POLITECNICO
MILANO 1863

Software Engineering 2 Project
Academic Year 2021 - 2022

DREAM

Requirements Analysis and Specifications Document

Valeria Detomas Sofia Martellozzo

Professor

Elisabetta Di Nitto

Version 1
December 21, 2021

Contents

1 Introduction

1.1 Purpose

The purpose of this document is to thoroughly describe Data-dRiven PrEdictive FArMing in Telengana(DREAM). It presents functional and non functional requirements of the system and its components. Moreover it provides use cases and scenarios for the users involved.
This document is meant as a contractual basis for the customer and the developer.

1.1.1 Goals

G1	Allow policy makers to retrieve information from farmers and to evaluate their performance
G2	Allow farmers to communicate with each other
G3	Allow farmers to insert data and advices on his production
G4	Allow farmers to send request of help to Policy Makers
G5	Allow farmers to retrieve information relevant for their activity
G6	Policy Makers and farmers should be able to consult the map of the zone (and the informations stored in it) with different levels of visibility
G6A	Policy Makers and farmers should be able to see the position of the farms on the map
G6B	Policy Makers should be able to see both the information about the production and the evaluation of each farm
G6C	Farmers should be able to see only the type of production of a farmer by the map

Table 1: Definition of goals

1.2 Scope

The aim of the system is to acquire and combine data and information of farmers in Telengana. The system will provide support both to Telengana's policy makers and to farmers thanks to new innovative technologies.
Through the system, policy makers are able to get a complete picture of the agriculture status in the whole state. In order to obtain this, Dream provides information that makes policy makers able to give incentives to those farmers who are performing well. Moreover it allows them to keep track of those farmers who need help.
The farmers have access to a forum where they are able to communicate with other farmers. The aim of the forum is to share useful suggestions and to let

farmers who struggle with something ask for help.
Therefore on the one side policy makers can have an entire perspective of the farms' situation in the entire state and on the other side farmers can take advantage of the application and discuss with their colleagues.

Hence the application is used by the policy makers as a way to monitor farmers. Through the system they are able to search a farm and have access to its general information. Once a month they also send each farmer an evaluation message where they specify if the farmer activity was performed good or bad. On the contrary farmers can ask for help or write advices. They are free to write messages and communicate with other farmers through a forum that is created specifically for them.

1.2.1 World Phenomena

- Farmer starts growing some type of crop
- Weather conditions influence production
- Telengana's state collects data concerning meteorological forecasts
- Farmer uses water irrigation system
- Farmer is struggling with his harvest

1.2.2 Shared Phenomena

- Farmer access Dream
- Farmer sends a message in the forum
- The amount of water used by the farmer is registered on the application
- Farmer inserts data about his production in the system
- Farmer sends a help request to the policy maker
- A user visualizes the map that identifies the farms
- Policy maker evaluates farmers' performances
- A user checks notifications

1.3 Glossary

1.3.1 Definitions

- **DBMS** is a software that works as an interface between the end user and the database. It manages the data, the database engine and the database schema.

- **HTTPS** is a protocol where encrypted HTTP data is transferred over a secure connection. It also guarantees the privacy and integrity of data.
- **API** is a programming code that helps communicate two different computer programmes.

1.3.2 Acronyms

- **API**: Application Programming Interface
- **HTTPS**: Hypertext Transfer Protocol Secure
- **DBMS**: Data Base Management System
- **UML**: Unified Modelling Language

1.3.3 Abbreviations

- **Gn** goal number n
- **Rn** requirement number n
- **DAn** domain assumption number n
- **UCn** use case number n
- **Sn** scenario number n

1.4 Revision History

- December 20, 2021: version 1.0 (First release)

1.5 Document Structure

1. **Introduction**
   This section offers an introduction and a brief overview of the system that is presented in the document. It highlights the purpose of the system and the goals that are meant to be achieved with it. At the end there is also a glossary that contains a list of definitions, acronyms and abbreviations.
2. **Overall Description**
   This section starts with a product perspective that contains a description of the system's domain through a class diagram. It includes also state diagrams which are used to give more details about the behavior of some objects in the model. The section contains also a clear description of the features offered by the system, it identifies the actors involved and it describes their characteristics. At the end there are domain assumptions and general constraints.

3. **Specif Requirements**
   This section enters into the details on how the system interacts with the external world. It describes the interfaces that are required and offered through several visual mockups. Moreover the section provides functional and nonfunctional requirements. Functional requirements are additionally described by use cases, sequence diagrams and scenarios. At the end the section focuses on nonfunctional requirements and various limitations that the system might face.

4. **Formal Analysis using Alloy**
   This section provides the model described through Alloy language.

5. **Effor Spent**
   This section has a record of the hours spent to complete this document.

2 Overall Description

2.1 Product Perspective

2.1.1 Class Diagram

The class diagram represented in Figure 1 represents the domain of the system. The main elements in the class diagram are:

- **User**: identifies two types of users who can access the application.
  Farmer and *Policy Maker* are the two categories of users that are present in the system.
  The distinction has to be applied in order to have different permissions.
  Both have a password used for the autentication and an email, in the case of a farmer, or an alphanumeric code owned by a policy maker.

- **Farm**: stores all the information related to a Farm.
  It can be owned only by a farmer, and is located in a specific position visible by the map.
  It contains also the data about the production inserted by the farmer himself, that could be not present yet. (if a farmer has not started yet or has not inserted these information).
  Each farm is provided of sensors of two kind: one that collect data about the amount of water used by the farmer in his farm and humidity every day.

- **Notification**: this class represents content sent between users (in order to communicate something) or something submitted in the sistem that is stored in the database.

  1. *Help*: it stores the single farmer that writes it, the type of production he selects and a body where hthe farmer explains in details the problem.
     When this notification is sent (to the policy makers) the system registers in it the actual date and time.
  2. *Advice*: the structure of it is the same as the Help class but it is not sent to anyone, it is immediatly stored in the database.
  3. *Evaluation*: this class represents the evaluation on exacly one farm by a single policy maker.
     Each policy maker can evaluate one or more farms more than once since this event happens once a month.
     Despite it, an evaluation is related only to one farm and the result can be positive (1) or negative (0).
     The evaluation is sent by the policy maker to the farm owner.
  4. *Solution*: this class is like the Help one, in fact is the reply of a policy maker to a help request (by Help notification) from a farmer with in the body the solution found by the sender.

- **Map**: this object contains a list of all the farms located in Telegana and registered in the application.
  It stores all the information about the farms due to provides them to the users (in different ways).
- **Weather Info**: this class is a rapresentation of the meteorological information of the country, provided by an external application.
  These information are specific for each position in which is located a farm.
- **Forum**: here are stored all the messages sent by the farmers on the forum page.
  It allows the system to make them always visible to all the farmers.
  In addition to that the messages are also ordered by date and time in which they were sent.

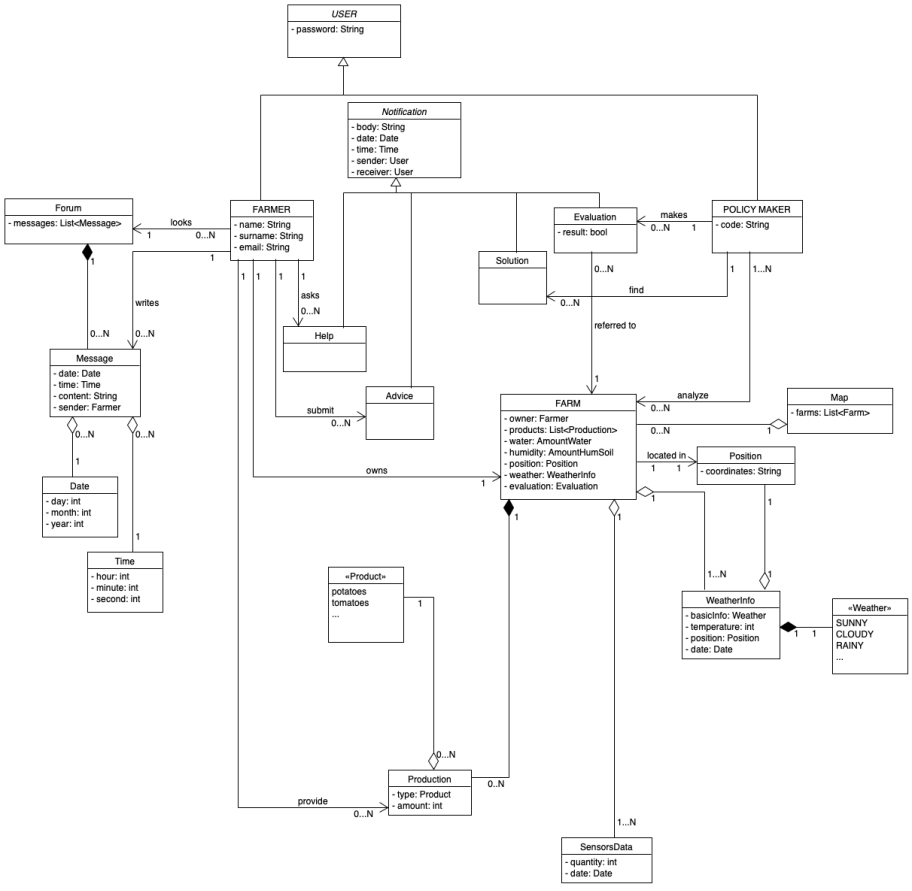

Figure 1: UML diagram.

2.1.2 State Diagrams

The following state diagrams describe the behavior of the main objects of the system's domain previously described(Figure 1). They consider all the potentional states that the element examined can have while a certain event takes place.

1. **Update of the Farmer Page** The state diagram represented in figure 2 represents the events that might occur whenever the farmer page is updated. The three main circumstances when the page is updated are:

   - Farmer inserts new production data
   - Every day the weather forecast is updated with the new data available on it
   - Data collected from sensors are always up to date

   The page is so update after one of this events occurs.

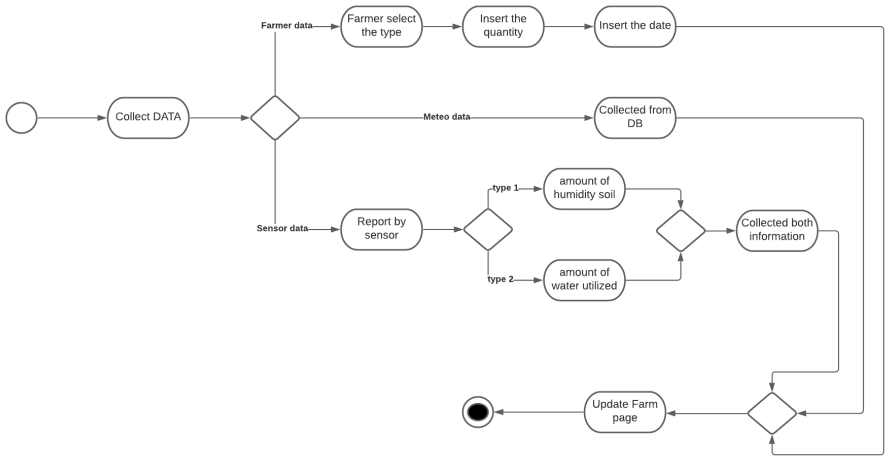

Figure 2: Update Farmer page.

2. **Analysis of different farmers** The state diagram in figure 3 represents the analysis that is performed by the policy makers once a month. As a matter of fact the policy maker starts the analysis after selecting the farm, he visualizes then the farmer's farm page and analyzes the information that is present in it. After the anlaysis the map is updated and the system sends a notification, that is then different if the farmer's performance is either good or bad.

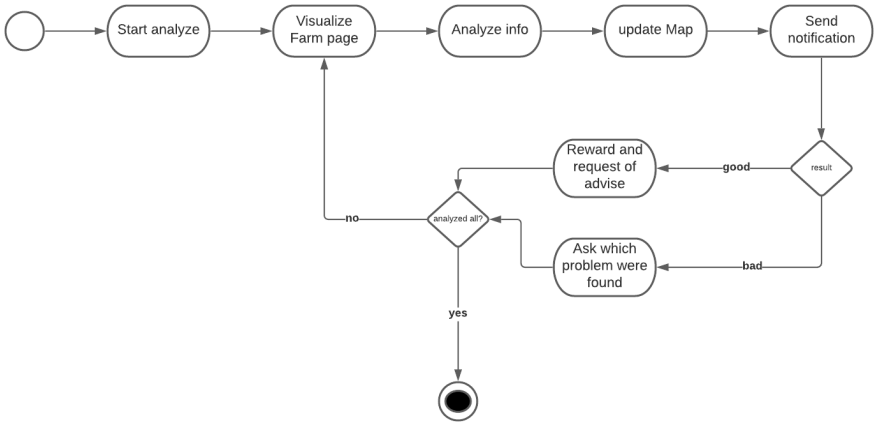

Figure 3: Analysis of farmers.

3. **Request of help from a farmer**

4. The diagram represented in Figure 4 shows the events that takes place when a farmer sends a request of help. It can happen by writing a message in the forum or by sending a request through a form in the homepage.

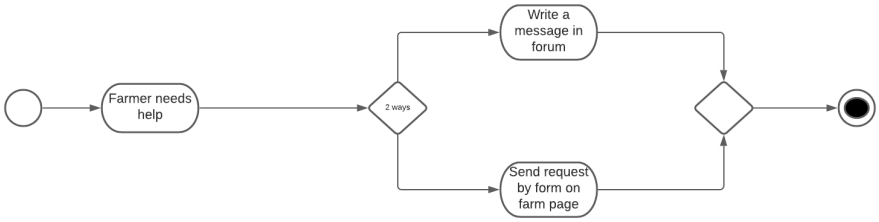

Figure 4: Request of help.

2.2 Product Functions

This section provides a summary of the main features and functions offered by the software regarding the goals already described in section 1.1.1

In the following description it is important to highlight that both the policy makers and the farmers must be logged in.

2.2.1 Farmers insert data

This functionality is accessible to all farmers. The application provides a form in which the farmer can easily insert data of his/her production. The form is easy to fill in, in order to complete it the farmer need to indicate:

- the **type of product**
- the **amount** producted of the selected type
- the **date** relative to the date of the production

If the farmer needs to add more than one type of product, he/she can fill in the form multiple times. After completing the form the user is redirected to the homepage and the policy makers can see the updated data. This functionality can be done more than once a day since the farmer can select the date, so it is possible for him to insert data of past days too. This operation can be repeated more than once a day since the farmer can select the date, so it is possible for him/her to insert data of past days too.

2.2.2 Farmers visualize data

This functionality lets the farmer visualize all the data acquired from the system. The farmer can visualize all data on his homepage.
The application shows:

- meteorological short-term and long-term forecast
- amount of water used by the farmer
- humidity of soil
- personalized suggestions concerning specific crops to plant or specific fertilizers to use – based on their location and type of production

Should I say why he needs to visualize this data or how he uses it

This functionality is always up to date, and does not need any input from the farmer. It is used by the farmer only to have a general view of his farm and on how he could improve the productivity of his farm.

2.2.3 Identify how farmers are performing

The main features of the policy makers is to evaluate the work of each Farmer. In order to do that, they periodically analyze each farm page: the system allows them to visualize all the data in those pages (but not to modify it). The analysis takes place twice a month. With this analysis they classify the workers in two different way:

- GOOD farmer : those how have been able to produce a significant amount of product with low resources, despite bad weather in that period.
- BAD farmer : those who did not produce much.

Policy makers inform each farmer the result they have achieved with a notification:

- GOOD farmers receive a special incentive, and also a request to submit from their personal web page some advice that could be useful to the others.
- To BAD farmers is asked to submit an explicit request of help specifying the problems they had.

The system has a specific web page that allows all the users to look at a map of the area in which are specified all the farms. It is also shown if the farm's owner has performed a good job in that period. At the end of each analysis the policy makers update the map (they are the only ones that are able to modify it).

2.2.4 Interaction between farmers

This functionality permits the farmers to comunicate with each other. The application has a specific web page were the farmers can send messages whenever they want. If a farmer has an issue, before submitting a formal request of help to the policy makers by their home page, he can ask informally an advice by sending a message in the Forum. It is not necessary to be a good farmer in order to answer someone elses message. All the messages are visible to everyone and 24h. Since it is an online application an internet connection is required to read or write on this page.

2.3 User Characteristics

Dream has two different customers that need to be distinguished in order to provide the various features specified in subsection 2.2.

2.3.1 Farmer

- Can register on Dream in order to be recognized as farmer
- Must log in on the website to use the services offered

- Can discuss with other farmers
- Is able to insert data about their daily production
- Can ask for help to other farmers or to policy makers
- Can retrieve data regarding weather forecast, water irrigation system or humidity of soil
- Can look for advice of several products
- Can check whether their performance is identified as good or bad

2.3.2 Policy Maker

- Already has the credential to access to the system
- Must login to Dream to benefit of its services
- Can look all farms' pages
- Can update the map
- Can send suggestions to whom explicitly notices a problem
- Can send notifications to the farmers
- Decides the value of the incentive for the good farmers
- Evaluates the performance of the farmers

2.4 Domain Assumptions, Dependencies and Constraints

This subsection focuses on what it is assumed in order for our system to offer the services as expected. Moreover it focuses on the limitations that the system could face.

2.4.1 Domain Assumptions

DA1 In order to access the system users need to have Internet connection.
DA2 Farmers always insert correct data on their production activity.
DA3 Data from sensors is always correct.
DA4 Date and Time on the system are always correct.
DA5 The position of the Farm is always correct.
DA6 Internet connection works always without errors.
DA7 Meteorological data is accurate.
DA8 Every farm has a different position.
DA9 Each farm belongs to exacly one farmer.
DA10 Discussion on the forum are related only to the farm activity.
DA11 Formal request of help must be related to a farmer's own production.
DA12 Advice on a product must be given by a farmer that produces the same

type.
DA13 Performances of farmers are always identified correctly.
DA14 Farmers can insert data more than once a day.
DA15 The email and farm's name must be unique.
DA16 Policy Makers have a given code and password to access the system.
DA17 All farmers that sign up own a real farm in Telengana.
DA18 The special incentive that the farmers receive for their good work, is used on stuff related to their farm activity.

2.4.2 Constraints

3 Specific Requirements

3.0.1 External Interface Requirements

This section of the document gives a clear description of all the requirments that the system needs in order to performe all the functionalities described in section 2.2.

3.0.2 User Interfaces

The following mockups of the web application provide a first idea on how the features of the system are offered to the users. Moreover it gives a real perpespective of how the users interact with the system. The Design Document (reference) provides a thorough and detailed description of the features represented in each mockup.

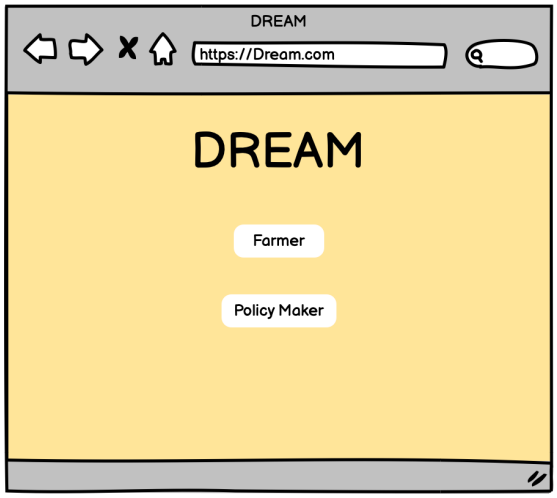

Figure 5: *DREAM* Web page

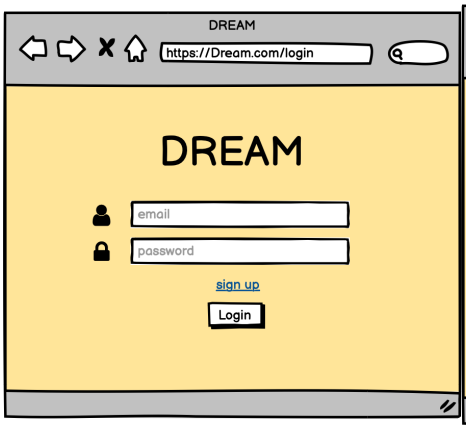

Figure 6: *Farmer Log in* Web page

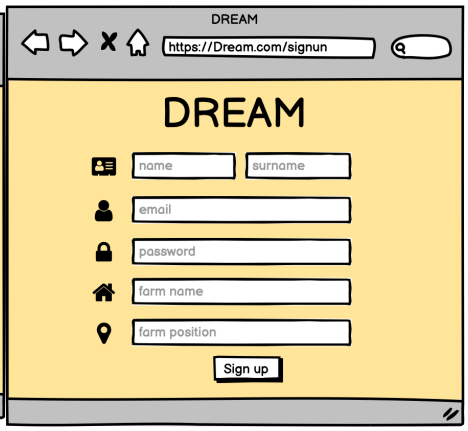

Figure 7: *Farmer Sign up* Web page

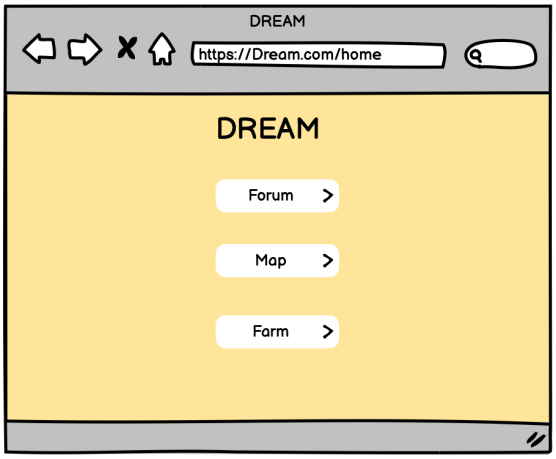

Figure 8: *Farmer* Home page

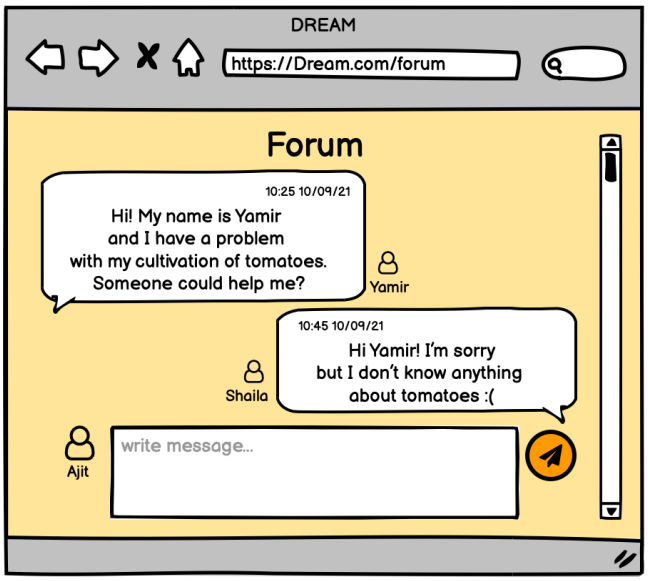

Figure 9: *Forum* Web page

Figure 10: *Farmer* Map page

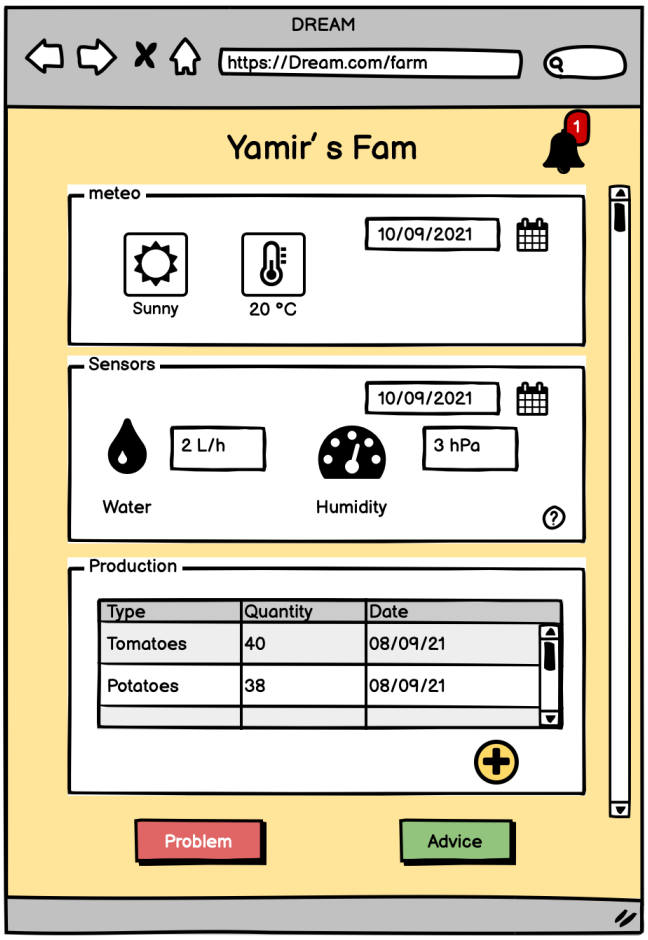

Figure 11: *Farmer's* Farm page

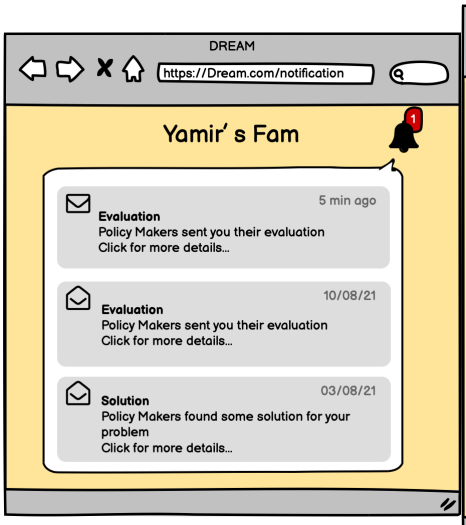

Figure 12: *Farmer's notifications list*

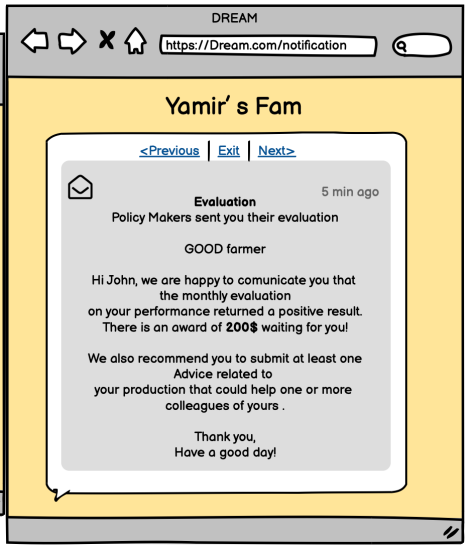

Figure 13: *Farmer's single notification*

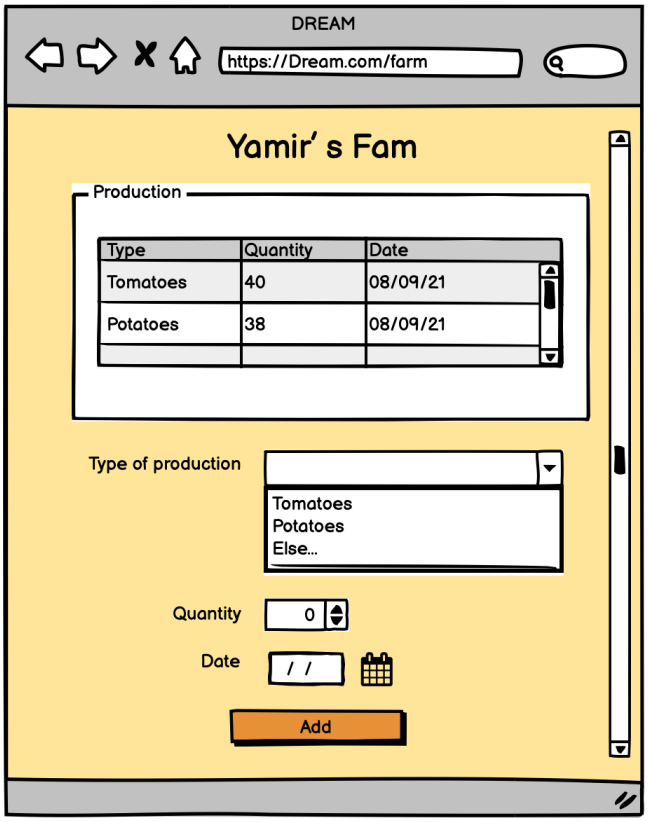

Figure 14: *Farmer add data* Web page

Figure 15: *Farmer send request of help* Web page

Figure 16: *Farmer submit advice* Web page

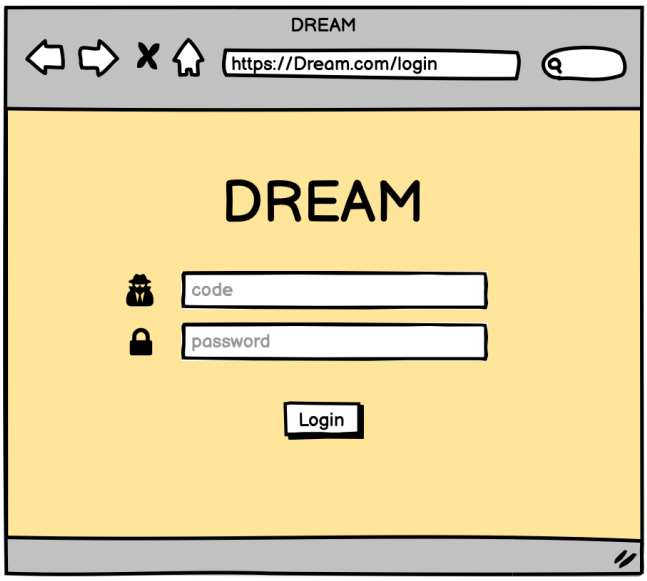

Figure 17: *Policy Maker Log in* Web page

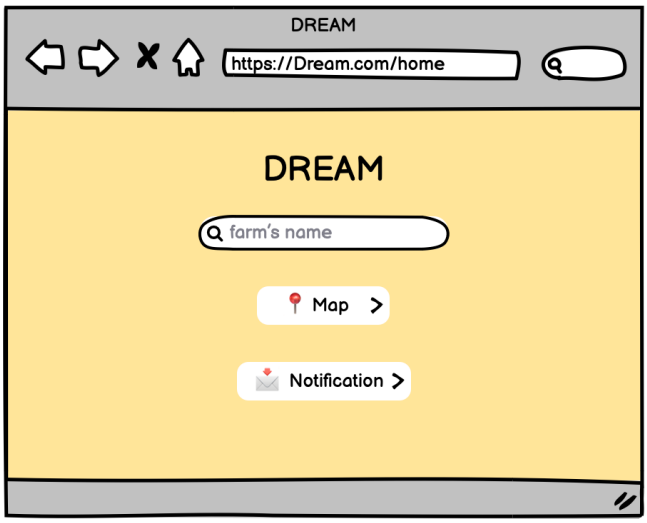

Figure 18: *Policy Maker* Home page

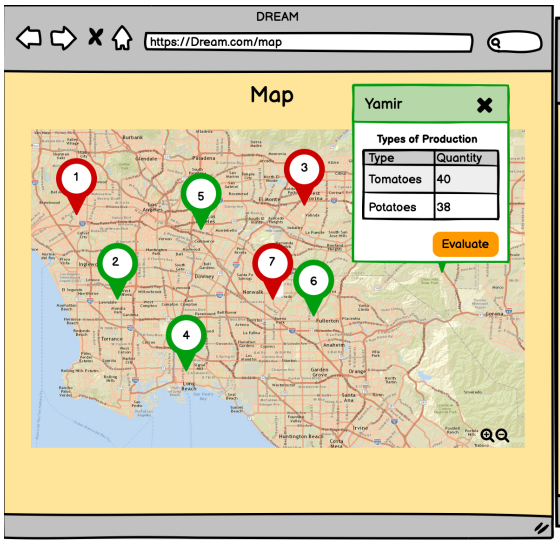

Figure 19: *Policy Maker* Map page

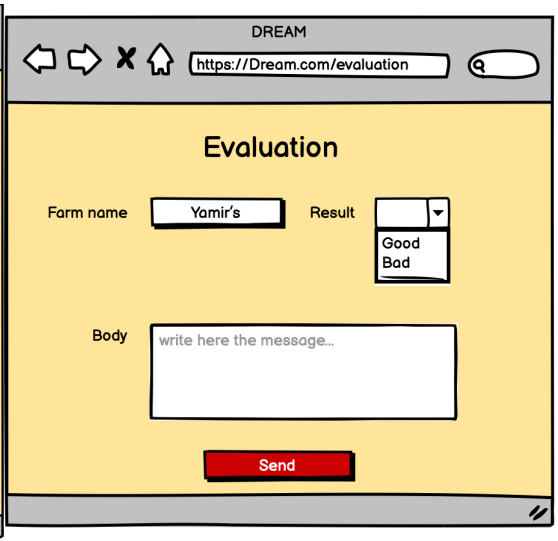

Figure 20: *Policy Maker* evaluation form

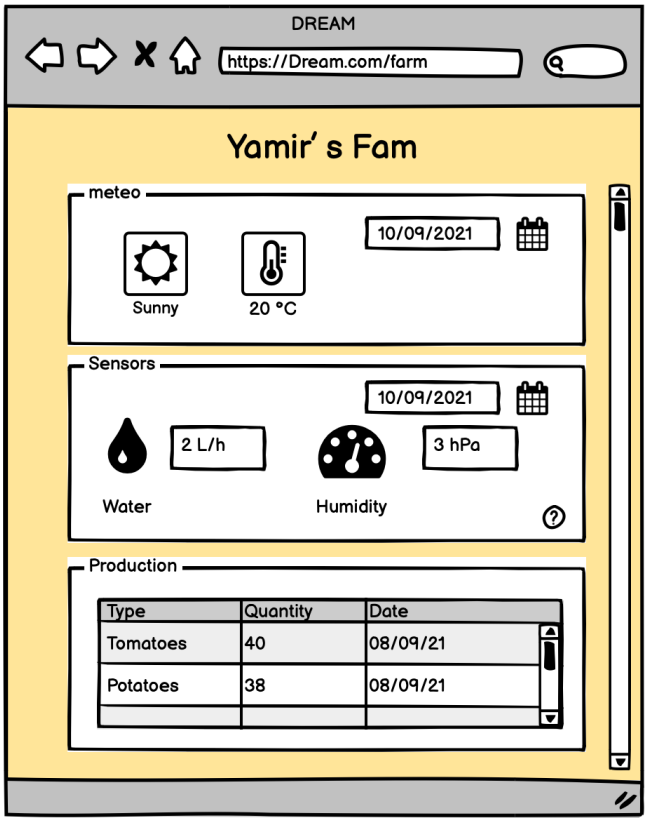

Figure 21: *Policy Maker* Farm's page visualization

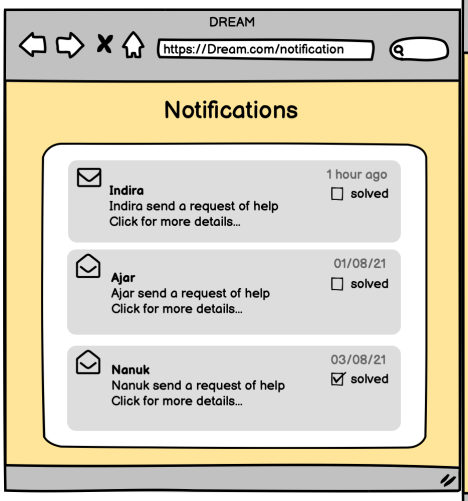

Figure 22: *Policy Maker* notifications list

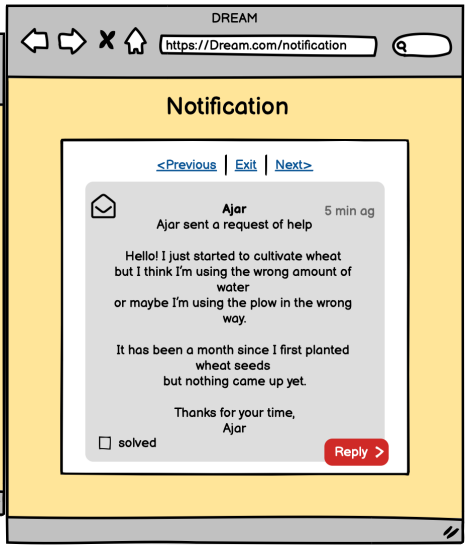

Figure 23: *Policy Maker* notification view

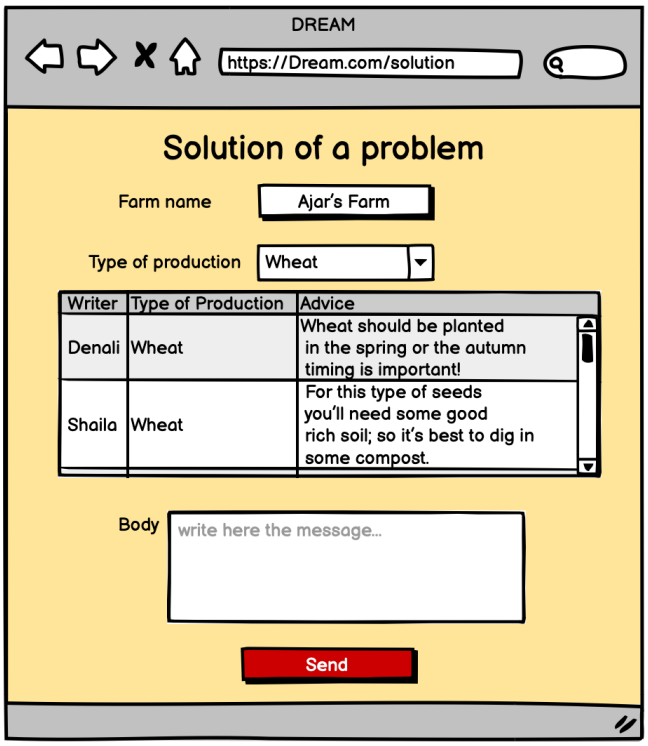

Figure 24: *Policy Maker* send solution page

3.0.3 Hardware Interfaces

The application does not need any specific hardware requirements.

3.0.4 Software Interfaces

The web app requires a computer with a web browser installed and connected to The system has to rely on a DBMS API. It allows the management of all the data the system needs in order to provide its functionalities, described in subsection 2.2.

3.0.5 Communication Interfaces

All the communications between users and Dream website are made via HTTPS.

3.1 Functional Requirements

3.1.1 Use Case Diagram

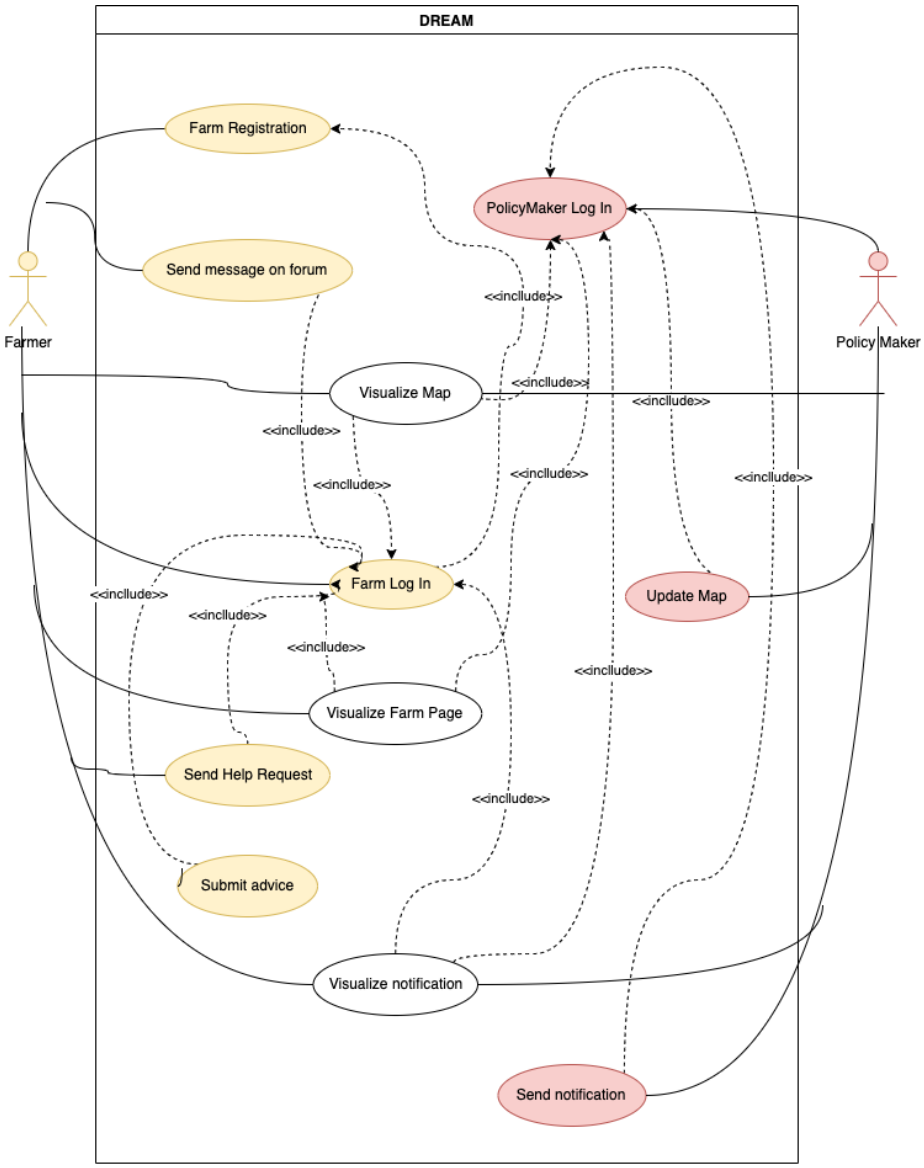

Figure 25: Use Case diagram.

3.1.2 Use Cases Description ad Sequence Diagram

1. **Farmer Registration**

Name	**Farmer Registration**
Actors	Farmer
Entry conditions	The web application has started
Flow of events	(a) The farmer wants to sign up (b) The farmer inserts email, name, password, farm's name and farm's position (c) The farmer clicks submit (d) The system checks if email is unique and if all the form is correctly fill up (e) The system inserts the information in the data base
Exit conditions	The farmer is signed up
Exceptions	• If the farmer did not insert data correctly the system will send an alert and let the user do that again • If the email is already present in the database the system will send an alert saying that the email already exists

Table 2: *Farmer Registration* use case description

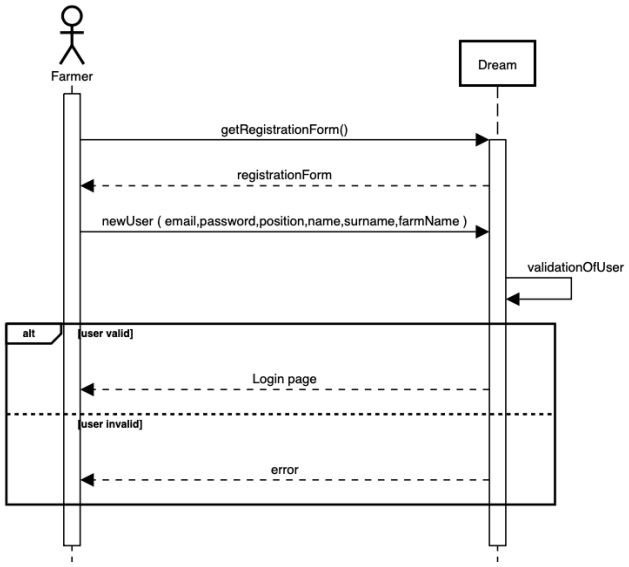

Figure 26: *Farmer Registration* sequence diagram

2. **Farmer Login**

Name	**Farmer Login**
Actors	Farmer
Entry conditions	The web application has started
Flow of events	(a) The farmer wants to log in (b) The farmer inserts email and password (c) The farmer clicks submit (d) The system checks if the credentials are correct (e) The system notifies the farmer about the correct login
Exit conditions	The farmer has logged in

Exceptions	• If the system does not recognize the email it will send and alert to the farmer saying that the email inserted is wrong • If the password is not correct the system will notify the farmer

Table 3: *Farmer Login* use case description

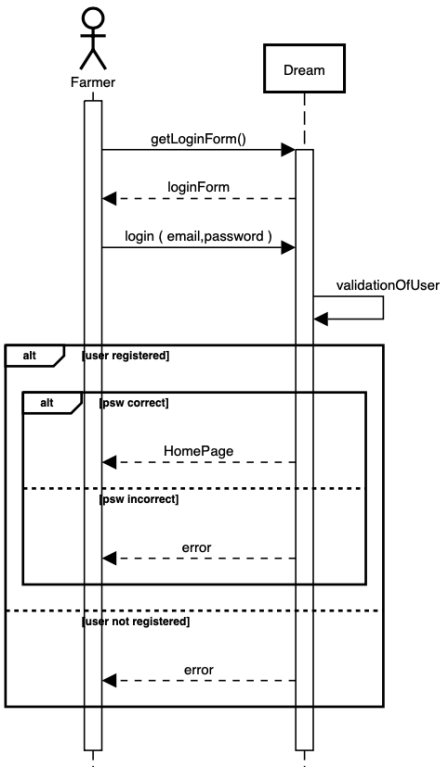

Figure 27: *Farmer Login* sequence diagram

3. **Farmer send a message on the Forum**

Name	**Farmer sends a message on the forum**
Actors	Farmer
Entry conditions	The farmer has logged in

Flow of events	(a) The farmer wants to send a message (b) The farmer clicks on forum button (c) The system send the ser to the forum page (d) The farmer inserts the message (e) The farmer clicks on send message (f) The system inserts the message into the database
Exit conditions	The farmer's message is published
Exceptions	• If the message body is empty the system shows an error alert

Table 4: *Farmer message* use case description

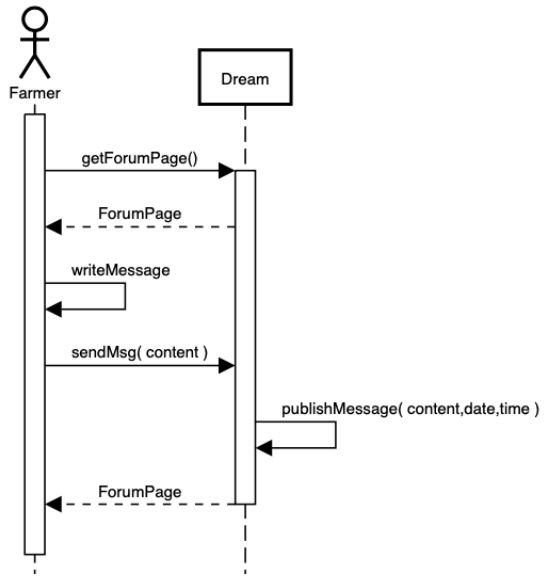

Figure 28: *Farmer message* sequence diagram

4. **Submit production data**

Name	**Farmer submit production data**

Actors	Farmer
Entry conditions	The farmer has logged in
Flow of events	(a) The farmer wants to add new data about his production (b) The farmer clicks on his own farm page button (c) The clicks on Add data button (d) The farmer selects the type of production on which he wants to add information (e) The farmer selects the quantity of product cultivated (f) The farmer selects the date in which he reaps (g) The farmer clicks on submit button (h) The system inserts the data in the database and it redirects the farmer to the updated page
Exit conditions	The farmer's data are stored
Exceptions	• If the type of production is empty the system shows an error alert • If the quantity is not specified the system shows an error alert • If the date is not selected the system shows an error alert

Table 5: *Submit data* use case description

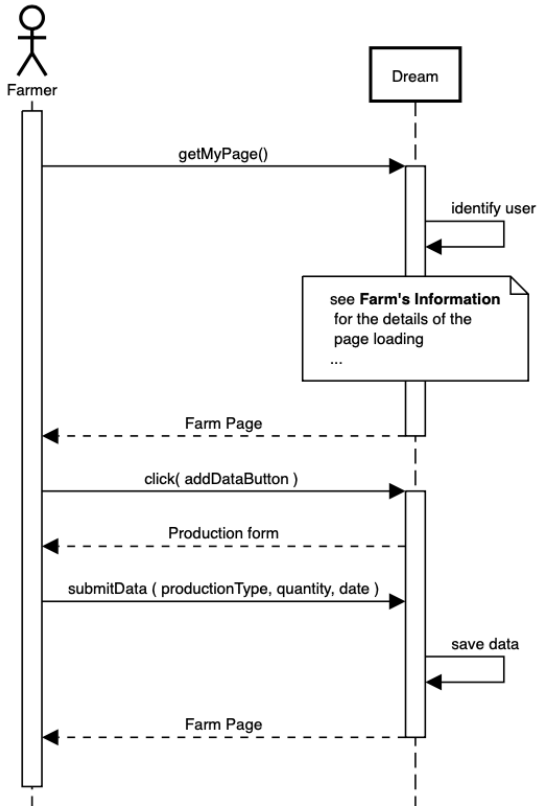

Figure 29: *Submit data* sequence diagram

5. **Find a farmer on the Map**

Name	**Farmer visualizes the map**
Actors	Farmer
Entry conditions	The farmer has logged in

Flow of events

(a) The farmer wants to visualize the map

(b) The farmer clicks on map button

(c) The system send the farmer to the map page

(d) The farmer visualizes the map and selects a farm

(e) The system retrieves the information about the farm and shows them to the farmer

(f) The farmer visualizes the data about the selected farm

Exit conditions	The farmer visualized the map
Exceptions	

Table 6: *Farms' map visualization* use case description

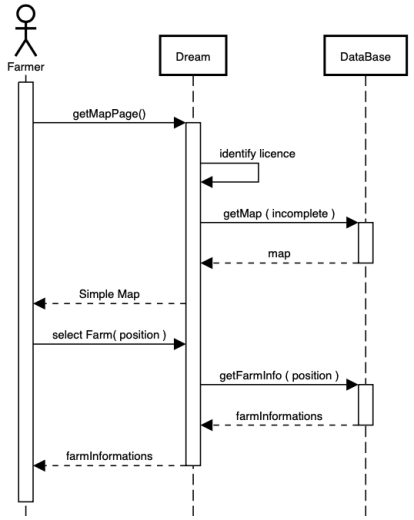

Figure 30: *Farms' map visualization* sequence diagram

6. **Find farm's information**

Name	**Farmer visualizes their own data**
Actors	Farmer
Entry conditions	The farmer has logged in

Flow of events	(a) The farmer wants to visualize his own page (b) The farmer clicks on the profile page button (c) The system retrives the information about the farmer and redirect him to the farmer's farm page (d) The farmer visualizes his own page
Exit conditions	The farmer visualizes his own data
Exceptions	(a)

Table 7: *Farm information visualization* use case description

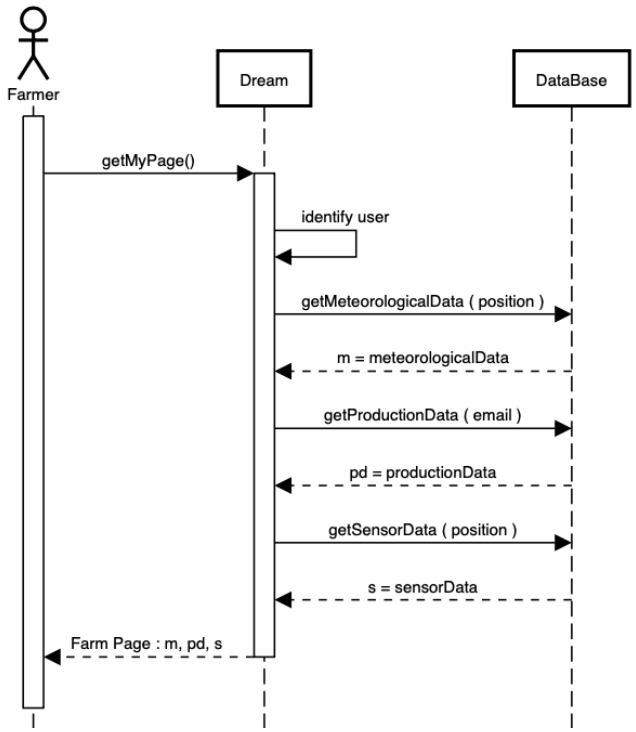

Figure 31: *Farm information visualization* sequence diagram

7. **Submit a request of help I put only the case of a forml request**

to the Policy Maker (button on Farm Page) if you want we can create 2 "secion" (alt) one with this formal request and one sending a message on Forum (but is yet specified in 3)

Name	**Farmer submit a request of help**
Actors	Farmer
Entry conditions	The farmer has logged in
Flow of events	(a) The farmer wants to ask for help (b) The farmer clicks on his own page button (c) The farmer clicks on Help button (d) The farmer chooses the type of production on which he had problems (e) The farmer fills the form specifying the problem (f) The farmer clicks on the submit button and sends the message (g) The system sends the notification to the Policy Makers (h) The system notifies the farmer that the operation went succesfully (i) The system returns a list of advices retrieved from the database about the type of production selected
Exit conditions	The help request is sent to the Policy Makers
Exceptions	• If no type of production is selected the system notifies the farmer and waits for him to insert it • If the request body is empty the system notifies the farmer and waits for him to fill it in

Table 8: *Farmer ask for help* use case description

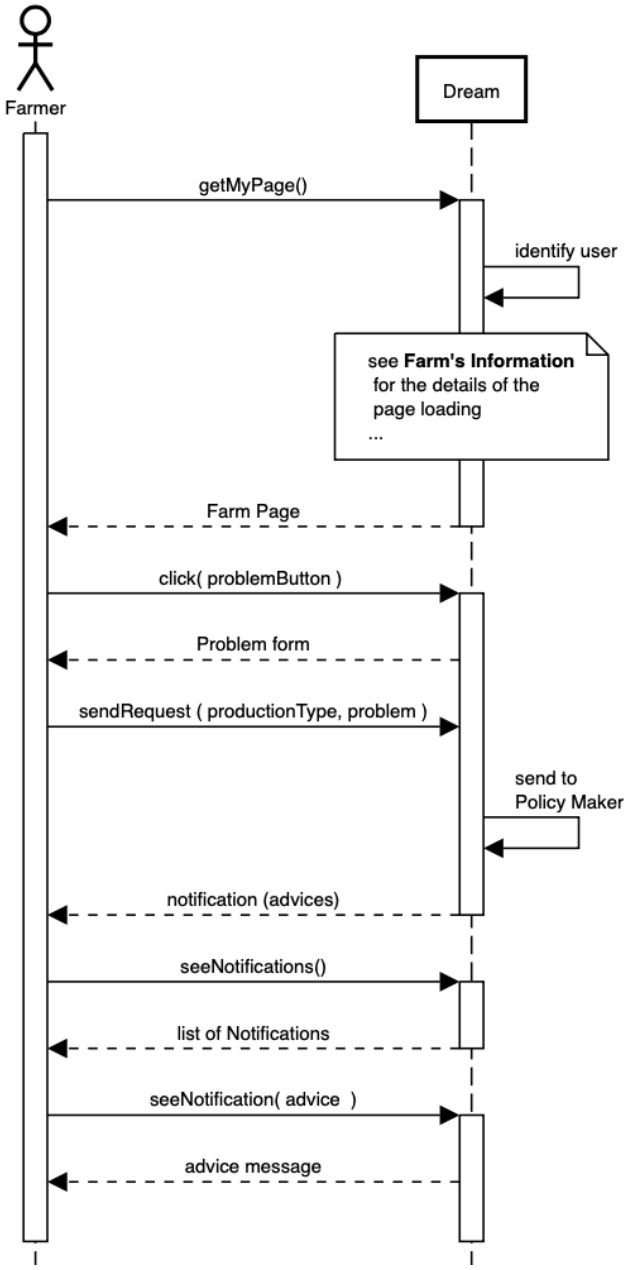

Figure 32: *Farmer ask for help* sequence diagram

8. **Submit an advice**

Name	**Farmer submits an advice**
Actors	Farmer
Entry conditions	The farmer has logged in
Flow of events	(a) The farmer has to submit an advice (b) The farmer clicks on his own page button (c) The farmer clicks on Advice button (d) The farmer chooses the type of production on which he wants to give advices (e) The farmer fills the form writing his advice (f) The farmer clicks submit button and sends the message (g) The system saves the advice in the database (h) The system notifies the farmer that the operation went succesfully
Exit conditions	The farmer submit an advice
Exceptions	• If the farmer is not a "good" one the system doesn't save the advice and nottifies the farmer with an error message

Table 9: *Farmer send advice* use case description

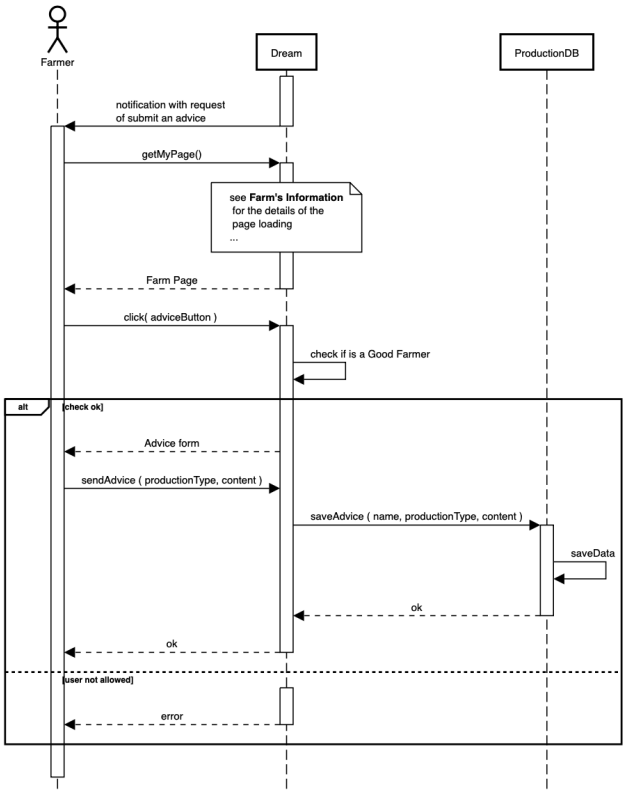

Figure 33: *Farmer send advice* sequence diagram

9. **Visualize notifications**

Name	**Farmer visualizes his notifications**
Actors	Farmer
Entry conditions	The farmer has logged in

Flow of events	(a) The farmer wants to read his notifications (b) The farmer clicks on his own page button (c) The farmer clicks on Notification button (d) The system returns a list of messages (e) The farmer selects one message between the ones in the list (f) The system returns the body of the message
Exit conditions	The farmer visualizes his notifications
Exceptions	•

Table 10: *Farmer visualize notifications* use case description

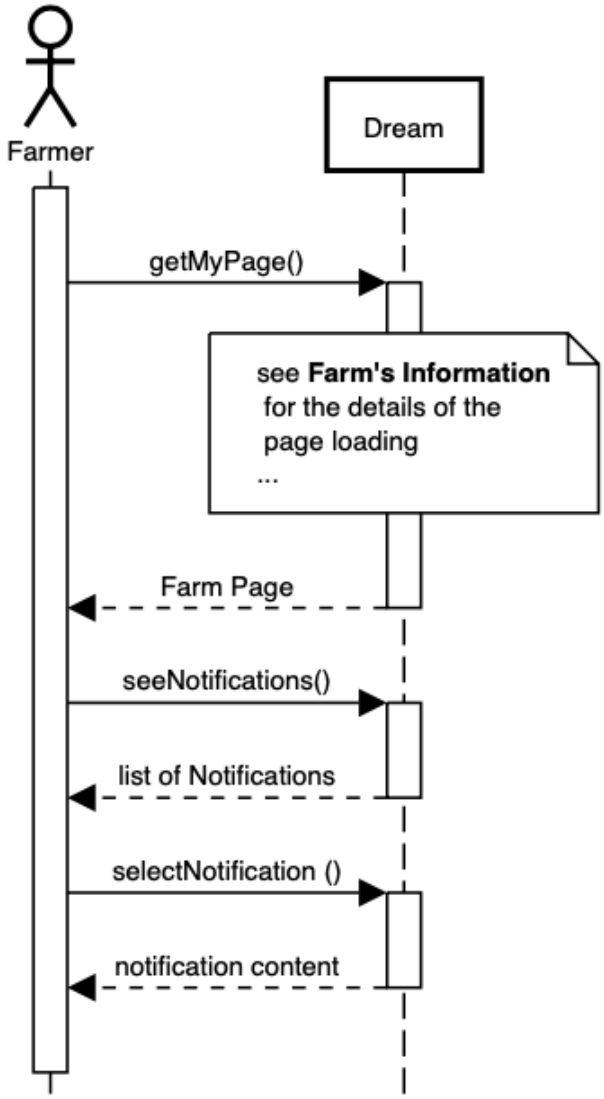

Figure 34: *Farmer visualize notifications* sequence diagram

10. **Policy Maker login**

Name	**Farmer visualizes their own data**
Actors	Policy Maker
Entry conditions	The web application has started

Flow of events	(a) The Policy Maker wants to log in (b) The Policy Maker inserts code and password (c) The Policy Maker clicks the submit button (d) The system checks if the credentials are correct (e) The system notifies the Policy Maker about the correct login
Exit conditions	The Policy Maker has logged in
Exceptions	• If the system does not recognize the code it will send an alert to the Policy Maker saying that the code inserted is wrong • If the password is not correct the system will notify the Policy Maker

Table 11: *Policy Maker login* use case description

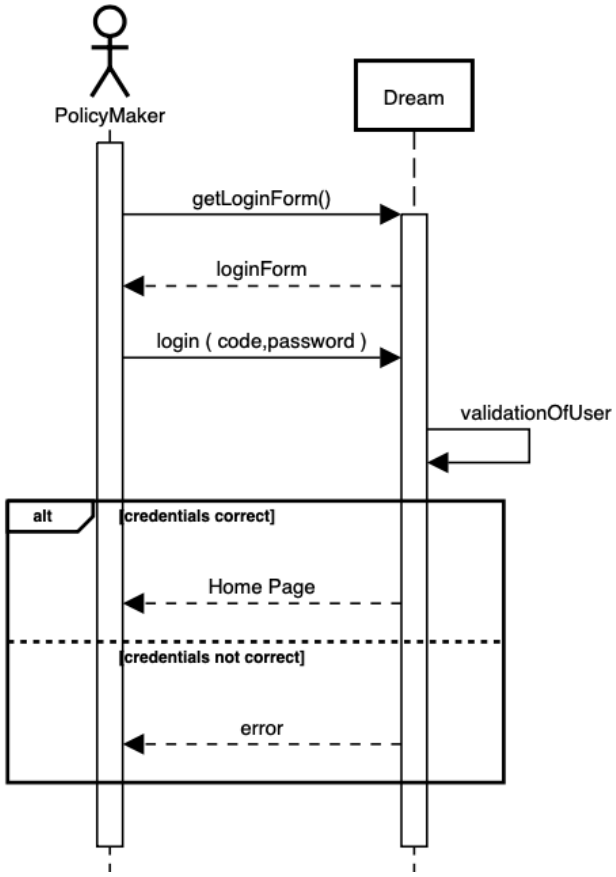

Figure 35: *Policy Maker login* sequence diagram

11. **Find a farm's page**

Name	**Policy Maker visualizes a farm's page**
Actors	Policy Maker
Entry conditions	The Policy Maker has logged in

Flow of events	(a) The Policy Maker wants to visualize a farm's page (b) The Policy Maker types the name of the farm on the search form (c) The Policy Maker clicks search button (d) The system redirects the Policy Maker to the farm's page
Exit conditions	The Policy Maker visualizes the farm's page
Exceptions	• If no name is inserted in the search form the system notifies the Policy Maker about the missing data • If the name inserted is not present in the system's database, the system sends an error message

Table 12: *Farm's page visualization* use case description

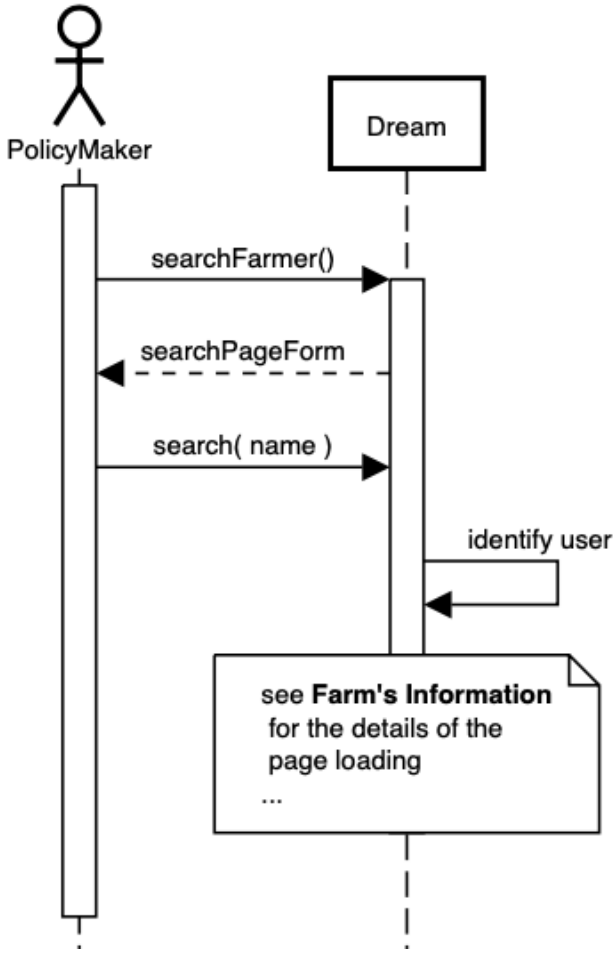

Figure 36: *Farm's page visualization* sequence diagram

12. **Find a farmer on the Map**

Name	**Policy Maker visualizes a farm's page**
Actors	Policy Maker
Entry conditions	The Policy Maker has logged in

Flow of events	(a) The Policy Maker wants to visualize the map (b) The Policy Maker clicks on map button (c) The system redirects the Policy Maker to the map page (d) The Policy Maker visualizes the map and selects a farm (e) The system retrieves the information about the farm (also the last evaluation) and shows them to the Policy Maker (f) The Policy Maker visualizes the data about the selected farm
Exit conditions	The Policy Maker visualizes his own data
Exceptions	•

Table 13: *Farm visualization on map* use case description

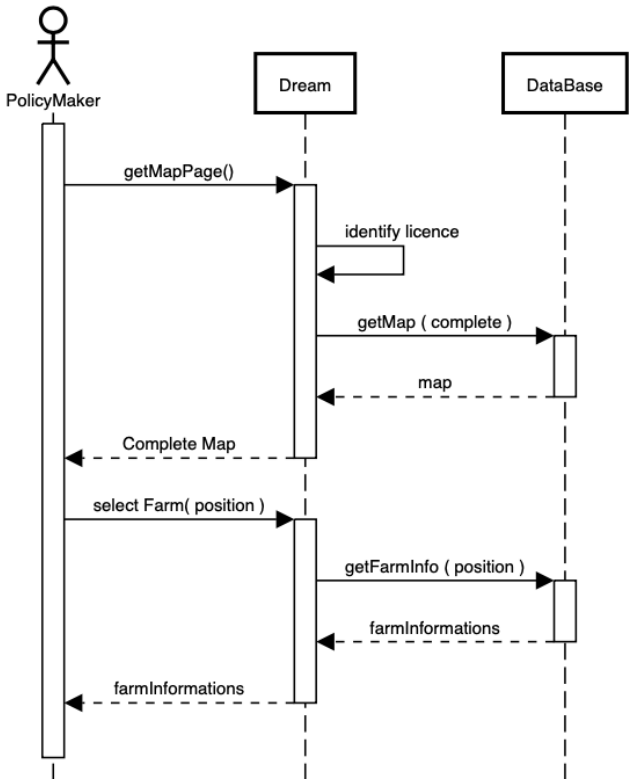

Figure 37: *Farm visualization on map* sequence diagram

13. **Update the Map**

Name	**Policy Maker update the map**
Actors	Policy Maker
Entry conditions	The Policy Maker has logged in

Flow of events	(a) The Policy Maker wants to report an evaluation (b) The Policy Maker clicks on the map button (c) The system redirects the Policy Maker to the map page (d) The Policy Maker visualizes the map and selects a farm (e) The system retrieves the information about the farm (also the last evaluation) and shows them to the Policy Maker (f) The Policy Maker clicks update button (g) The Policy Maker select the evaluation • The Policy Maker selects "good", decides an amount of money for the award and writes the body of the message • The Policy Maker selects "bad" and writes the body of the message (h) The Policy Maker clicks submit button and sends the message (i) The system saves the evaluation on the map (j) The system sends the message to the Farm's owner (k) The system notifies the Policy Maker that the operation went succesfully
Exit conditions	The Policy Maker visualizes the map updated
Exceptions	•

Table 14: *Map updating* use case description

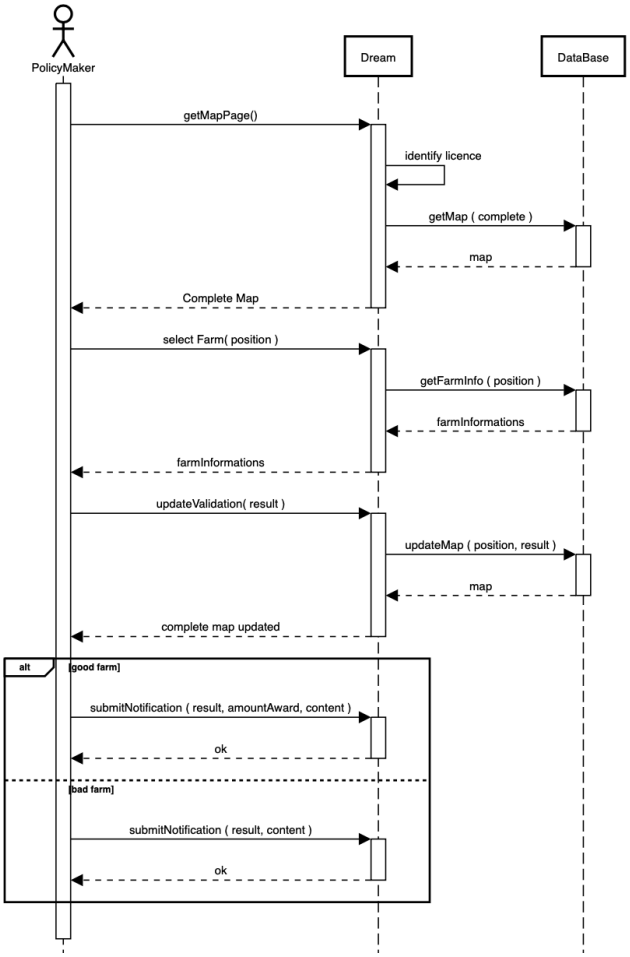

Figure 38: *Map updating* sequence diagram

14. **Reply to a request of help**

Name	**Policy Maker send anvice**
Actors	Policy Maker
Entry conditions	The Policy Maker has logged in

Flow of events

(a) The Policy Maker wants to send specific advices to a farmer

(b) The Policy Maker wants to see the information-about a farm

(c) The Policy Maker types the name of the farm on search form

(d) The Policy Maker clicks search button

(e) The Policy Maker wants to see the advices stored in the database on the type of production the farmers asked for help on

(f) The Policy Maker analyzes the data

(g) The Policy Maker selects the farmer (name) and the type of production and writes the solution he found

(h) The Policy Maker clicks send button

(i) The system sends the message to the farmer

(j) The system notifies the Policy Maker that the operation was succesful

Exit conditions	The Policy Maker send solutions
Exceptions	•

Table 15: *Problem resolution* use case description

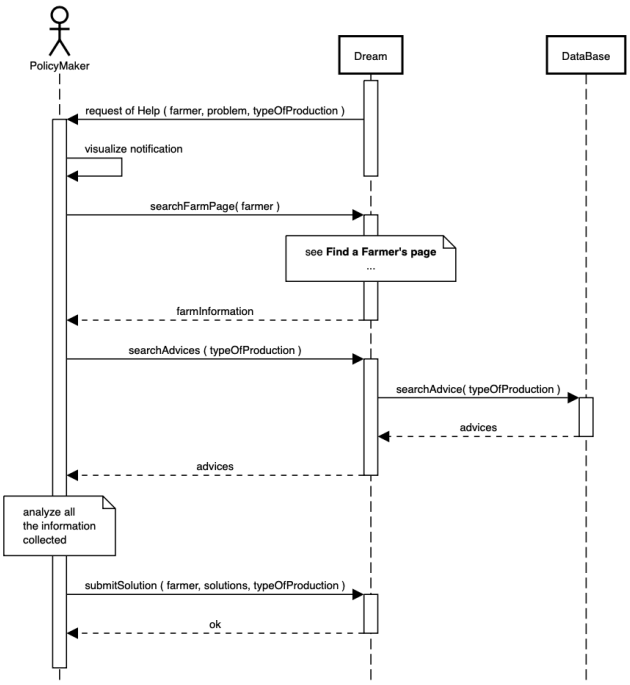

Figure 39: *Problem resolution* sequence diagram

3.1.3 Scenarios

1. **Scenario 1**
   Dayanita is a young woman that lives in Singaram, a small town in the state of Telengana in India. She lives with her parents in their family's farm.
   Their farm was mostly used to cultivate rice, so they are not really expert in other types of crop. She decides to try to expand their business, starting to grow soyabean but her parents do not really know how to help her with it.
   Dayanita discovers, thanks to an ad she found online, a new web application that has been recently released.
   Therefore she decides to sign up and write a message on the forum hoping that some other farmer can share with her some advices.
   Ajar, the owner of Ajar's farm, answered her message saying to be careful and check the weather condition before planting the seeds and some suggestions about the appropriate depth for planting them. Dayainita some days later checks the forum again and reads Ajar's message. The following days she checks the weather forecast that is present on her personalized farm page and actually finds the right period to start planting soyabean.

2. **Scenario 2**
   Nanuk is a Telengana's policy maker. Dream IT providers distributed to each policy maker reported by Telengana's government a secret code. Through this code and password Nanuk is able to log in to Dream web application. Before the 10th of the month all the policy makers are suppose to update the past evaluation of the farms already present in the system and add the farms that are added in the system in the past month. Using Dream the Nanuk searches on the map the farms that already been evaluated last month. He finds out that Shaila's Farm did not perform that well last time it was evaluated. Thus he decides to see if her performance is improved in the last month, he check the quantity of wheat producted and saw it increased dramatically from the month before. Therefore he decides to classify the farm as GOOD with a small incentive so that the owner of the farm is stimulated to keep on doing good.

3. **Scenario 3**
   Yamir is struggling with his Sesame crop. He tried everything, but still can not grow a quantity that could increase his profit.
   His friend Ishaan only grows corn so he said he could not help him with it. However he suggested him to try to sign up to Dream saying that there are a lot of ways he could get support from there.
   After he signed up he tried to write a message on the forum but nobody answered his text. So he sent a request of help to the policy makers and finally discovered that he need to give more water to his plants.

4. **Scenario 4**
   A policy maker working for Telengana's government is working with Dream application. The policy maker logs in with his code and password to check his notification.
   The first notification he notices is a new request of help from Ajar's farm. He then looks at Ajar's farm page to look up all the information.
   Ajar is looking for suggestions on his wheat crop. The policy maker at this point send the solution of Ajar's problem, filtering for type of production: wheat. He reads all the advices written by other farmers and chooses the one written by Denali: wheat should be planted in autumn or spring not in the other seasons. The policy maker writes then the solution and sends it to the farmer Ajar.

3.1.4 Requirements

R1 The system must allow farmers to register
R2 The system must allow farmers to log in
R3 The system must save the farmers registration data
R4 The system must guarantee that each email address is unique
R5 The system must verify that the email address is valid (the type!)
R6 The system must save the farmers information about their production submitted

R7 The system must allow farmers to insert the type of production
R8 The system must allow farmers to insert the amount of production type
R9 The system must allow farmers to specify a problem they faced to the Policy Makers
R10 The system must allow farmers to select the type of production on which they had troubles
R11 The system must save the advice submitted by the farmers
R12 The system must allow farmers to select the type of product in their suggestion
R13 The system must be able to show to the farmers advices send by the Policy Makers (as a notification)
R14 The system must be able to show the meteorological data of the Farm's position
R15 The system must be able to show the farm's sensor data
R16 The system must allow farmers to send messages on the forum
R17 The system must register date and time of a message in the forum
R18 The system must be able to show all the messages on the forum
R19 The system must be able to show the map of the zone
R20 The system must be able to show the farms position on the map
R21 the system must be able to show on the map if a farm is performing well or not
R22 The system must allow farmers to visualise notification send by Policy Makers
R23 The system must allow farmers to see all information about his own farm
R24 The system must be able to show the types of production cultivated in a farm
R25 The system must be able to show the quantity of product has been cultivated for each type in a farm
R26 The system does not allow Policy Makers to register
R27 The system must allow Policy Makers to log in
R28 The system must verify that the code is valid
R29 The system must allow Policy Makers to search a farm by name
R30 The system must allow Policy Makers to see all farms' pages
R31 The system must not allow Policy Makers to modify any farm page
R32 The system must allow Policy Makers to update the performance of a farmer
R33 The system must allow Policy Makers to send notifications to the farmers
R34 The system must allow Policy Makers to receive requests of help by the farmers

3.1.5 Goals

Goals	Domain Assumpitons	Requirements
G1	DA1, DA2, DA3, DA4, DA5, DA6, DA7, DA8, DA9, DA13, DA16, DA18	R14, R15, R26, R27, R28, R29, R30, R31, R32
G2	DA1, DA4, DA6, DA10, DA15, DA17	R1, R2, R3, R4, R5, R16, R17, R18
G3	DA1, DA2, DA6, DA9, DA12, DA14, DA15, DA17	R1, R2, R3, R4, R5, R6, R7, R8, R11, R12
G4	DA1, DA6, DA9, DA11, DA15, DA17	R1, R2, R3, R4, R5, R9, R10, R13, R22, R33, R34
G5	DA1, DA3, DA4, DA5, DA6, DA7, DA8, DA9, DA10, DA13, DA15, DA17	R1, R2, R3, R4, R5, R14, R15, R22, R23, R24, R25
G6A	DA1, DA5, DA6, DA8, DA9, DA15, DA16, DA17	R1, R2, R3, R4, R5, R19, R20, R26, R27, R28
G6B	DA1, DA2, DA5, DA6, DA8, DA9, DA13, DA15, DA16, DA17	R19, R20, R21, R24, R25, R26, R27, R28, R32, R33
G6C	DA1, DA2, DA5, DA6, DA8, DA9, DA15, DA16, DA17	R1, R2, R3, R4, R5, R19, R20, R24

Table 16: Mapping of goals, domain assumptions and requirements

- **G1** Allow Policy Makers to retrieve information about a farm and to evaluate their performance
  - **R14** The system must be able to show the meteorological data of the Farm's position
  - **R15** The system must be able to show the farm's sensor data
  - **R26** The system does not allow Policy Makers to register
  - **R27** The system must allow Policy Makers to log in
  - **R28** The system must verify that the code is valid
  - **R29** The system must allow Policy Makers to search a farm by name
  - **R30** The system must allow Policy Makers to see all farms' pages

  - **R31** The system must not allow Policy Makers to modify any farm's page
  - **R32** The system must allow Policy Makers to update the performance of a farmer
  - **DA1** In order to access the system users need to have Internet connection
  - **DA2** Farmers always insert correct data on their production activity
  - **DA3** Data from sensors is always correct
  - **DA4** Date and Time on the system are always correct
  - **DA5** The position of the Farm is always correct
  - **DA6** Internet connection works always without errors
  - **DA7** Meteorological data is accurate
  - **DA8** Every farm has a different position
  - **DA9** Each farm belongs to exacly one farmer
  - **DA13** Performances of farmers are always identified correctly
  - **DA16** Policy Makers have a given code and password to access the system
  - **DA18** The special incentive that the farmers receive for their good work, is used on stuff related to their farm activity

- **G2** Allow farmer to comunicate with each other
  - **R1** The system must allow farmers to register
  - **R2** The system must allow farmers to log in
  - **R3** The system must save the farmers registration data
  - **R4** The system must guarantee that each email address is unique
  - **R5** The system must verify that the email address is valid
  - **R16** The system must allow farmers to send messages on the forum
  - **R17** The system must register date and time of a message in the forum
  - **R18** The system must be able to show all messages on the forum
  - **DA1** In order to access the system users need to have Internet connection
  - **DA4** Date and Time on the system are always correct
  - **DA6** Internet connection works always without errors
  - **DA10** Discussion on the forum are related only to the farm activity
  - **DA15** The email and farm's name must be unique
  - **DA17** All farmers that sign up own a real farm in Telengana

- **G3** Allow farmer to insert data and advices on his production
  - **R1** The system must allow farmers to register
  - **R2** The system must allow farmers to log in
  - **R3** The system must save the farmers registration data
  - **R4** The system must guarantee that each email address is unique
  - **R5** The system must verify that the email address is valid
  - **R6** The system must save the farmers information about their production submitted
  - **R7** The system must allow farmers to insert the type of production
  - **R8** The system must allow farmers to insert the amount of production type
  - **R11** The system must save the advice submitted by the farmers
  - **R12** The system must allow farmers to select the type of product in their suggestion
  - **DA1** In order to access the system users need to have Internet connection
  - **DA2** Farmers always insert correct data on their production activity
  - **DA6** Internet connection works always without errors
  - **DA9** Each farm belongs to exacly one farmer
  - **DA12** Advice on a product must be given by a farmer that produces the same type
  - **DA14** Farmers can insert data more than once a day
  - **DA15** The email and farm's name must be unique
  - **DA17** All farmers that sign up own a real farm in Telengana
- **G4** Allow farmer to send request of help to Policy Makers
  - **R1** The system must allow farmers to register
  - **R2** The system must allow farmers to log in
  - **R3** The system must save the farmers registration data
  - **R4** The system must guarantee that each email address is unique
  - **R5** The system must verify that the email address is valid
  - **R9** The system must allow farmers to specify a problem they faced to the Policy Makers
  - **R10** The system must allow farmers to select the type of production on which they had troubles
  - **R13** The system must be able to show to the farmers advices send by the Policy Makers

  - **R22** The system must allow farmers to visualise notification send by Policy Makers
  - **R33** The system must allow Policy Makers to send notification to the farmers
  - **R34** The system must allow Policy Makers to receive request of help by the farmers
  - **DA1** In order to access the system users need to have Internet connection
  - **DA6** Internet connection works always without errors
  - **DA9** Each farm belongs to exacly one farmer
  - **DA11** Formal request of help must be related to a farmer's own production
  - **DA15** The email and farm's name must be unique
  - **DA17** All farmers that sign up own a real farm in Telengana
- **G5** Allow farmers to retrieve information relevant for their activity
  - **R1** The system must allow farmers to register
  - **R2** The system must allow farmers to log in
  - **R3** The system must save the farmers registration data
  - **R4** The system must guarantee that each email address is unique
  - **R5** The system must verify that the email address is valid
  - **R14** The system must be able to show the meteorological data of the Farm's position
  - **R15** The system must be able to show the farm's sensor data
  - **R22** The system must allow farmers to visualise notification send by Policy Makers
  - **R23** The system must allow farmers to see all information about his own farm
  - **R24** The system must be able to show the types of production cultivated in a farm
  - **R25** The system must be able to show the quantity of product has been cultivated for each type in a farm
  - **DA1** In order to access the system users need to have Internet connection
  - **DA3** Data from sensors is always correct
  - **DA4** Date and Time on the system are always correct
  - **DA5** The position of the Farm is always correct
  - **DA6** Internet connection works always without errors

  - **DA7** Meteorological data is accurate
  - **DA8** Every farm has a different position
  - **DA9** Each farm belongs to exacly one farmer
  - **DA10** Discussion on the forum are related only to the farm activity
  - **DA13** Performances of farmers are always identified correctly
  - **DA15** The email and farm's name must be unique
  - **DA17** All farmers that sign up own a real farm in Telengana

- **G6** Policy Makers and farmers should be able to consult the map of the zone (and the informations stored in it) with different levels of visibility

  1. **G6A** Policy Makers and farmers should be able to see the position of the farms on the map
     - **R1** The system must allow farmers to register
     - **R2** The system must allow farmers to log in
     - **R3** The system must save the farmers registration data
     - **R4** The system must guarantee that each email address is unique
     - **R5** The system must verify that the email address is valid
     - **R19** The system must be able to show the map of the zone
     - **R20** The system must be able to show the farms position on the map
     - **R26** The system does not allow Policy Makers to register
     - **R27** The system must allow Policy Makers to log in
     - **R28** The system must verify that the code is valid
     - **DA1** In order to access the system users need to have Internet connection
     - **DA5** The position of the Farm is always correct
     - **DA6** Internet connection works always without errors
     - **DA8** Every farm has a different position
     - **DA9** Each farm belongs to exacly one farmer
     - **DA15** The email and farm's name must be unique
     - **DA16** Policy Makers have a given code and password to access the system
     - **DA17** All farmers that sign up own a real farm in Telengana
  2. **G6B** Policy Makers should be able to see both the information about the production and the evaluation of each farm
     - **R19** The system must be able to show the map of the zone
     - **R20** The system must be able to show the farms position on the map
     - **R21** The system must be able to show on the map if a farm is performing well or not

- **R24** The system must be able to show the types of production cultivated in a farm
- **R25** The system must be able to show the quantity of product has been cultivated for each type in a farm
- **R26** The system does not allow Policy Makers to register
- **R27** The system must allow Policy Makers to log in
- **R28** The system must verify that the code is valid
- **R32** The system must allow Policy Makers to update the performance of a farmer
- **R33** The system must allow Policy Makers to send notification to the farmers
- **DA1** In order to access the system users need to have Internet connection
- **DA2** Farmers always insert correct data on their production activity
- **DA5** The position of the Farm is always correct
- **DA6** Internet connection works always without errors
- **DA8** Every farm has a different position
- **DA9** Each farm belongs to exacly one farmer
- **DA13** Performances of farmers are always identified correctly
- **DA15** The email and farm's name must be unique
- **DA16** Policy Makers have a given code and password to access the system
- **DA17** All farmers that sign up own a real farm in Telengana

3. **G6C** Farmers should be able to see only the type of production of a farmer by the map
   - **R1** The system must allow farmers to register
   - **R2** The system must allow farmers to log in
   - **R3** The system must save the farmers registration data
   - **R4** The system must guarantee that each email address is unique
   - **R5** The system must verify that the email address is valid
   - **R19** The system must be able to show the map of the zone
   - **R20** The system must be able to show the farms position on the map
   - **R24** The system must be able to show the types of production cultivated in a farm
   - **DA1** In order to access the system users need to have Internet connection
   - **DA2** Farmers always insert correct data on their production activity
   - **DA5** The position of the Farm is always correct

- **DA6** Internet connection works always without errors
- **DA8** Every farm has a different position
- **DA9** Each farm belongs to exacly one farmer
- **DA15** The email and farm's name must be unique
- **DA16** Policy Makers have a given code and password to access the system
- **DA17** All farmers that sign up own a real farm in Telengana

3.1.6 Traceability Matrix

Requirements	Use Cases	Scenarios
R1	UC1, UC3, UC4, UC5, UC6, UC7, UC9	S1, S3
R2	UC2, UC3, UC4, UC5, UC6, UC7, UC8, UC9	S1, S3
R3	UC1, UC6	S1, S3
R4	UC1, UC2	
R5	UC2	
R6	UC2, UC3	S1, S3
R7	UC4	S1, S3
R8	UC4	S1, S3
R9	UC7	S3
R10	UC7	S3
R11	UC8	
R12	UC8	
R13	UC9, UC14	S3
R14	UC6	S1
R15	UC6	S1
R16	UC3	S1, S3
R17	UC3	S1, S3

Table 17: Traceability matrix from R1 to R17

Requirements	Use Cases	Scenarios
R18	UC3	S1, S3
R19	UC5, UC12	S2
R20	UC5, UC12	S2
R21	UC12	S2
R22	UC9	S3
R23	UC6, UC9	S1, S3
R24	UC5, UC6, UC11	S1, S3
R25	UC6, UC11	S1, S2, S3, S4
R26		S2
R27	UC10	S2, S4
R28	UC10	S2, S4
R29	UC11	S2, S4
R30	UC11	S2, S4
R31		S2, S4
R32	UC13	S2
R33	UC13, UC14	S2, S4
R34	UC14	S4

Table 18: Traceability matrix from R18 to R34

3.2 Performance Requirements

The system serves its user with a web application. All the computations will take place on the server side, thus the app is meant to be lightweighted. Moreover the load in the night is expected to be really low. There are no problems about reliability. The insertion of new data requires a quick response in order to store it in the system.

3.3 Design Constraints

3.3.1 Standards Compliance

The only standard that needs to be highlighted here is the interaction with the database. It is important that the information are stored in a standardized form. In such manner, it is easier to memorize and retrieve data.

3.3.2 Any Other Constraints

Interaction between Dream and users needs to consider also regulatory policies. As a matter of fact the application asks and retrieves data of each farmer. More information about security and privacy will be provided in the section 3.4.3

3.4 Software System Attributes

3.4.1 Reliability

The system must prevent any failure in order to guarantee continuity. Simultaneous accesses are expected to work, especially in the afternoon when users are expectd to insrt and retrieve more information.

3.4.2 Availability

It is expected that the system has the lowest downtime possible. The system is available with a minimum time of 96%, so that it about 14 days a year of downtime are allowed.

3.4.3 Security

Security of the data and of the communication user-system is a primary concern. Users credential are stored in a data base, so the system crypts the password data before storing it.
The system recognises the right type of user during the log in phase to ensure the correct level of visibility of the data and the permission to update the data or not. In this way farmer's privacy is guaranteed. As a matter of fact a farmer can not see another farmer page.

3.4.4 Maintanability

The web application requires ordinary maintenance for improvements and in order to fix potential bugs. It is going to be scheduled at local night time, when user traffic is the lowest. Moreover the system must be designed in a way that allows future addition of features.

3.4.5 Portability

The system as a web application must run on different software system as Windows, Linux an macOS. On the server side is crucial focusing on the interaction between APIs and the data base to insert, update or read data.

4 Formal Analysis using Alloy

This section provides a formal model of the problem described with Alloy modeling language. The model considers the most important constraints. In addition to the model there are some possible worlds meant to clarify some critical aspects. Simplifications:

4.1 Alloy model

```
1
2  sig Username{}
3  sig Password{}
4  {
5   all p: Password | ( some u: User |  u.password  = p)
6  }
7  sig Surname{}
8  sig Email{}
9  sig Code{}
10
11 abstract sig User{
12   password: one Password
13 }
14
15 sig Farmer extends User{
16     username: one Username,
17     surname: one Surname,
18     email: one Email
19 }
20
21 sig PolicyMaker extends User{
22     code: one Code
23 }
24
25 sig Date{
26     day: one Int,
27     month: one Int,
28     year: one Int
29 }{
30     day > 0
31     month > 0 || month< 13
32     year > 0
33 }
34
35 sig Time{
36     hour : one Int,
37     minute : one Int
38 }{
39     hour > 0
40     minute >= 0
```

```
}

sig MessageContent{}

sig Message{
    date: one Date,
    time: one Time,
    content: one MessageContent,
    sender: one Farmer
}

one sig Forum{
    messages: set Message
}

sig ProductName{}

sig Production{
    type: one ProductName,
    amount: one Int,
    date : one Date
}{
  amount > 0
}

sig SensorData{
    quantity: Int,
    date: Date
}{
  quantity >= 0
}

abstract sig Bool{}
one sig True extends Bool{}
one sig False extends Bool{}

abstract sig Notification{
    body: one MessageContent,
    date: one Date,
    time: one Time
}

sig Help extends Notification{
    sender: one Farmer,
    receiver: one PolicyMaker,
    typeOfProduction: one Production
}
sig Advice extends Notification{
    sender: one Farmer,
    receiver: one PolicyMaker,
```

```
    typeOfProduction: one Production
}

sig Solution extends Notification{
    sender: one PolicyMaker,
    receiver: one Farmer,
    typeOfProduction: one Production
}

sig Evaluation extends Notification{
    sender: one PolicyMaker,
    receiver: one Farmer,
    result: one Bool
}

sig Position {}
sig Weather{}
sig WeatherInfo{
    basicInfo: Weather,
    temperature: Int,
    position: one Position,
    date: Date
}

sig FarmName{}
sig Farm{
    name: one FarmName,
    owner: one Farmer,
    products: set Production,
    water: some SensorData,
    humidity: some SensorData,
    position: one Position,
    weather: some WeatherInfo,
    evaluation: set Evaluation
}

one sig Map{
    farms: some Farm
}


////////////////////////////////////////
//   FACTS
////////////////////////////////////////

//Bool can only be true or false
fact boolean{
    True + False = Bool
}

```

```
//Uniquness of farmer's username and email
fact UniqueUsernames{
    no disj f1,f2: Farmer| f1.username = f2.username
}

fact UniqueEmail{
    no disj f1,f2: Farmer| f1.email = f2.email
}

//username and surname can't exist without farmers
fact noUsernameWithoutFarmer {
all un: Username | one f: Farmer | f.username = un
}

fact noSurnameWithoutFarmer {
all sur: Surname | one f: Farmer | f.surname = sur
}

//policy maker's code is unique
fact UniqueCode{
    no disj pm1,pm2: PolicyMaker| pm1.code = pm2.code
}

//farm name is unique
fact UniqueFarmName {
    no disj f1, f2: Farm | f1.name = f2.name
}

//no farmname without farm
fact noFarmNameWithoutFarm {
    all fn: FarmName | one f: Farm | f.name = fn
}

//there are not two farms in the same position
fact UniqueFarmsPosition{
    no disj f1,  f2: Farm | f1.position = f2.position
}

//no farmer without farm
fact noFarmerWithoutFarm {
    all f: Farmer | one farm: Farm | farm.owner = f
}

//There is no weather info with the same date and position
    with different temperature or weather info
fact singleWeatherInfo{
    all w1, w2: WeatherInfo | (w1.position=w2.position and
    w1.date=w2.date)  implies (w1.temperature= w2.temperature
     and w1.basicInfo=w2.basicInfo)
}
```

```
188
189 //weather position is the same one of the farm
190 fact weatherValidity{
191   all f: Farm | f.position = f.weather.position
192 }
193
194 //production does not exist without farm
195 fact noProductionWithoutFarm{
196     all p: Production | one f: Farm | p in f.products
197 }
198
199 //product does not exist without production
200 fact noProductWithoutProduction{
201   all p: ProductName | one production: Production | p =
      production.type
202 }
203
204 //no different production's type in one farm each day
205 fact noProductionsOfSameType{
206     no disj p1, p2: Production | one f: Farm  | p1.type=p2.
    type and p1 in f.products and p2 in f.products and p1.
    date=p2.date
207 }
208
209 //all messages are stored in the forum
210 fact messageValidity{
211   all m:Message | one f:Forum | m in f.messages
212 }
213
214 //all messages have different content
215 fact noEqualMessages{
216   no disj m1, m2: Message | m1.content = m2.content
217 }
218
219 //all notifications have different content
220 fact noEqualNotifications{
221   no disj n1, n2: Notification | n1.body = n2.body
222 }
223
224 //no messages sent at the same moment by the same farmer
225 fact oneMessageAtTime{
226     no disj m1, m2: Message | m1.sender=m2.sender and m1.
    date=m2.date and m1.time=m2.time
227 }
228
229 //every help request has exacly one Solution
230 fact oneSolutionForOneHelp{
231     all h: Help | one s: Solution | h.sender = s.receiver
    and h.typeOfProduction=s.typeOfProduction
232 }
```

```
233
234 //no sensor data exists without a farm
235 fact noSensorWithoutFarm{
236     all s1, s2 : SensorData | one f: Farm | s1 in f.water
    and s2 in f.humidity
237 }
238
239 //all the farms are in the map
240 fact allFarmsMapped {
241     all f: Farm | one m: Map | f in m.farms
242 }
243
244 //only one evaluation per month for a farm
245 fact singleEvaluation {
246     no disj e1, e2: Evaluation | e1.receiver=e2.receiver and
     e1.date.month=e2.date.month and e1.date.year = e2.date.
    year
247 }
248
249 //the evaluation receiver must be the same of the owner of
    the farm on which is made
250 fact evaluationValidity{
251   all f: Farm | #f.evaluation >0 implies f.owner = f.
    evaluation.receiver
252 }
253
254 //no evaluations exists without the associated farm
255 fact noEvaluationWithoutFarm{
256   all e: Evaluation | one f: Farm | f.owner=e.receiver
    implies e in f.evaluation
257 }
258
259 //only farmers who have a positive evaluation can write
    advices
260 fact howCanSubmitAnAdvice {
261     all advice: Advice | one farm: Farm | one e: Evaluation
    | advice.sender=farm.owner and e in farm.evaluation and
    advice.date.month=e.date.month and advice.date.year=e.
    date.year and e.result = True
262 }
263
264 ////////////////////////////////////////
265 //   ASSERTIONS
266 ////////////////////////////////////////
267
268 // G1: allow policy makers to retrieve information from
    farmers and to evaluate their performance
269 assert evaluatePerformormance{
270   no e:Evaluation | one f: Farm |  e.receiver = f.owner
    implies e not in f.evaluation
```

```
}
check evaluatePerformormance for 10

// G2: allow farmers to communicate with each other
assert farmersCommunication{
  all m: Message | one f: Forum | m in f.messages implies (
    one farmer: Farmer | m.sender = farmer)
}
check farmersCommunication for 10

// G3: allow farmers to insert data and advices on his
    production
assert insertData{
  all p: Production | some f: Farm | p in f.products implies
     (one product:ProductName |
product = p.type) and (#f.products >0)
}
check insertData for 10

assert insertAdvice{
  no a: Advice | one f: Farm | a.sender = f.owner and #f.
    evaluation<1
}
check insertAdvice for 10


//////////////////////////////////////
//   PREDICATES
//////////////////////////////////////


pred world1{
  #PolicyMaker = 0
  #Notification=0
  #Farmer = 2
  #Code = 0
  #MessageContent = 0
}
run world1 for 3

pred world2{
  #Farmer = 1
  #Evaluation = 3
  #Notification = 3
  #Message = 0
  #PolicyMaker = 2
}
run world2 for 6

pred world3{
```

```
317   #Farmer = 2
318   #Message = 4
319   #PolicyMaker = 0
320   #Code = 0
321   #Production = 2
322   #Date = 2
323   #Time = 3
324 }
325 run world3 for 5
```

4.2 First World

In the first world (Figure 40) the focus is on the *registration* of two farmers. Since we assume that they use the application for the first time, the only information present are the ones required for the registration. Email and username are unique. On the contrary password and surname can be the same for different farmers. Another thing to notice is that email, username, surname and password do not exist without an associated user. Given that each farmer is associated to a farm, creating a world with two farmers, automatically considers the same amount of different farm as well. The number of policy makers is zero to highlight the fact that farmers do not depend directly with policy makers.

4.3 Second World

The second world showed in Figure 41 focuses on a single farm with *multiple evaluations*. Therefore it is considered a farm, with its owner and all their characteristics. Every evaluation made on the farm has a different date, more precisely it does not have the same month, and a different messageContent. They are written by different policy makers but the most important thing to notice is that the receiver of the evaluation is the same owner of the farm that has the evaluation.

4.4 Third World

The third world (Figure 42) highlights the possibility of the farmers to communicate to each other through the forum. The farmer can send messages and fill in a report with a production he made in his farm in a specific date. Each message sent in the forum has exactly one sender (a farmer only). The system also verifies that the sender can write one message at a time. The forum is unique but it can be empty (look at world 1) if nobody has sent any message yet. On the other side each production submitted in a day must be of different types. As a matter of fact the form has the field 'quantity' used to specify the amount so it is not necessary to generate a production for each harvest.

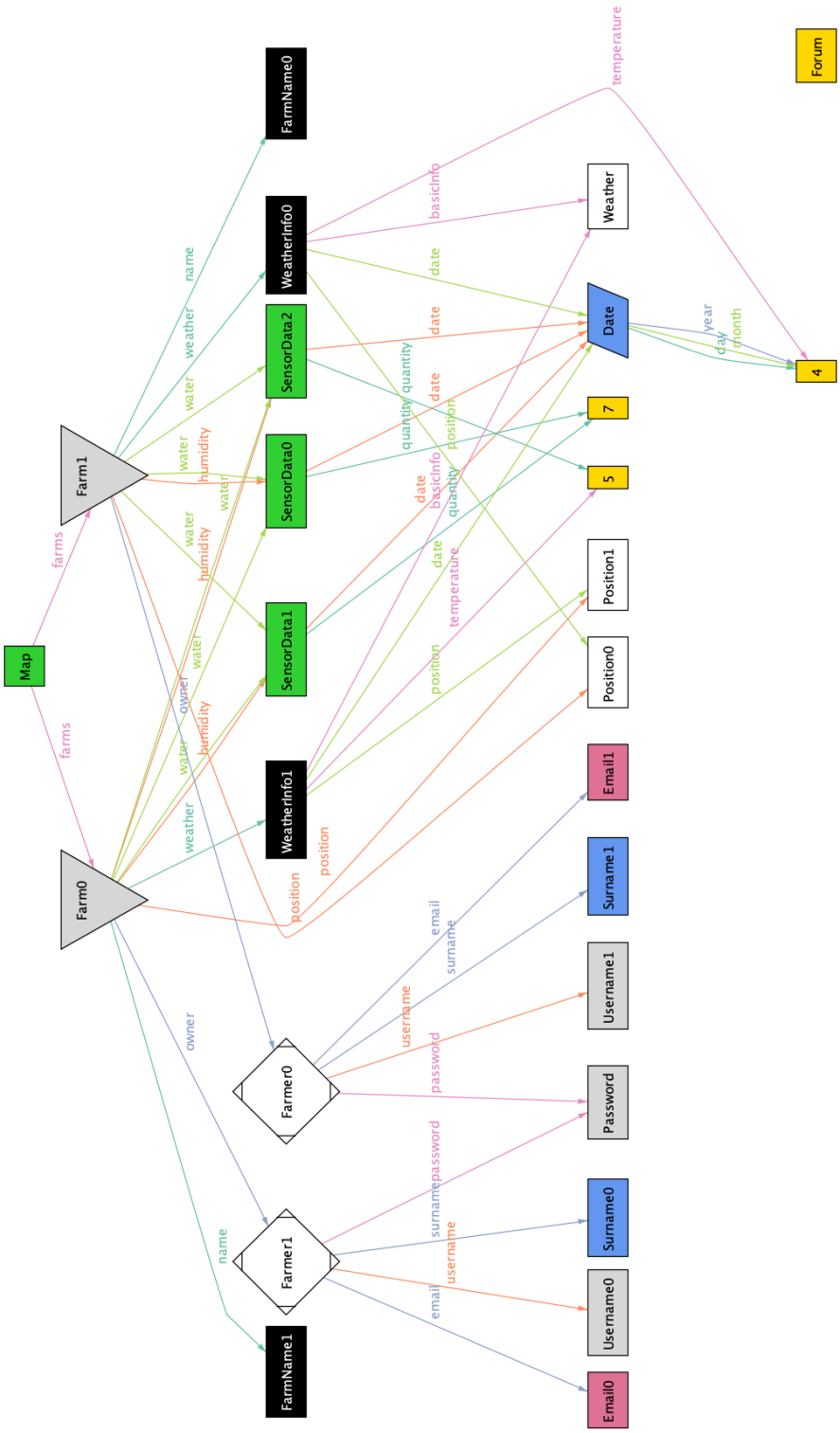

Figure 40: FirstWorld

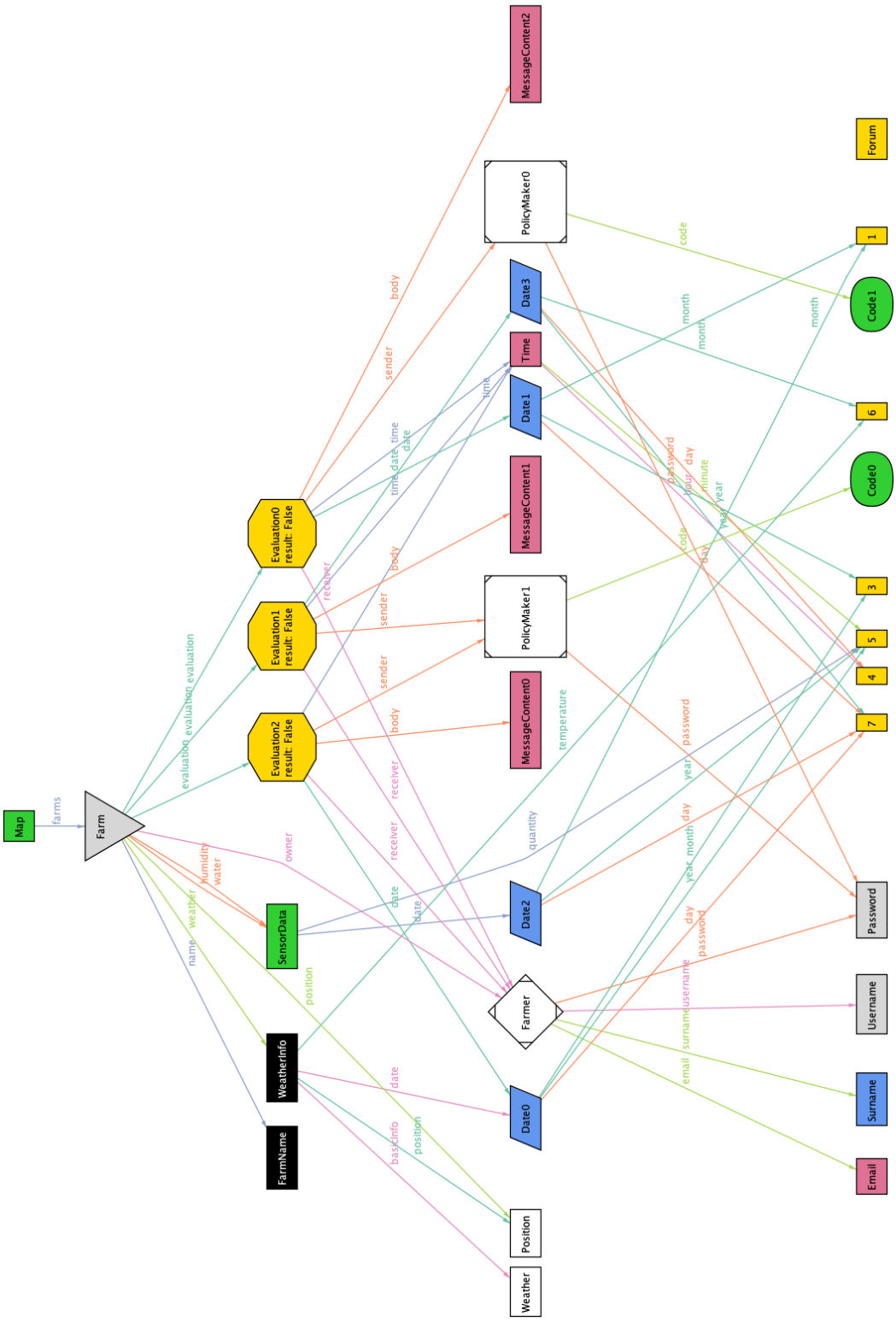

Figure 41: Second World

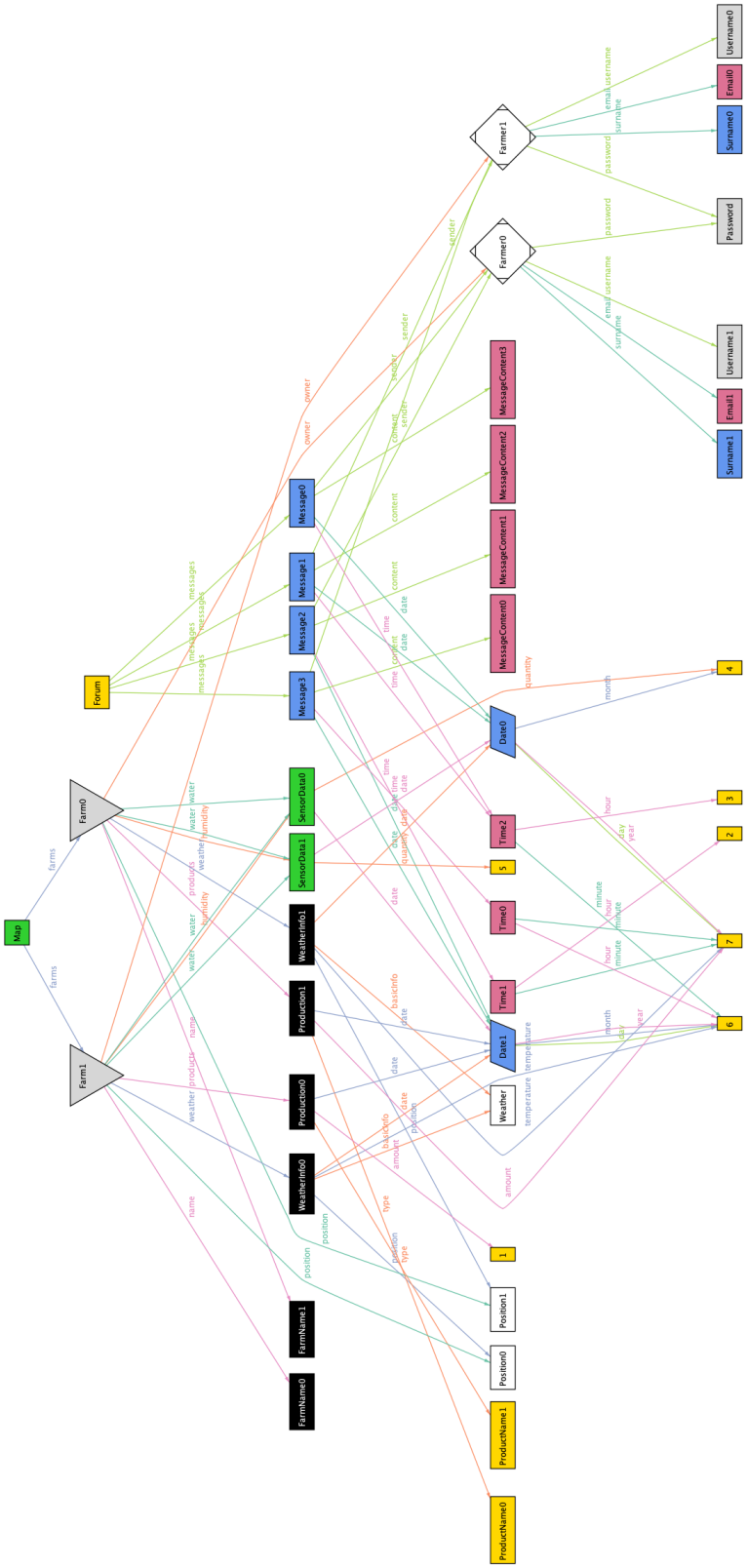

Figure 42: Third World

5 Effort Spent

Student	Time for S.1	Time for S2	Time for S.3	Time for S.4
Valeria Detomas	5h	6h	15h	8h
Sofia Martellozzo	5h	5h	17h	8h

6 Software and Tools used

- LaTeX as document preparation system
- Lucid for the state chart
- SequenceDiagram.org for the sequence diagrams
- Umletino for UML diagram
- Diagrams for the use case diagram
- Balsamig for the mockups
- **Alloy** as a model analyzer

7 References

- Specification document: R&DD Assignment A.Y. 2021-2022