and worst farmers in all Telengana's state. In order to satisfy this need, the DREAM system will provide the policy maker with a map of the state divided into regions. Every region will be clickable, and by clicking on a region it will be displayed its farmers ranking, computed as described above.

2.3 User characteristics

The DREAM application has 3 different type of users:

- **Farmers**: there is no restriction on age and familiarity with technology. The application should be easy to use and intuitive, so every farmers, even the older ones, can have access to its functionalities.
- **Agronomists**: they are those in charge of monitoring a specific region of Telengana. They will have to answers to farmer's help requests and decide some strategies to increase productions. Agronomists too don't require experience with technology since the interface should be easy to use.
- **Policy Makers**: they are in charge of deciding policies for the whole Telengana state. In order to have a better overview of the best farmers and agronomists, the application will provide a global and regional ranking, so, a bit of experience on managing tables is preferred, but not needed.

2.4 Assumptions, dependencies and constraints

Domain Assumption	Description
D.1	The area overseen by the agronomist is assigned by the government and cannot be changed
D.2	The farmers always follow the directives of the agronomists
D.3	The agronomists are observant of the rules and limit their interactions to the farmers of their area
D.4	The weather forecasts and analytics are always accurate
D.5	Farmers always update their productions in a correct and regular way
D.6	The directives given by the policy makers are always respected
D.7	The farmers use the forum section and the help section only when necessary and in a proper way
D.8	The agronomists are punctual and precise when answering the farmers questions
D.9	The agronomist answers submitted in the help section are always assumed to be correct
D.10	During the registration process the user always selects his real occupation (farmer, agronomist, policy maker)
D.11	Agronomists are certified by the government
D.12	Policy makers are certified by the government
D.13	The data concerning the amount of water used by each farmer is made available by the government irrigation system
D.14	The data concerning the humidity of the soil measured by the sensors is directly available and sent to the government

Chapter 3

Specific Requirements

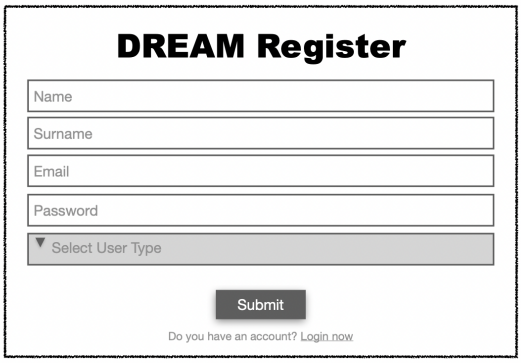
3.1 External Interface Requirements

3.1.1 User Interfaces

In the following section will be presented the early phase interfaces of the core functions of the DREAM system application.

Login and Registration

The first interfaces displayed when launching the application are the **Login** and **Register** pages:



The **login** page asks the user is to insert his email and password in order to access his profile by clicking on the *Login* button. From this page is also possible to click the *Register now* link present below the login button in order to access the **Register** page, which allows the user to create a profile by inserting essential information like his *full name*, the *email* and by choosing a *password*. Finally the user is asked to select his *user type* among those available from a **drop-down menu**; the available user types are **Farmer**, **Agronomist**, **Policy Maker**.

Farmer UI

From the **Personal Page** interface the user is able to access all the core operations offered by the application. He can access is profile to modify his personal information by clicking on the icon in the top-right corner, he can access the application settings by clicking on the gear icon in top left corner or access each main functionality by clicking the proper button.

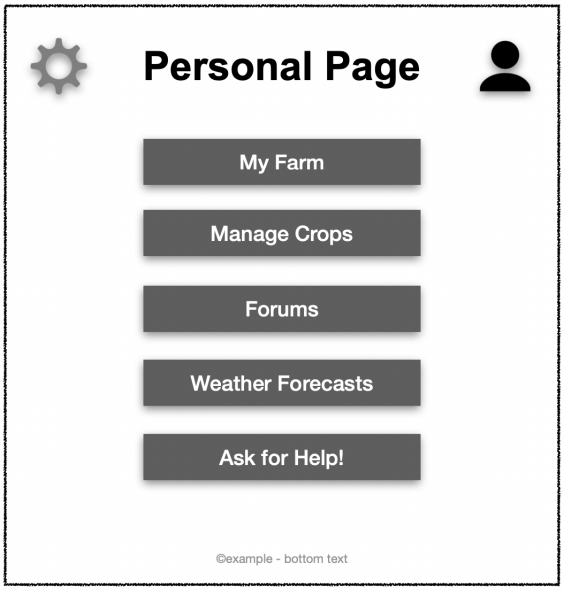


Figure 3.1: Farmer personal page

- **My Farm**: lets the user access his farm page which will contain all the farm important data, such as its name, location, extension, past months productions and costs. Figure 3.5
- **Manage Crops**: in this page the farmer will be able to manage his plantations by updating the status of his crops. In particular he will be able to **add** a new crop or to mark those that have been collected as *Harvested* (as detailed in section: 2.2.1 Crop management). Figure 3.2
- **Forums**: The forum section allows the user to browse all the different forums created by the other farmers by typing keywords in the top search bar, create new forums as well as creating posts on existing forums, (as detailed in section: 2.2.2 Forum interaction). The user is also ably to reply to other users by clicking the *Reply* button. Figure 3.3
- **Weather Forecasts**: This section of the Dream application will display up-to-date weather forecasts and conditions regarding temperature and humidity. Figure 3.6
- **Ask for Help!**: This section allows the user to submit an help request that will be send to the agronomist assigned to the farmer's area, and to all farmers. In this section the user can browse, and eventually answer to all the other help requests submitted by the other farmers. The user can see all his previous requests by clicking the icon on the top-right corner. Figure 3.4

Figure 3.2: Manage Crops Interfaces

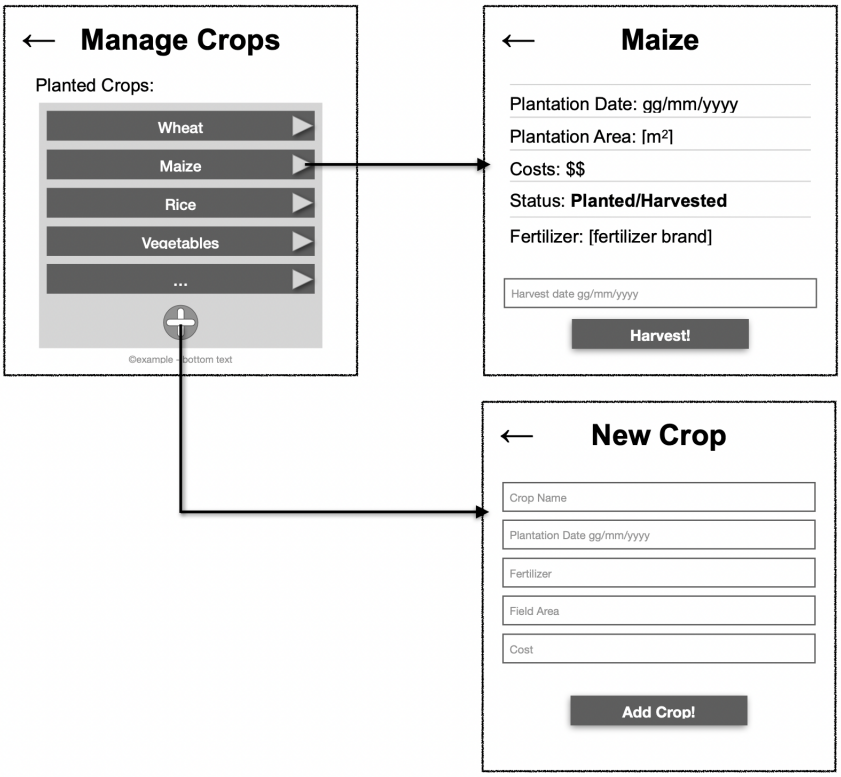


Figure 3.3: Forums Interfaces

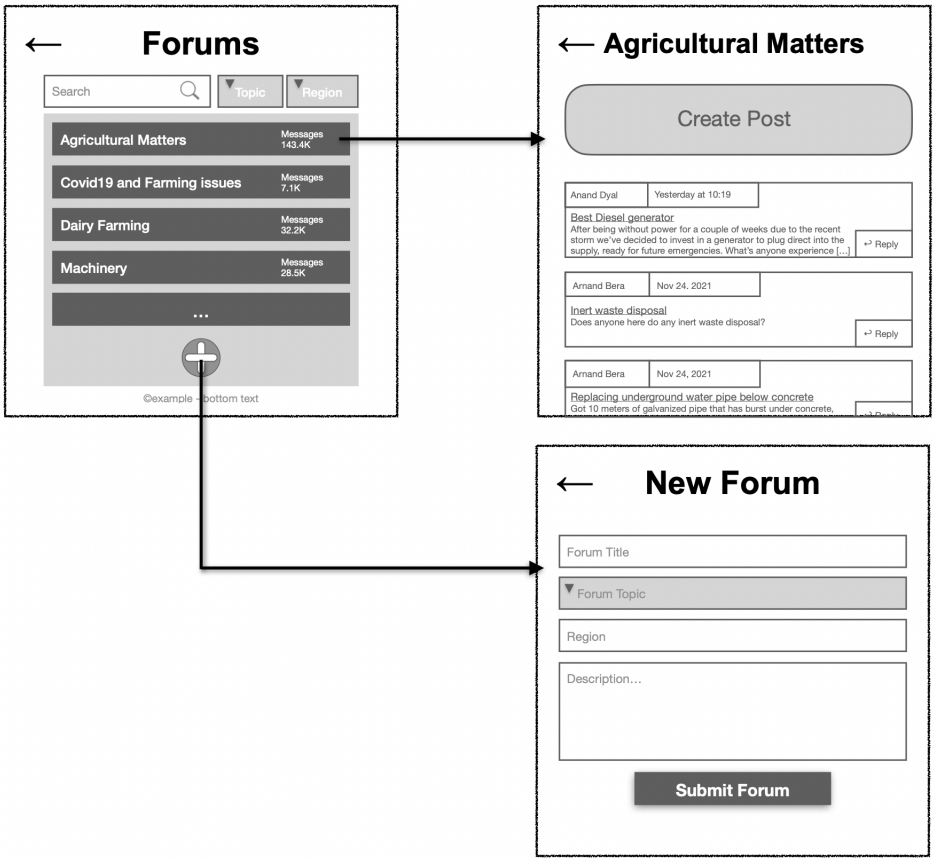


Figure 3.4: Help section Interfaces

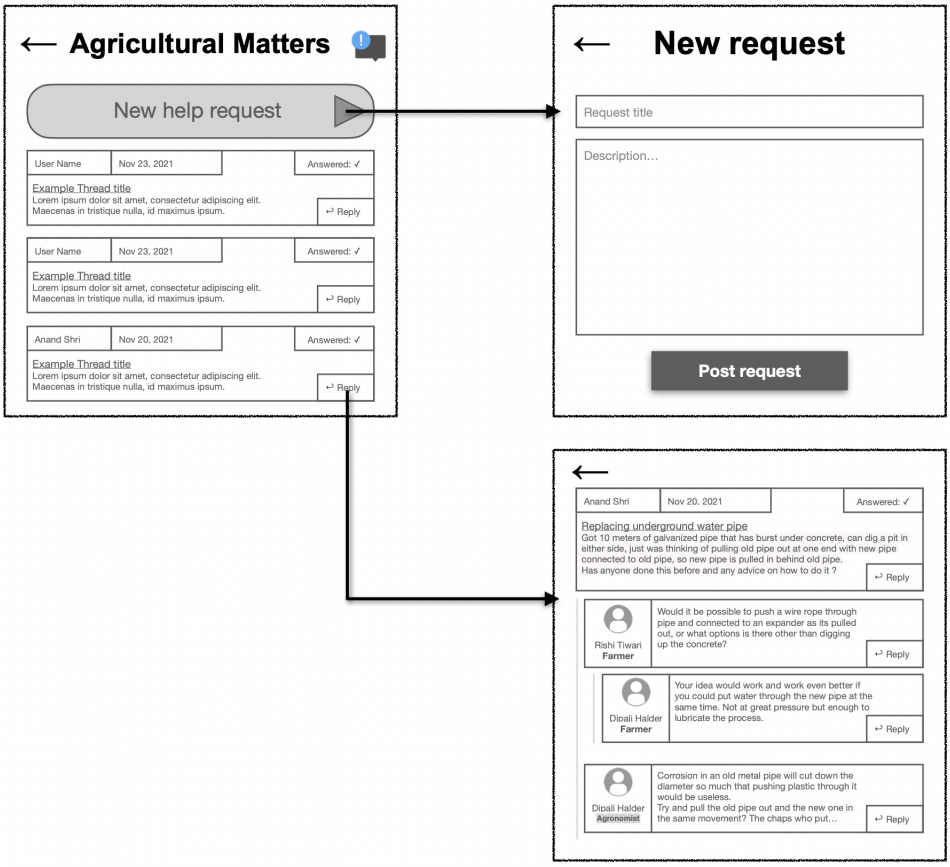


Figure 3.5: *My Farm* section Interfaces

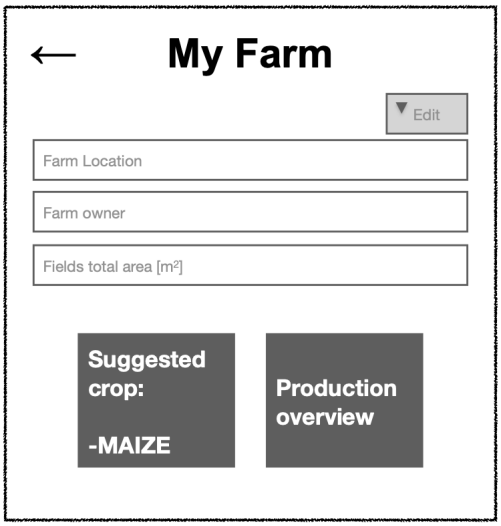
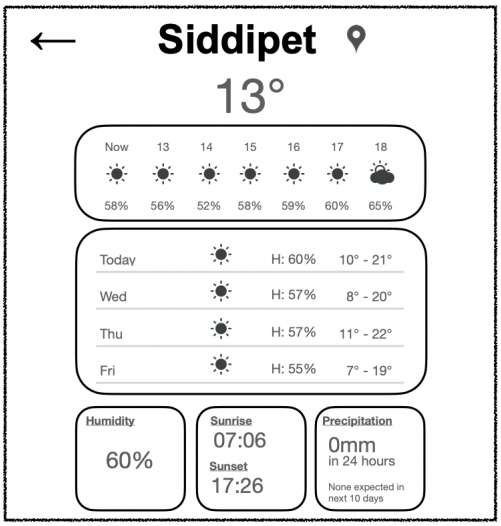


Figure 3.6: Weather section Interfaces

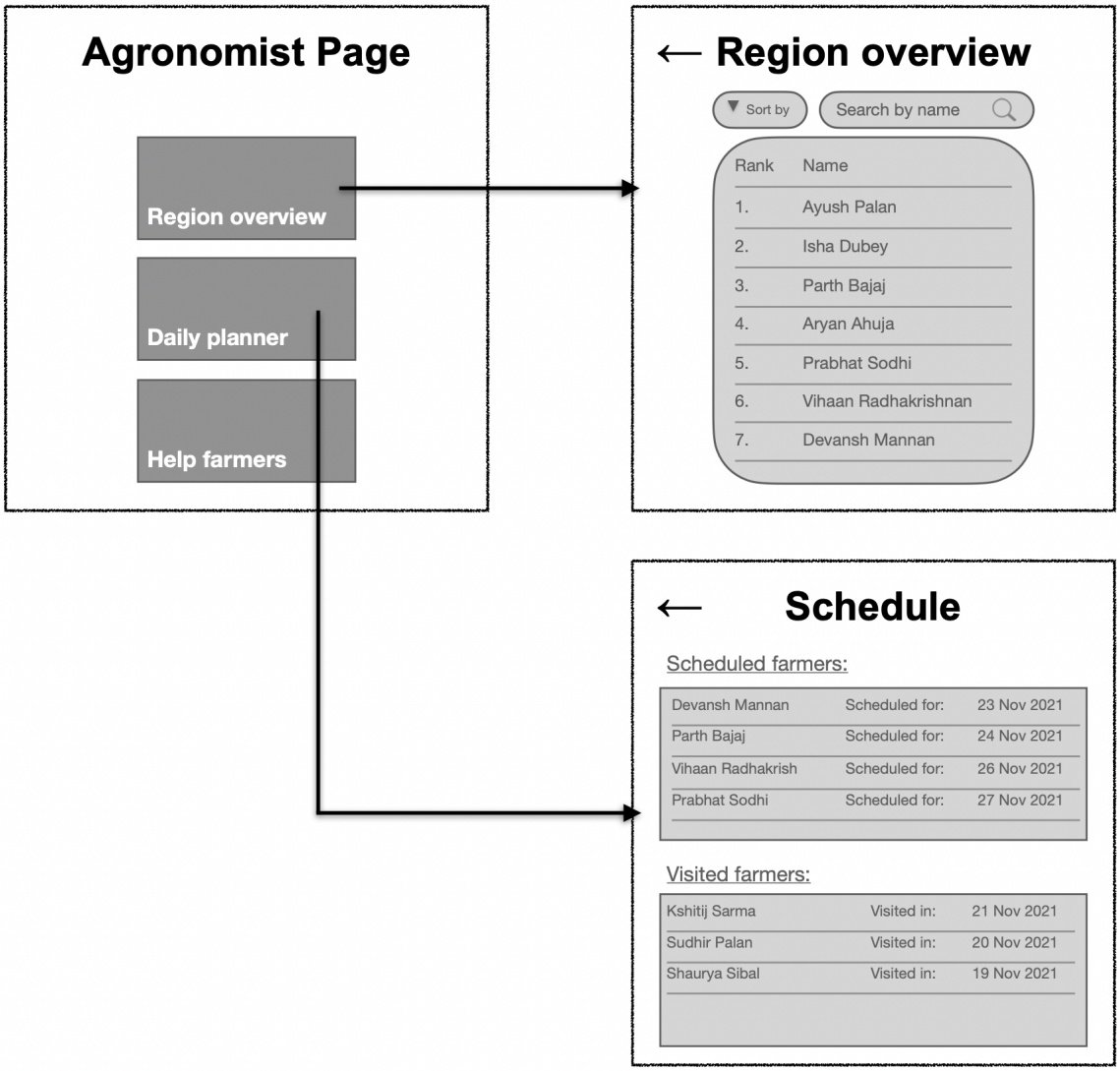


Agronomist UI

From the **Agronomist page** the user is able to access to three core interfaces: the *Region overview*, the *Daily planner* and the *Help section* interface.

- **Region overview**: From this interface the agronomist is able to visualize the region's farmers ranking. The user can sort the ranking in an ascending or descending order by using the *Sort by* drop down menu and search a specific farmer by typing his name in the search bar. From this interface the agronomist is able to access each farmer personal page (in *read only* mode) by clicking on the farmer name.
- **Daily planner**: This interface displays the agronomist schedule and keeps track of the farms that are yet to be visited.
- **Help farmer**: From this interface the agronomist can answer to all the help request submitted by the farmers.

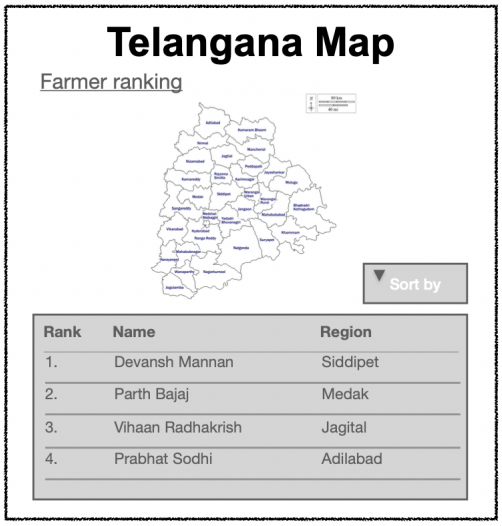
Figure 3.7: Agronomists Interfaces



3.1.2 Policy maker UI

The Policy maker interface allows the user to visualize the overall list of Telangana's farmers ranked by performances. The user is able to access each farmer personal page (in *read only* mode) by clicking on the farmer name.

Figure 3.8: Policy maker Interface



3.1.3 Hardware Interfaces

In order to maximize the effectiveness of the DREAM system application and to facilitate its employment, users are only required to have at least one mobile device capable of connecting to the internet (such as a smartphone or a tablet), to access the application directly from the workplace. All the operations offered by the mobile app could also be performed from a PC with an internet connection.

3.1.4 Software interfaces

The DREAM application will interface with a weather forecast API, in order to provide the user with reliable and up-to-date weather data regarding meteorological agents such as weather conditions, temperature and humidity.
The Dream system application should also interface with an API provided by the Telangana's government, to obtain information by the *state irrigation system*, regarding the amount of water used by each farmer in the region and by the sensors deployed on the territory to collect the data concerning the humidity of the soil.

3.1.5 Communication Interfaces

To perform the operations provided by the application, or to consult the weather service, an internet connection (Wi-Fi, mobile or wired), is required.
The user of type **Farmer** could also temporally use the application without an internet connection but he won't be able to update its personal page, receive live updates and notifications, consult the weather service or access the *Forums* and *Help* sections.
To perform their tasks users of type **Agronomists** and **Policy Maker** always need a stable internet connection.

3.2 Functional Requirements

3.2.1 List of Requirements

Requirement	Description
R1	The system shall allow an unregistered user to register
R2	The system shall allow a registered user to unregister
R3	The system shall display weather forecasts in the farmer's personal page
R4	The system shall store information about the irrigation system
R5	The system shall store information about the soil humidity
R6	The system shall display the old production of a farmer in the personal page
R7	A farmer should be able to update his production
R8	The system shall allow a farmer to store data about multiple productions
R9	A policy maker should be able to see the worst farmers of a specific region
R10	A farmer should be able to create a new forum
R11	A farmer should be able to post something on an existing forum
R12	A farmer should be able to close a forum
R13	The farmer who created a forum, should be able to change its topic
R14	A farmer should be able to send a request to an agronomist
R15	A farmer should be able to view his and other farmer's help requests and responses
R16	A farmer should be able to answer to other farmer's help requests
R17	An agronomist should be able to view his daily schedule
R18	An agronomist should be able to postpone a meeting with a farmer
R19	The system should send notifications to agronomists when a meeting is close
R20	The system shall delete a completed meeting from the calendar
R21	An agronomist should be able to view all help requests and related responses from farmers of his area
R22	An agronomist should be able to respond to all farmer's help requests from his area
R23	An agronomist should be able to see the ranking of best farmers of his area
R24	A policy maker should be able to see the best farmers of the whole Telangana state
R25	A policy maker should be able to see the worst farmers of the whole Telangana state
R26	A policy maker should be able to see the best farmers of a specific region

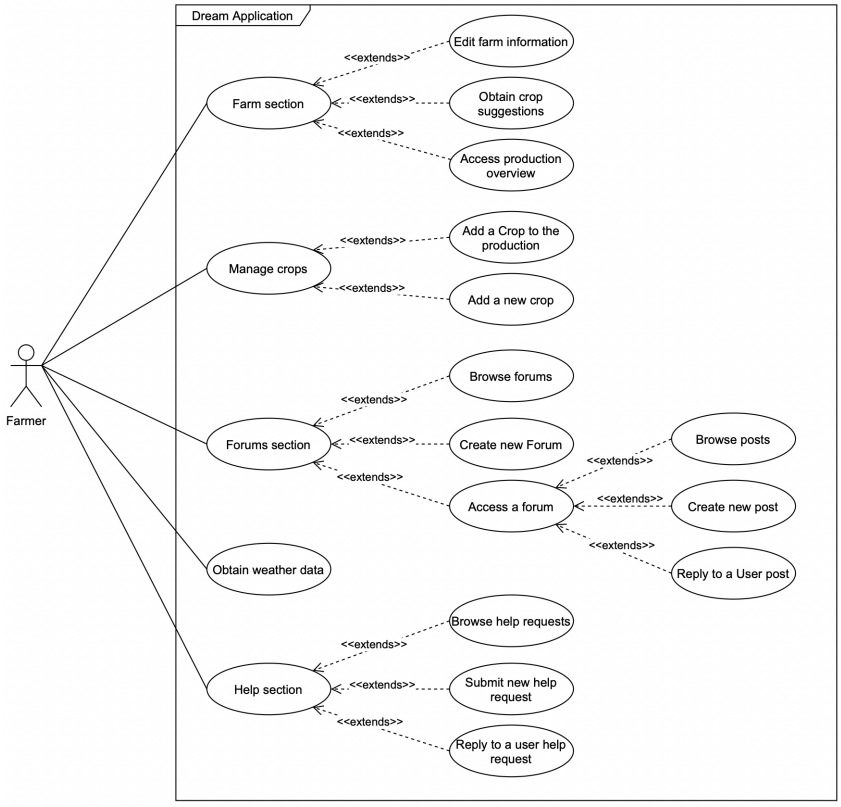
3.2.2 Mapping on Goals

Goal	Domain assumption	Requirement
G.1.1	D.4, D.6, D.12, D.13, D.14	R.4, R.5, R.24, R.26
G.1.2	D.4, D.6, D.12, D.13, D.14	R.4, R.5, R.9, R.25
G.2	D.4, D.6, D.12, D.13, D.14	R.4, R.5, R.9, R.24, R.25, R.26
G.3.1	D.2, D.4, D.8	R.3, R.14, R.15
G.3.2	D.5, D.7	R.6, R.7, R.8, R.14, R.15, R.16
G.4	D.7	R.10, R.11, R.12, R.13, R.15, R.16
G.5	D.1, D.2, D.3, D.7, D.8, D.9	R.15, R.16, R.21, R.22
G.6	D.1, D.4, D.11, D.13, D.14	R.23
G.7	D.1, D.11	R.17, R.18, R.19, R.20
G.8	D.10	R.1, R.2

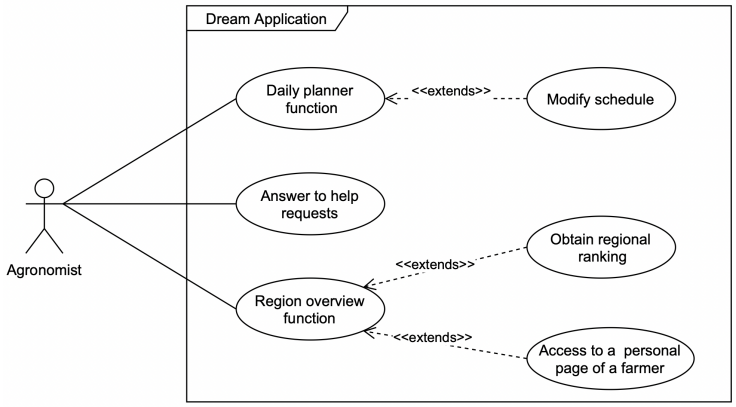
3.2.3 Use Cases diagrams

The following section will include use cases diagrams depicting all the actions and operations offered to each user type.

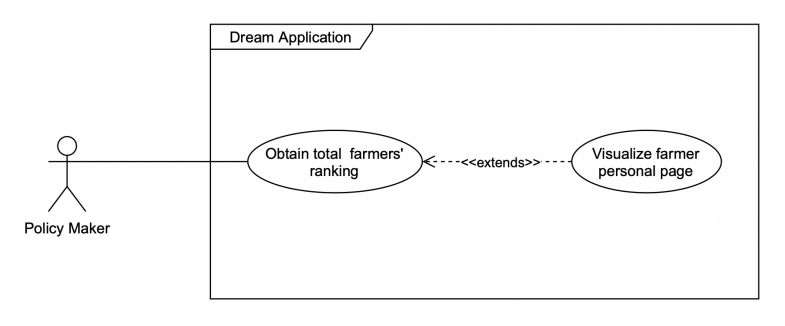
Registered farmer use cases diagram



Registered agronomist use cases diagram



Registered policy maker use cases diagram



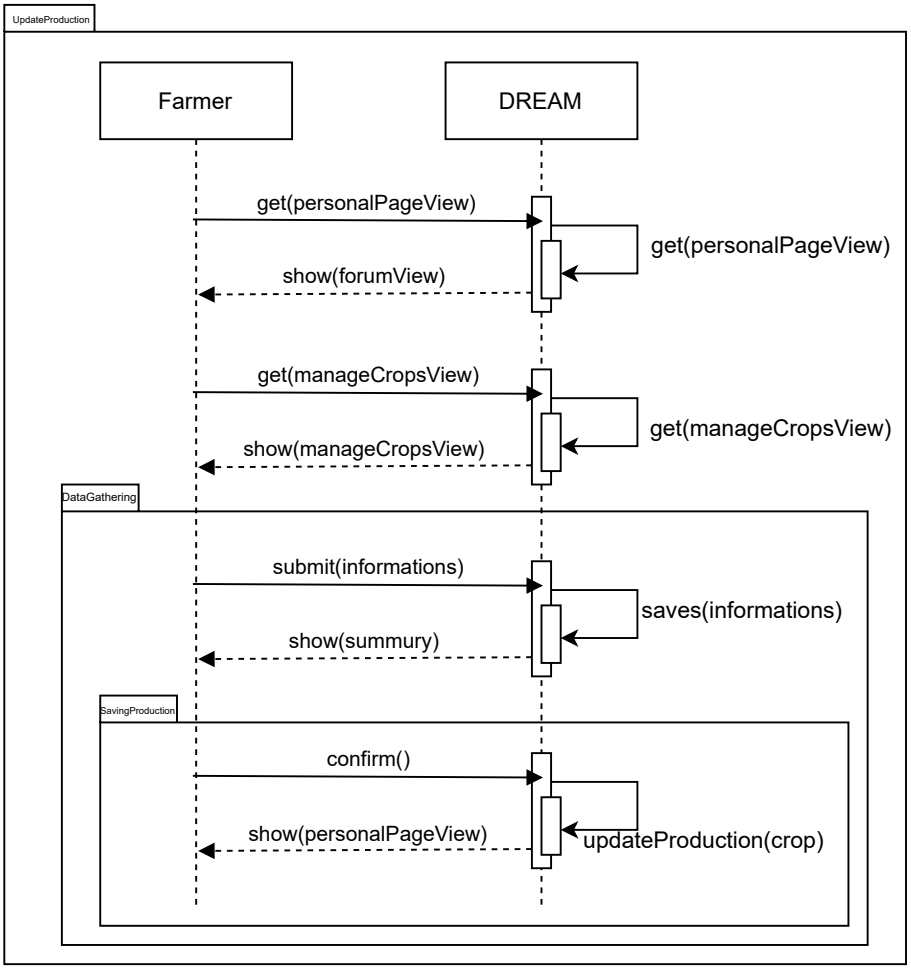
3.2.4 Use Cases and Sequence diagrams

A farmer updates his production

Name	Update personal page
Actors	Farmer
Entry conditions	• The farmer has access to the application on his device. • The farmer has a new production to register.
Flow of events	• The farmer opens the application. • The farmer presses the "Manage Crops" button. • The farmer selects from a drop down menu the the type of crop he is interested to update the production. • The farmer types in the amount he produced • The farmer clicks the "confirm" button
Exit conditions	The system saves the update and the change will now be showed in the personal page.
Exceptions	• If the farmer closes the application in the middle of the process, the system should allow the farmer to continue the update once he re-opens the application. • If the farmer has started to produce a new type of crop, the system will display the production of both crops.

Table 3.1: *Update personal page* use case description.

Figure 3.9: Update personal page sequence diagram

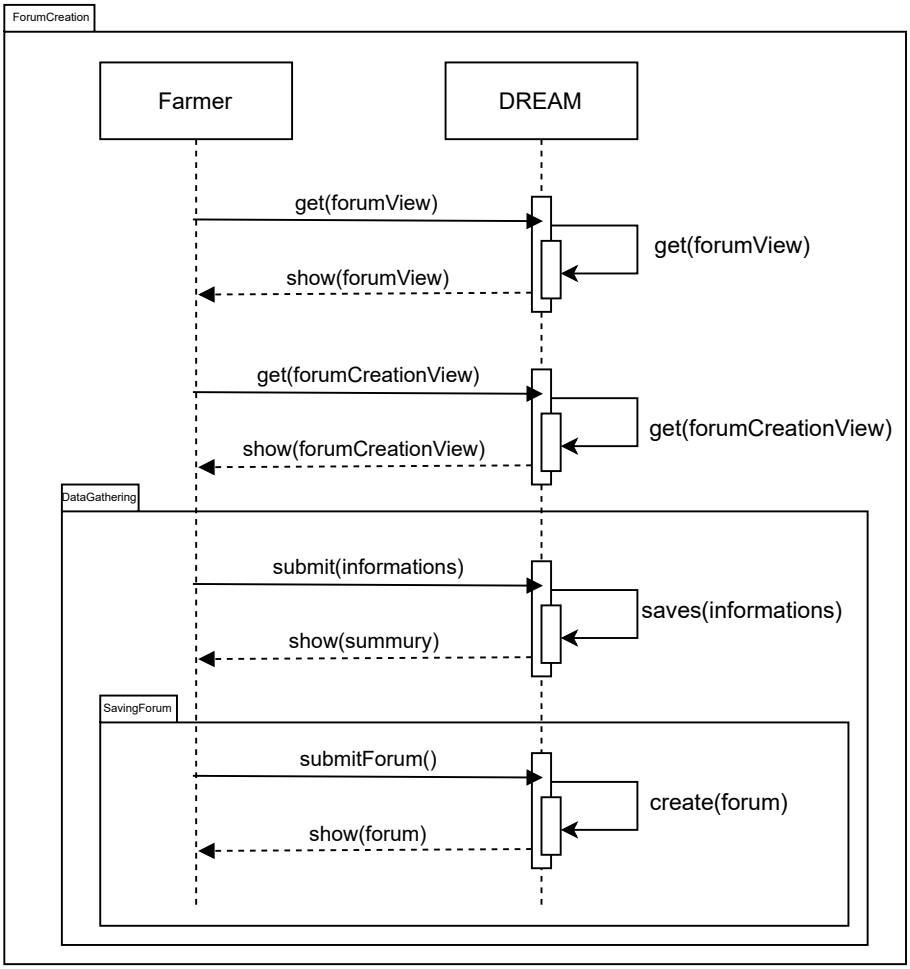


A farmer creates a forum

Name	Creation of a forum
Actors	Farmer
Entry conditions	• The farmer has access to the application on his device. • The farmer wants to learn something more on a topic.
Flow of events	• The farmer opens the application. • The farmer presses the "Forum" button. • The farmer presses the "Create a new forum" button. • The farmer writes the title and selects the topic from a drop down menu. • The Farmer clicks the "Submit forum" button.
Exit conditions	The system saves the data related to the forum and posts it.
Exceptions	• If the farmer closes the application in the middle of the process, the system should allow the farmer to continue the creation once he re-opens the application. • If the farmer selects the wrong topic, he should be able to change it.

Table 3.2: *Creation of a forum* use case description.

Figure 3.10: Forum creation sequence diagram

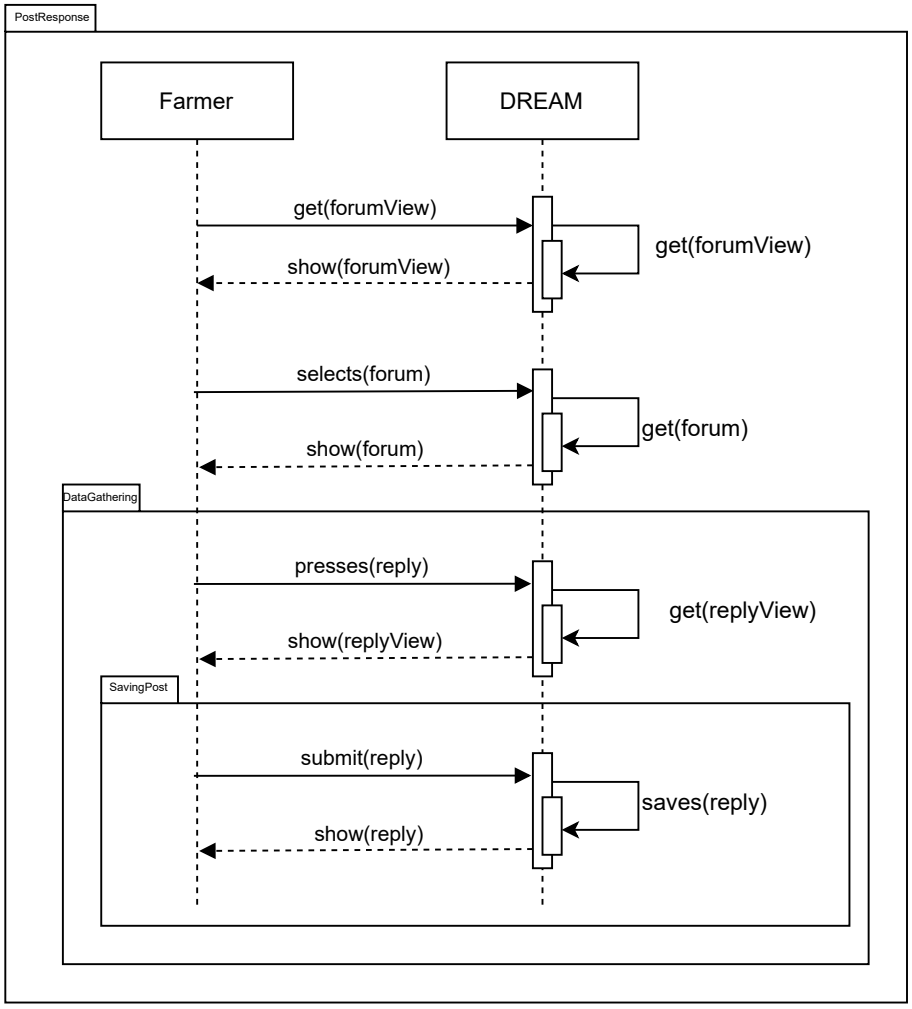


A farmer posts something on a forum

Name	Posting in a forum
Actors	Farmer
Entry conditions	• The farmer has access to the application on his device. • The farmer wants to post something in a forum.
Flow of events	• The farmer opens the application. • The farmer presses the "Forum" button. • The farmer chooses the forum he wants to answer to. • The farmer presses the "Reply" button • The Farmer writes his reply.
Exit conditions	The system saves reply and it will send a notification to the forum's owner.
Exceptions	• If the farmer closes the application in the middle of the process, the system should allow the farmer to continue the operation once he re-opens the application. • If the farmer post a wrong answer, he should be able to delete it and re write his post.

Table 3.3: *Submission of a post* use case description.

Figure 3.11: Post submission sequence diagram

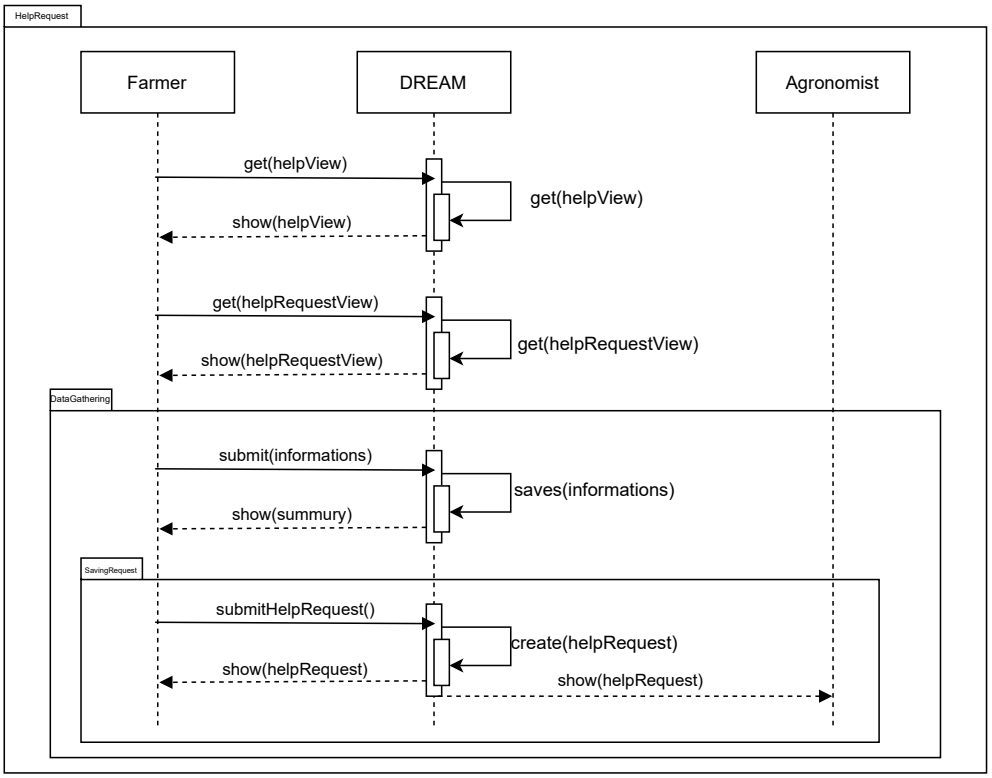


A farmer asks for help

Name	asking for help
Actors	• Farmer. • Agronomist.
Entry conditions	• The farmer has access to the application on his device. • The farmer has a practical question that requires the help pf the agronomist.
Flow of events	• The farmer opens the application. • The farmer presses the "Help!" button. • The farmer presses the "new help request" button. • The farmer formulate his question.
Exit conditions	The system register his question and it will be visible to the Agronomist.
Exceptions	If the farmer closes the application in the middle of the process, the system should allow the farmer to continue the operation once he re-opens the application.

Table 3.4: *A farmer asks for help* use case description.

Figure 3.12: Help Request sequence diagram

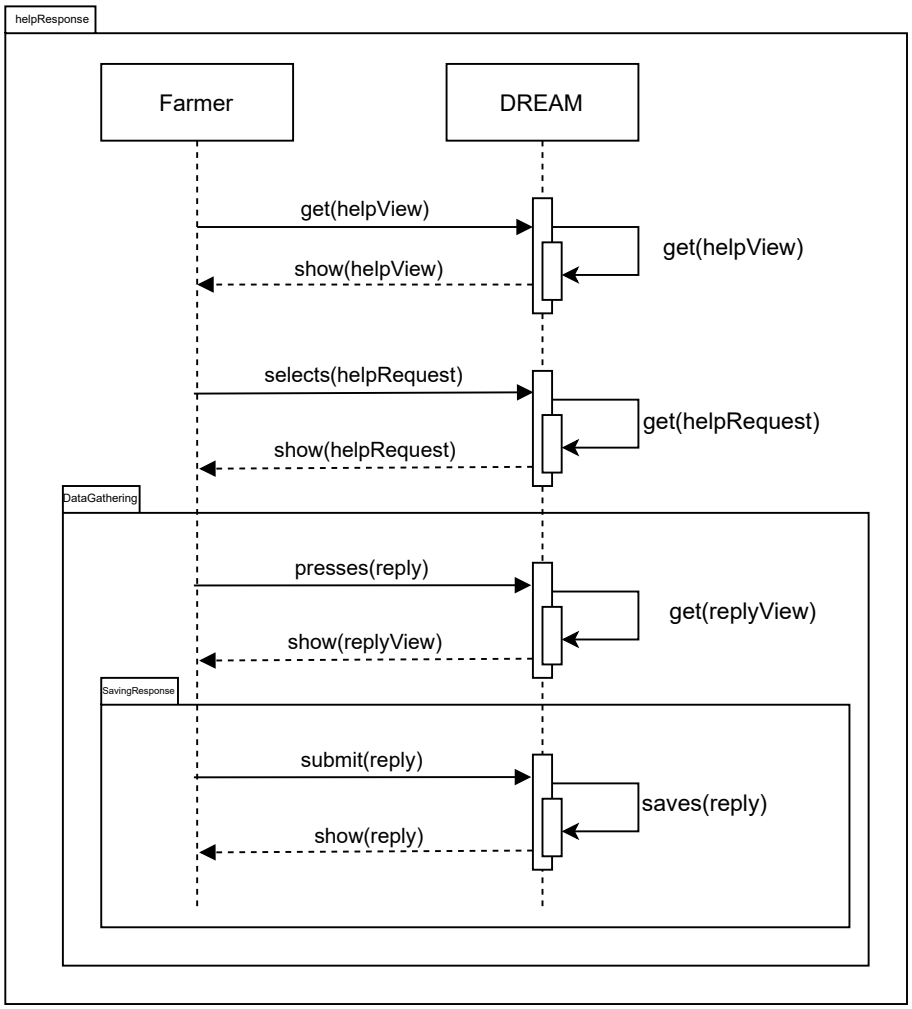


A farmer answers to an help request

Name	respond to another farmer
Actors	Farmer
Entry conditions	• The farmer has access to the application on his device. • The farmer wants to answer to another farmer help request.
Flow of events	• The farmer opens the application. • The farmer presses the "Help!" button. • The farmer selects the request he wants to respond to. • The farmer presses the "Reply" button. • The farmer writes his reply. • The farmer presses the "Submit" button.
Exit conditions	The system register his reply and it will be visible to all the other farmers.
Exceptions	If the farmer closes the application in the middle of the process, the system should allow the farmer to continue the operation once he re-opens the application.

Table 3.5: *A farmer answers to an help request* use case description.

Figure 3.13: Help response sequence diagram

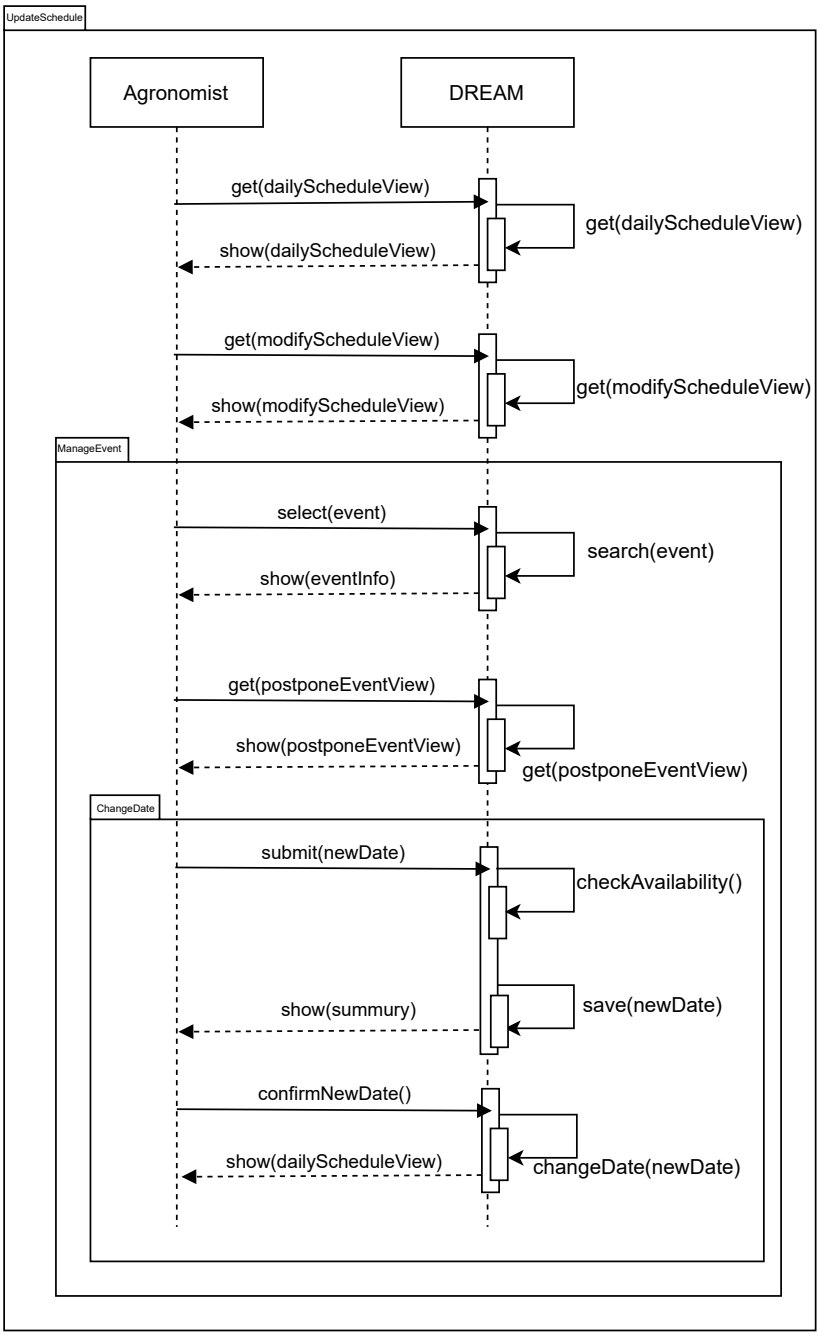


An agronomist updates his daily schedule

Name	Update Daily Schedule
Actors	Agronomist
Entry conditions	• The agronomist has access to the application on his device. • The agronomist can't attend a scheduled meeting with a farmer.
Flow of events	• The agronomist opens the application. • The agronomist goes to the "Daily schedule" section. • The agronomist presses the "Modify" button. • The agronomist selects the interested event. • The agronomist presses the "Modify Event" button. • The agronomist selects the new date from the calendar. • The agronomist presses the "Confirm" button.
Exit conditions	The system saves the change in the schedule and the notification system changes accordingly.
Exceptions	• If the agronomist closes the application in the middle of the process, the system should allow the agronomist to continue the operation once he re-opens the application. • If the agronomists wants to cancel the operation, the system doesn't perform any change in the calendar.

Table 3.6: *Update Daily Schedule* use case description.

Figure 3.14: Modify daily schedule sequence diagram

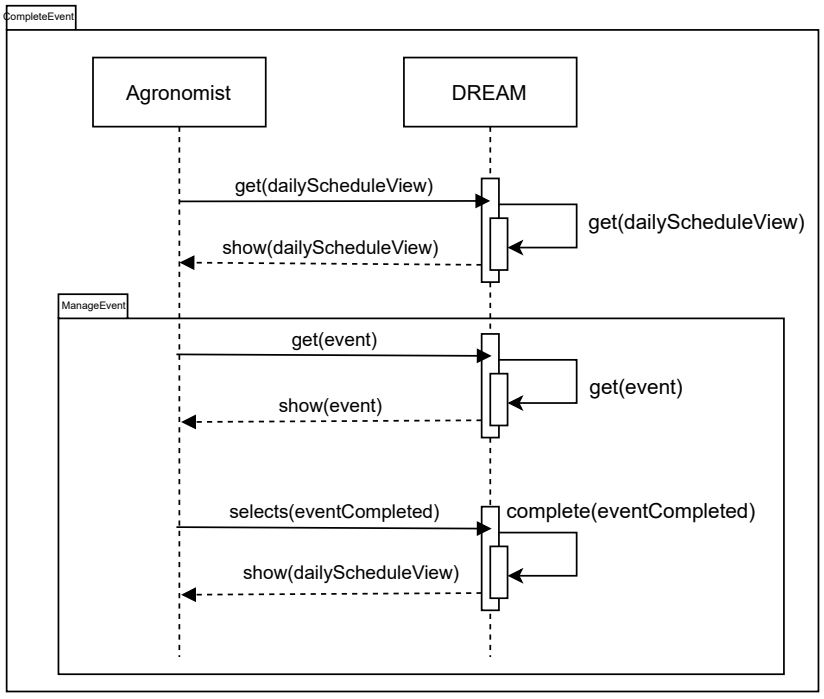


An agronomist completes a visit

Name	Completes a visit
Actors	Agronomist
Entry conditions	• The agronomist has access to the application on his device. • The agronomist has completed a visit.
Flow of events	• The agronomist opens the application. • The agronomist goes to the "Daily schedule" section. • The agronomist selects the visit he has completed. • The agronomist presses the "Event completed" button.
Exit conditions	The system display the newly completed event in the "Visited farmers" section.
Exceptions	• If the agronomist closes the application in the middle of the process, the system should allow the agronomist to continue the operation once he re-opens the application.

Table 3.7: *event completed* use case description.

Figure 3.15: Event completed sequence diagram

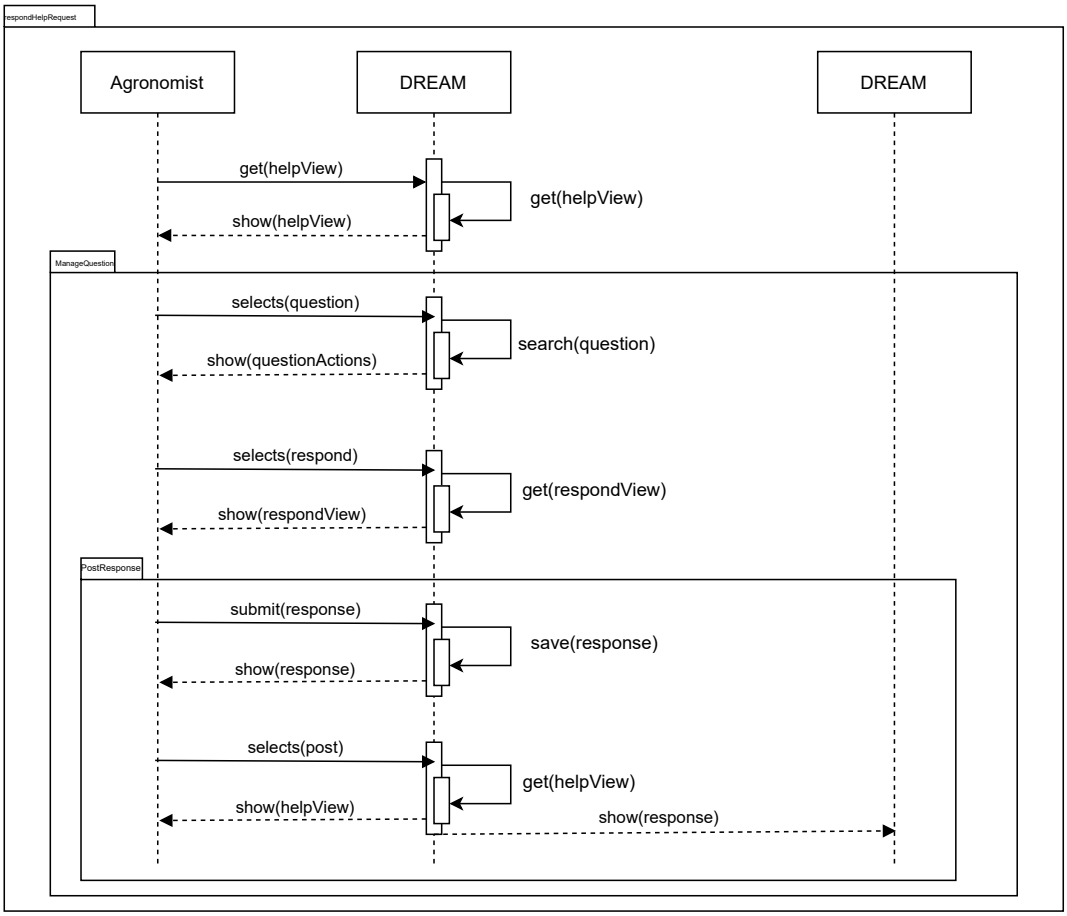


An agronomist answers to an help request

Name	Answer an help request
Actors	Agronomist
Entry conditions	• The agronomist has access to the application on his device. • A farmer has submitted an help request.
Flow of events	• The agronomist opens the application. • The agronomist goes to the "Help!" section. • The agronomist selects the help request he wants to answer to. • The agronomist presses the "Answer" button. • The agronomist writes the answer. • The agronomist presses the "Post" button.
Exit conditions	The system notifies the farmer that the agronomist answered to his request and the request is set as "Answered".
Exceptions	If the agronomist closes the application in the middle of the process, the system should allow the agronomist to continue the operation once he re-opens the application.

Table 3.8: *Answer a question* use case description.

Figure 3.16: Answer help request sequence diagram

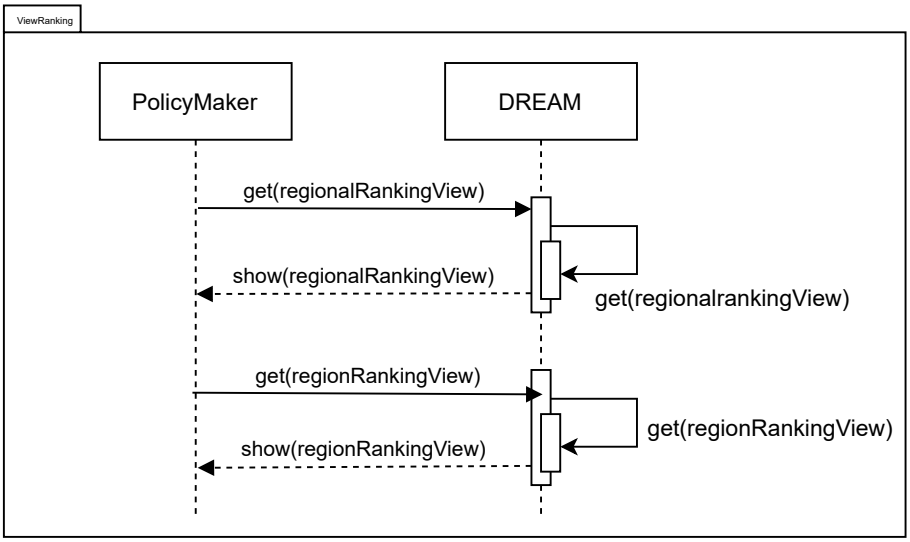


A policy maker checks the ranking of the farmers of a specific region

Name	Checking the ranking of a specific area
Actors	Policy Maker
Entry conditions	• The policy maker has access to the application on his device. • The policy maker wants the check the productivity of a specific area.
Flow of events	• The policy maker opens the application. • The policy maker goes to the "Regional ranking" section. • The policy maker selects the region from the map he is interested in.
Exit conditions	The system shows the ranking of the best farmers of the selected area.
Exceptions	If the policy maker closes the application in the middle of the process, the system should allow the policy maker to continue the operation once he re opens the application.

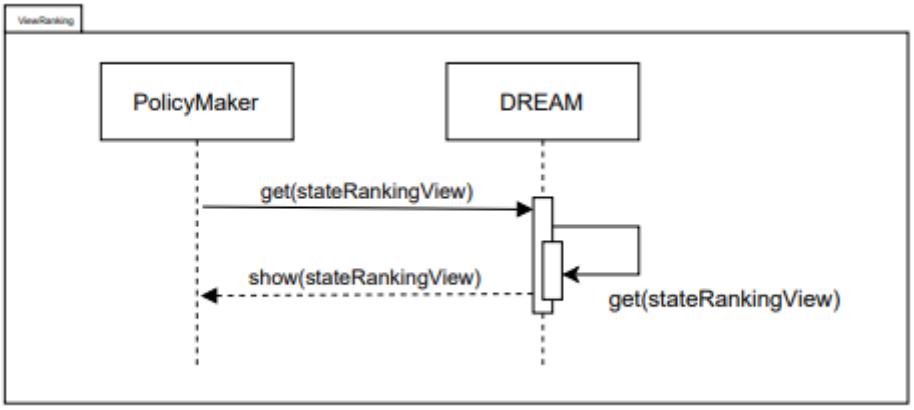
Table 3.9: *Checking the ranking of a specific area.*

Figure 3.17: View regional ranking sequence diagram



A policy maker views the best farmers of the entire state

Figure 3.18: View State Ranking sequence diagram

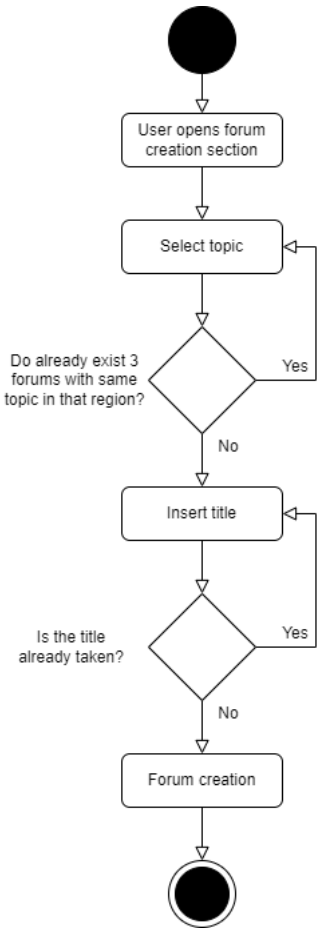


3.2.5 Activity diagrams

Below are described the **forum creation** and **post creation** processes, which are the ones that require choices during the processes.

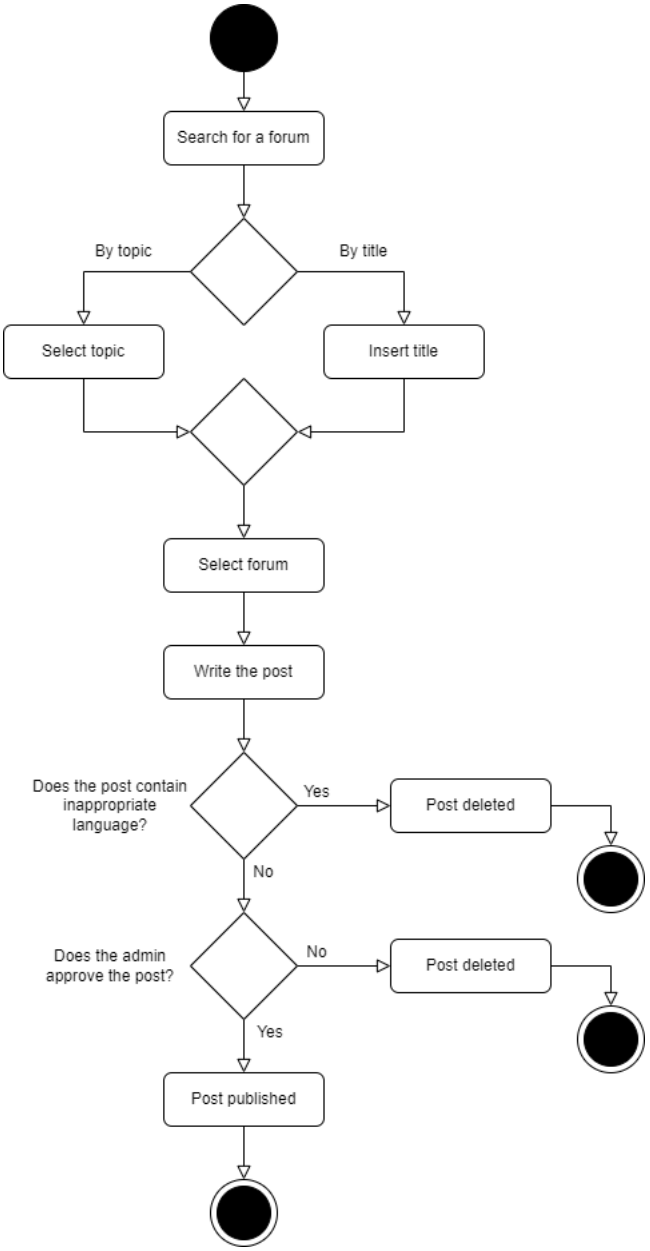
Forum creation

This diagram depicts the *forum creation* process:



Post creation

This diagram shows the steps necessary to write a post on a forum. It is assumed that the post author is different from the forum owner, in order to show the admin validation process.



3.2.6 Mapping on Requirements

Use Case	Requirements
Registration and unregistration	R.1, R.2
Checking the ranking of a specific area	R.3, R.4, R.5
A farmer checks another farmer's personal page	R.6
A farmer update his production	R.7, R.8
A farmer creates a forum	R.10, R.12, R.13
A farmer post something on a forum	R.11
A farmer asks for help	R.14
A farmer answers to an help request	R.15, R.16
An agronomist updates his daily schedule	R.17, R.18
An agronomist completes a meeting	R.19, R.20
An agronomist answers to an help request	R.21, R.22
An agronomist views the ranking of his area	R.23
A policy maker checks the ranking of the farmers of a specific region	R.9, R.26
A policy maker views the best farmers of the entire state	R.24, R.25

3.3 Performance Requirements

Most of the computation will be carried out by the server, so the app and the web portal should be light-weighted.
The farmer ranking and weather forecast functionalities should be processed in less than 5 seconds
Every other operation does not require particular computational power, so they must be processed in less than one second

3.4 Design Constraints

3.4.1 Standard Compliance

Some sensitive data is needed to be stored in DREAM's DB, such as user's personal data or farms GPS coordinates. Consequently, when a user logs in for the first time, he will be asked for the permission to handle these data. All data must be processed in order to respect the current privacy laws.

3.4.2 Hardware Limitations

- **Customer device**
  - Stable internet connection (3G or more for mobile devices)
  - A supported version of one of the most common web browsers (to access to the web portal)
  - Android 9 - Pie or newer (for Android mobile devices)
  - iOS 12.5 or newer (for Apple mobile devices)
  - Enough memory to download and install the application
- **Server**
  - Stable and cabled internet connection
  - Enough memory and computational power to handle thousands parallel requests
  - Enough disk space to store the DB

3.5 Software System Attributes

3.5.1 Reliability

The service is required to be fault tolerant, in order to guarantee its continuity: exception handling has to be arranged to keep the system working even if some errors occur. It is also necessary to schedule a periodic backup to prevent data loss.

3.5.2 Availability

Most of DREAM's functionalities need a high level of availability. In particular, all the functionalities concerning policy makers and agronomists, and the farmer's help requests, shall guarantee 99.9% (*three-nines*) of availability, that correspond to a maximum downtime of 8.76 hours per year.

The other functionalities (crop updating and forum discussions) have a less strict constraint regarding the downtime: these functions shall be operative for 99% of time (*two-nines*), so at most 3.65 days per year.
Indeed, the crop date is editable and discussion forums weren't meant for crucial issues.

3.5.3 Security

The system should provide a 2-step login method, which includes authorization and authentication

- **Authentication**: request and verify the identity of DREAM's stakeholders, through a login page that requests for username and password **Authorization**: grant permissions to the user who logged in, relying on its user type (**Policy maker**, **Agronomist** or **Farmer**)

3.5.4 Maintainability

It is crucial to develop an error logging system which collects every error that may occur and warns the maintenance staff providing all possible information needed for troubleshooting, such as when the error occurred, which user typology is concerned and what kind of action was being performed.

It is also needed to keep the code of the functionalities separated as much as possible, in order to simplify updating, bug fixes and testing operations.

Data import for weather forecasts and updating operations shall be completed at night (approximately between 1.00 and 5.00 AM, local time) to minimize the disservice.

3.5.5 Portability

Mobile application will be released for Android and iOS, and the web portal will be accessible from all the main web browsers.

Chapter 4

Formal Analysis Using Alloy

In this section, a formal model of the problem is presented. The model has been obtained by using the Allot tool.
The following model represents only the core parts of the problem. Other minor feature's constraints have been simplified in order to preserve readability.

4.1 Alloy Model

```
///////////////
//Signatures//
///////////////
abstract sig AppUser {
  id: one Int
}{id > 0}

sig Farmer extends AppUser {
  farm: one Farm,
  region: one Region,
  fourm: set Forum,
  crops: some Crop,
  globalRanking: one GlobalRanking
}{id > 0}


sig Agronomist extends AppUser {
  schedule: some Visit,
  region: one Region
}{id > 0}

sig PolicyMaker extends AppUser {
  globalRanking : one GlobalRanking,
  regionalRankings: some RegionalRanking
}{id > 0}

sig Visit {
  farmer_id: one Int,
  state: one VisitState,
  date: one Date
}

sig Date {}
```

```
sig Coordinates {}

sig Farm {
  farmer: one Farmer,
  coords: one Coordinates
}

sig Region{
  region_id: one Int,
  agronomist: one Agronomist,
  ranking : one RegionalRanking
}{region_id > 0}


sig Forum {
  topic: one Topic,
  owner: one Farmer,
  posts: some Post
}

sig Post {
  author: one Farmer
}

sig RegionalRanking {
  region : one Region,
  policyMakers: some PolicyMaker
}

sig GlobalRanking {
  farmers : some Farmer
}


sig Crop{
  cropType: one CropType,
  farmer: one Farmer
}

sig HelpRequest {
  farmer: one Farmer,
  agronomist: one Agronomist,
  state: one HelpRequestState
}

//////////////
////Enums/////
//////////////
enum VisitState {
  Scheduled,
  Completed
}

enum Topic {
  Agricultural_matters,
  Weather,
  Politics_and_Covid19,
  Dairy_Farming,
```

```
  Livestock,
  Cropping,
  Machinery,
  Buildings_and_infrastructures,
  Renewable_Energy,
  News
}

enum CropType {
  Maize,
  Wheat,
  Rice,
  Tobacco
}

enum PostState {
  Pending,
  Approved,
  NotApproved
}

enum HelpRequestState {
  NotAnswered,
  Answered
}

//////////////
////Facts/////
//////////////
fact eachFarmerOwnsOneFarm {
  all a: Farmer | one b:Farm |
    a.farm = b and b.farmer = a
}

fact eachFarmIsOwnedByOneFarmer {
  all a: Farm | one b:Farmer |
    a.farmer = b and b.farm = a
}

fact eachFarmHasDifferentCoordinates {
  all c: Coordinates | one f:Farm | f.coords = c
}

fact eachUserHasUniqueId {
  no disj u1 , u2 : AppUser | u1.id = u2.id
}

fact eachDateCorrespondsToAVisit {
  all d: Date | some v:Visit | v.date = d
}

fact eachVisitToOneAgronomist {
  all v: Visit | one a:Agronomist |
    v in a.schedule
}

fact eachVisitToOneFarmer {
  all v: Visit | one f:Farmer |
```

```
    v.farmer_id = f.id
}

fact eachFarmerAtLeast2Visits {
  all f:Farmer | some v1,v2:Visit |
    v1.farmer_id = v2.farmer_id and
      v1 != v2 and
        f.id = v2.farmer_id
}

fact visitsDividedByRegion {
  all a:Agronomist | all v:Visit | all f:Farmer |
    (v in a.schedule and v.farmer_id = f.id) implies
      a.region = f.region
}

fact noRegionWithoutFarmers {
  all a:Agronomist | some f:Farmer |
    a.region = f.region
}

fact noRegionWithoutAgronomists {
  all f:Farmer | some a:Agronomist |
    a.region = f.region
}

fact postRefersToOneForum {
  all p:Post | one f:Forum | p in f.posts
}

fact helpRequestSameRegion {
  all h:HelpRequest | all f:Farmer | all a:Agronomist |
    (f = h.farmer and a = h.agronomist) => (f.region = a.region)
}

fact eachAgronomistAssignedToOneRegion {
  all a:Agronomist | one r:Region |
    a.region = r and r.agronomist = a
}

fact eachReagionHasAnAgronomist {
  all r: Region | one a:Agronomist |
    r.agronomist = a and a.region = r
}

fact eachReagionHasARanking {
  all r: Region | one a:RegionalRanking |
    r.ranking = a and a.region = r
}

fact eachRegionalRankingHasARegion {
  all a:RegionalRanking | one r:Region |
    a.region = r and r.ranking = a
}

fact onlyOneGlobalRankingExists {
  no g1, g2 : GlobalRanking | g1 != g2
}
```

```
208
209
210 fact eachPolicyMakerHasAccessToAllRegionalRankings{
211   all r:RegionalRanking | all p:PolicyMaker |
212     ((p in PolicyMaker) => (r in p.regionalRankings)) and
213       ((r in RegionalRanking) => (p in r.policyMakers))
214 }
215
216
217
218
219 ///////////////
220 //Predicates//
221 ///////////////
222 pred show {
223   #Farmer = 3
224   #Agronomist = 2
225   #PolicyMaker = 1
226 }
227
228 run show for 10
```

4.2 Alloy Model Execution Outcome

Executing "Run show for 10"
Solver=sat4j Bitwidth=4 MaxSeq=7 SkolemDepth=1 Symmetry=20
52829 vars. 3600 primary vars. 123529 clauses. 1040ms.
Instance found. Predicate is consistent. 363ms.

4.3 World generation

4.3.1 First World

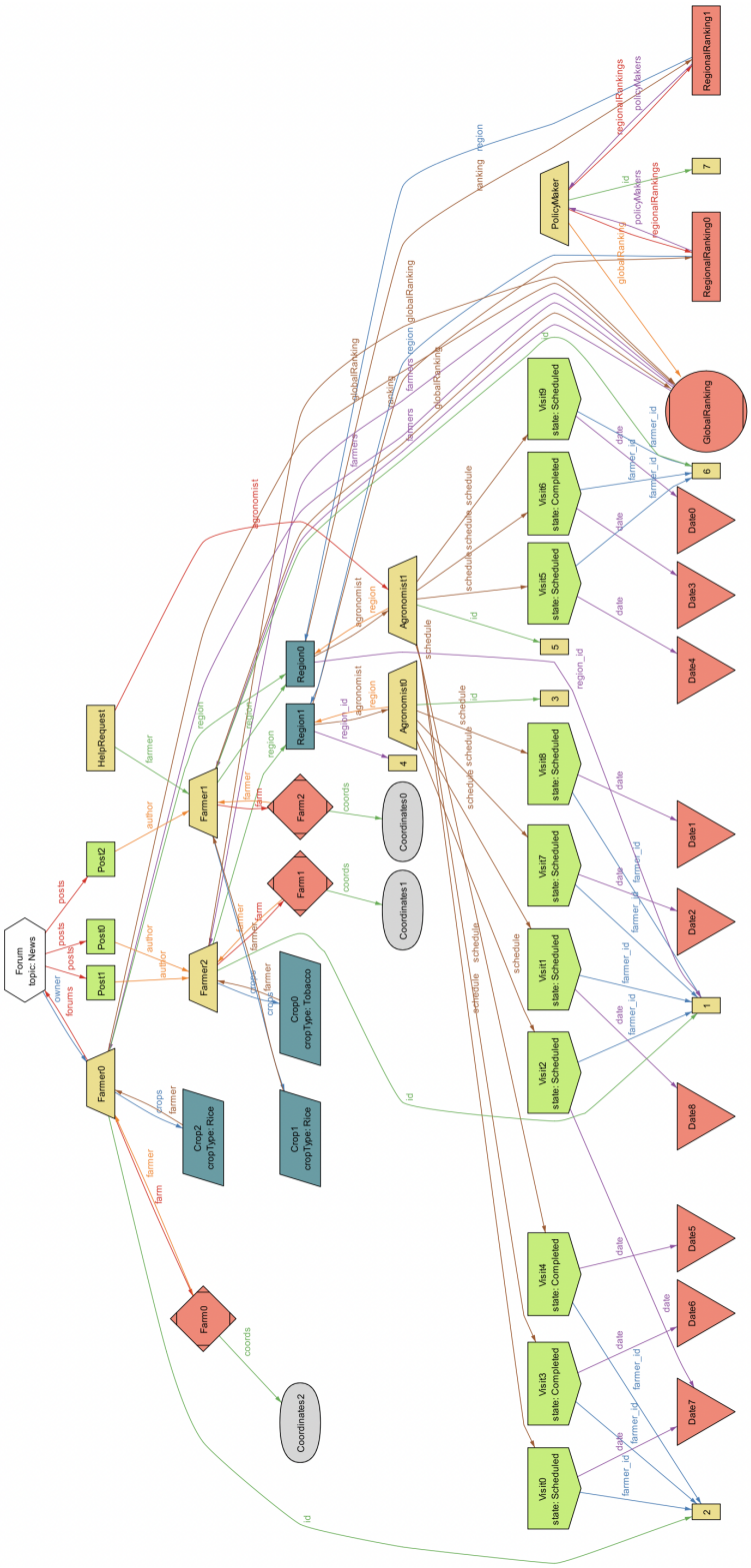


Figure 4.1: First world generated using alloy tool

4.3.2 Second World

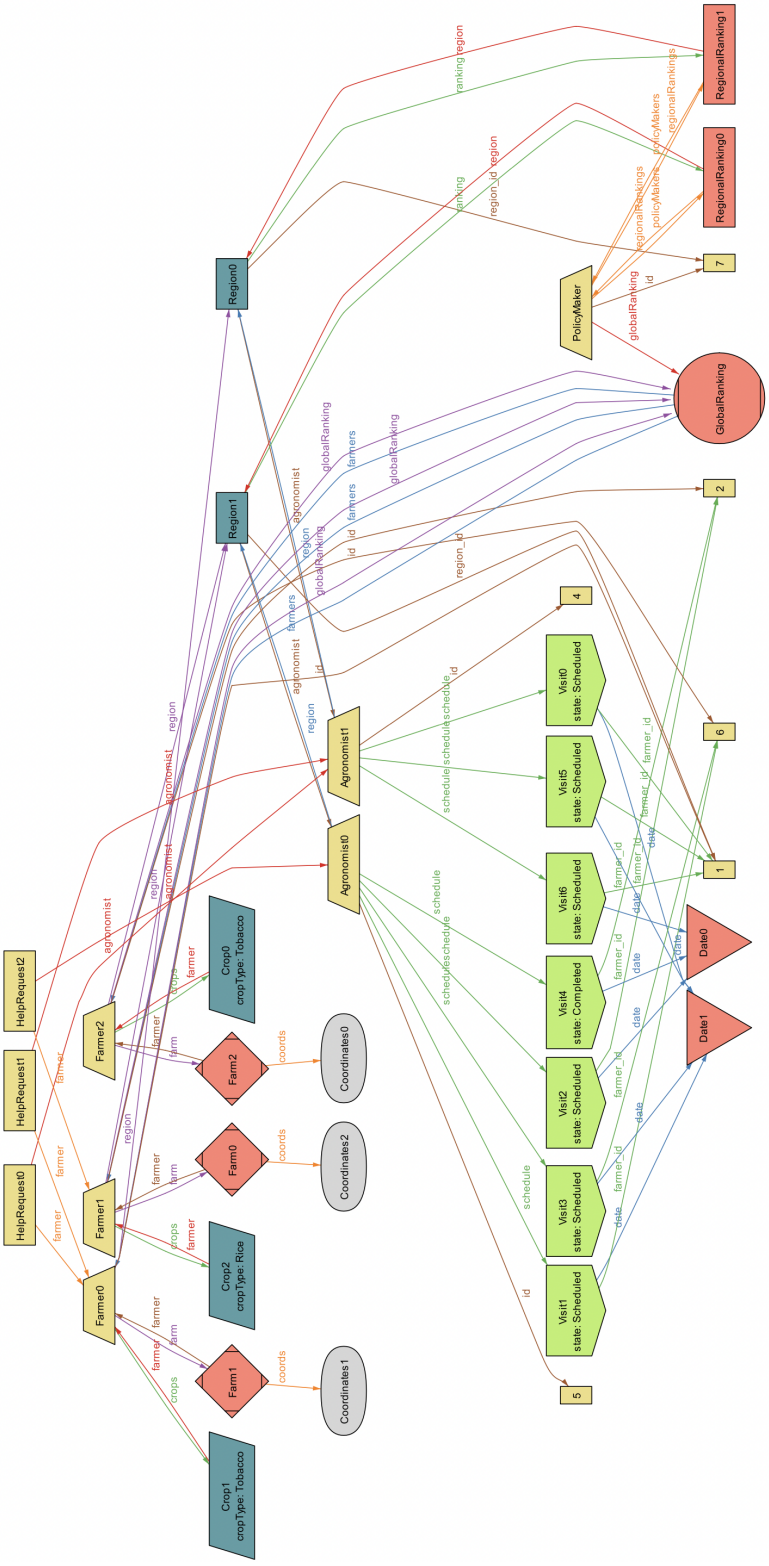


Figure 4.2: Second world generated using alloy tool

Chapter 5

Effort Spent

5.1 Mariani Samuele

Task	Hours
Initial briefing	2
Introduction	5
Product functions	8.5
Mapping on goals	3
Design Constraints, System Attributes	3.5
Alloy	12
Revision	6
Total	40

5.2 Molteni Alessandro

Task	Hours
Initial briefing	2
Introduction	5
Scenarios	6
User Characteristics	1.5
Functional Requirements	4
Use Cases	6.5
Sequence Diagrams	5
Mapping on requirements	2
Alloy	2
Revision	6
Total	40

5.3 Monti Matteo

Task	Hours
Initial briefing	2
Introduction	5
Class Diagram	4
Domain Assumptions	2
Interface Requirements	9
Use case diagrams	2
Activity diagrams	2
Alloy	8
Revision	6
Total	40

Chapter 6

References

- Course slides on WeBeeP.
- Course lectures notes.
- ISO/IEC/IEEE 29148 - Systems and software engineering.
- All *class*, *activity*, *use case* and *sequence* diagrams were made using the **draw.io** online tool.
- The Alloy code was made using the **Alloy analyzer** tool.