Travlendar+ project DILETTA QUARTICELLI ALESSANDRO PINDOZZI

Requirement Analysis and Specification Document

Deliverable:	RASD
Title:	Requirement Analysis and Verification Document
Authors:	DILETTA QUARTICELLI ALESSANDRO PINDOZZI
Version:	2.0
Date:	6-February-2022
Download page:	https://github.com/apindozziPolimi/PindozziQuarticelli
Copyright:	Copyright © 2021, DILETTA QUARTICELLI ALESSANDRO PINDOZZI – All rights reserved

Contents

List of Figures

1 Introduction

1.1 Purpose

Agriculture is the basic source of India's economic, in fact 58% of rural households depend on it. In past decade, it is observed that there is not much crop development in agriculture sector consequently food price are increasing. Moreover the climate change defeats more and more the agriculture sector which impacts productivity. The climate change is not the only problem, the population will increase in 2050 in order to the food demand will increase.

Among these problems, also the COVID-19 crisis had a negative impact in this sector, hurting farms of all sizes in India. It has exposed the vulnerability of India's food system and accentuated the need for agricultural market reforms and digital solutions to connect farmers to markets, to create safety nets and ensure reasonable working conditions (as is written in the article How Indian agriculture should change after COVID-19).

Telangana is the 11th largest state in India with a geographical area of 112,077 km2 and 35,193,978 residents (data from 2011) and is one of the fastest-growing states in India posing average annual growth rate of 13.90% over the last five years . Agriculture form a backbone of Telangana's Economy.

In the light of the above-mentioned problems, the farmers must cultivate more, to deal with the problem of the population increase, and better to address the problem of the climate changes and the crisis due to the COVID-19 pandemic. For this reason Telangana's government wants to find a way to design develop and demonstrate anticipatory governance models for food systems using digital public goods and community-centric approaches to strengthen data-driven policy making in the state.

Figure 1: Logo of DREAM

In order to do this, the system "DREAM" will support the work of different stakeholders from farmers to policy makers. Farmers can use it to insert information about their productions, to receive suggestions, to check weather conditions everyday because DREAM interacts with Weather.com, in order to take better decisions. They also can ask for help to the agronomists, or chat with other farmers in the forums they create to share their experiences and help each other. The policy Maker, instead can check the work of every farmer, seeing the productions' information they insert in the system. Therefore DREAM is able to compute a ranking of the farmer for the production of a certain product, based on the amount of product collected, the size of the field, the age of the farmer and the weather conditions that the farmer faced during the period of this production. In this way the policy makers can identify the farmers who are performing well, and who are more resilient to adverse weather conditions, in order to send them incentives and encourage them ton give advice to the others. In the same way they can identify the farmer who are most in difficulty, and contact the agronomists who can help them. So DREAM give them all the cards they need in order to improve the food system and to take future decisions.

This paper has the purpose to describe the application "DREAM" trough the description of the scenarios, uses cases, and the models describing the requirements and specification. The documentation is created in order to accomplish the request of the stakeholders.

In the following pages there are precise descriptions of the functional and non-functional requirements, the different scenarios and uses cases to describe the system and how it could be implemented. In the second part of the system there are a more formula definition of the requirements with the use of the

Alloy. This documentation is useful for the developers involved in the creation of the system and for the stakeholders to understand the system.

1.1.1 Goals

In the previous part are described the main goals of DREAM, which are shown in the following table for clarity.

Goals	Description
G1	Allow farmers to visualize suggestions concerning specific product to cultivate based on the position of their field
G2	Allow farmers to visualize suggestion concerning specific fertilizer to use based on the position of their filed
G3	Allow farmers to see weather forecasts based on the location of their fields
G4	Allow farmers to interact each others in a forum
G5	Allow farmers to request for help to the agronomists
G6	Allow Policy makers to identify farmers who are performing well
G7	Allow Policy makers to identify farmers who are performing bad
G8	Allow Policy makers to check the work of the agronomists who help the farmers
G9	Allow Policy makers to see information about Telagana's farmers productions

1.2 Scope

DREAM is offered to all the farmers and the policy maker of Telangana, in order to create a continuous collaboration between these parties.

A farmer, after registering and logging in, will be able to insert his/her fields' information, like the location of the field, the size and information about what he/she produces, such as the type of products, the amount, the fertilizers used. Anyway what is meant when referring to field's information and production information is better explained in the subsection 1.3. DREAM allow farmers to check the weather forecasts, in a certain location and for certain dates, so that the farmer can better manage his/her productions. The farmer may need help or suggestions. In the system a differentiation is made between help and suggestion. When the farmer asks for help, inserting data about the location of his/her field and the type of problem he/she faced, the system sends a message to the agronomists, interacting with their own platform. The agronomist will receive the request for help and will response by communicating a date in which they will physically go to the field indicated by the farmer in order to help him. The farmer will receive the message thanks to the system, which provides a section for the notifications. The suggestions, instead, are given directly by DREAM, which is able to provide suggestions about the best products and fertilizer to use in each location. So the farmer only has to enter the location of the field for which he needs suggestions and DREAM will respond immediately. The last important feature of the system is the forum. They can interact each other, either to ask for help or to give advice to other farmers. A farmer can create his/her own forum, or join a forum created by other farmers, in order to chat with them. Each forum has a topic, so that a farmer who join a forum already knows what is being talking about in it.

DREAM offers also functionality to Policy makers. A policy maker, already registered and logged into the system, can see the information of each farmer's fields and each farmer's production. Moreover, for each production DREAM provides data about the average quantity of water used by the farmer and the average soil humidity level (obtained through the interaction by API with the water irrigation system and the soil humidity sensors), as well as information on weather conditions (like the amount of rainfall) during that production. Another important feature offered to the policy maker is Ranking: the system computes a ranking for each farmer who has cultivated the product in order to allow policy makers to see who are the best and worst farmer. In this way the policy makers can decide to send incentives to the best farmers, and messages to the agronomist in order to help the worst ones. Pay attention that DREAM does not provide an online payment system, but allow policy makers to send to the farmers notifies that they will receive an incentive (for example via bank transfer). Therefore, in the notify sent to the farmer, he/she is encouraged to create a forum by which he can explain why and how he is performing so well, so that he can help other farmers. Also in this case, when a message is sent to the agronomists, it is indicated the farmer who need help, the location of the field and production, and the agronomist will receive the request by another platform, as explained previously for the case of the request for help sent by the farmers. Last but not least, policy makers can check the work of the agronomist who help the farmers. In fact, when they finish their intervention they write a report on their platform, which is sent to DREAM and provided to the policy makers. In this way they can see who are the agronomists and the farmers who have collaborated, what the intervention was about, and what improvements are visible at the end of the intervention.

1.2.1 World Phenomena

World Phenomena	Description
WP1	The farmer is affected by adverse meteorological events
WP2	The farmer cultivates the fields
WP3	The farmer collects the products
WP4	The farmer has a problem
WP5	The farmer needs help
WP6	The farmer needs suggestions
WP7	The farmer uses fertilizer
WP8	The farmer irrigates the field
WP9	The agronomist helps the farmer
W11	The water irrigation system collects data about amount of used water
W12	The soil's sensor system collects data about the humidity of the soil

1.2.2 Shared phenomena

Shared Phenomena	Description	Control
SP1	The farmer can register in the system	World controlled
SP2	The farmer can log in in the system if he is already registered in the system	World controlled
SP3	The farmer inserts his/her field's information	World controlled
SP4	The farmer inserts his/her production's information	World controlled
SP5	The farmer ask for suggestions about products	World controlled
SP6	The farmer ask for suggestions about fertilizers	World controlled
SP7	The system gives suggestions about fertilizers	Machine controlled
SP8	The system gives suggestions about products in a specific field	Machine controlled

SP9	The farmer asks for help to the agronomist	World controlled
SP10	The system sends farmer's request for help to the Agronomist Platform	Machine controlled
SP11	The farmer creates a forum	World controlled
SP12	The farmer joins a forum	World controlled
SP13	The farmer checks weather forecasts	World controlled
SP16	The farmer checks notifications	World controlled
SP17	The policy maker inserts the secret code to sign up in the system	World controlled
SP18	The system sends the login credentials to the policy maker	Machine Controlled
SP19	The policy maker logs in the system	World controlled
SP20	The policy maker sees the list of fields and their owner	World controlled
SP21	The policy maker sees production's information	World controlled
SP32	The system collects data concerning the quantity of water used from the water irrigation systems	Machine controlled
SP22	The system shows to the policy maker data concerning the amount of water used for a certain production	Machine controlled
SP33	The system collects data concerning the humidity of the soil from the humidity sensors	Machine controlled
SP23	The system shows to the policy maker data concerning the humidity of the soil related to a certain production	Machine controlled
SP24	The system compute a ranking of farmers based on their productions	Machine controlled
SP25	The policy maker sees the farmers' ranking for a certain product in descending order	World controlled
SP34	The policy maker sees the farmers' ranking for a certain product in growing order	World controlled
SP26	The policy sends the notification to the farmer that he/she will receive an incentive	World controlled
SP27	The policy maker sees the history of old incentives sent to farmers	World controlled
SP28	The system collects the agronomist's reports from the Agronomist platform	Machine controlled
SP31	The policy maker sees the agronomist's report	Machine controlled
SP29	The system shows the different score between before and after the agronomist's help to the policy maker	Machine controlled
SP30	The policy makers sends a message to the agronomist in order to help farmer who perform bad	World controlled

1.3 Definitions and abbreviations

Definition	Descriptions

Field's information	Information about field: • the size • the location • the details of the field
Farmer's information	Information of the farmer: • name • surname • birth date • Country • sex
Production	Product cultivated in a single field in a period of time.
Production's information	Information of the cultivated product: • product: the type of production cultivated in the field • planted date • collected date • planted amount • collected amount • fertilizers used
Farmer's Notification	There are two kinds of notification a farmer can receive: • The policy maker sends to a farmer a message which notify he/she will receive an incentive and is encouraged to create a forum to give advice to other farmer. • The agronomist sends a message in response to the request for help of the farmer where he/she explains when he/she will come to help the farmer.

Suggestions	The suggestions are tips given to the farmer directly by the system, interacting with a database created by us with the help of some agronomists which contains the best products to cultivate and fertilizers to use depending on the location. There are two types of suggestions a farmer can ask for: • suggestions about products: the system shows the best product to cultivate in a certain location • suggestions about fertilizers: the system shows the best fertilizer to use in a certain location
Request for help	It means that a farmer needs help and the request, which can be sent either by a farmer or by a policy maker, is addressed to an agronomist, not to the system or to other farmers.
Agronomists Platform	This is a platform which is external to DREAM. It is used by the agronomist to receive the requests for help sent by the farmers or by the policy maker in order to help the farmers and to notify the farmers about the date of the intervention.
Weather.com	This is a platform which is external to DREAM and provides the information about the weather conditions.
Ranking	The system compute a ranking for each product, in order to allow the Policy Makers to identify the best and the worst farmers. Obviously different products imply different rankings. The ranking is based on the Score.
Score	A value calculated by the system which is assigned to each Production, in order to insert it in the corresponding Ranking in a position based on this value. More information about it are in the subsection 2.2.
Secret Code	This is a secret code, a Nonce, which is given by us to the Policy Makers when the final version of the DREAM will be released. They will use it in order to Register as PolicyMaker into the system. It is a way to not allow Guests who are not real Policy Maker to register in the system as PolicyMaker and see all the information provided by the farmers and to improve the security of the system.

1.4 Revision history

Version	Description
Version 0.1	First release
Version 0.2	• Delete World Phenomena W10 "The agronomist receives request for help from the farmer" • Insert a new Shared Phenomena SP30 where Policy Maker send a message to the Agronomist

Version 0.3	• Review of Introduction • Delete SP14 and S15 because they had already specified in SP21 ans SP23 • Add SP25 to explain better the role of ranking in the system. • Delete S28 and S29 • Added UML diagrams • Improved some use cases • Changed actors • Better explained the scenario "The farmer creates a new forum"
Version 0.4	• remove some assumptions, divided assumption D8 in D8 and D9 • added some definitions • added the main user interfaces • added some alloy code (to complete)
Version 1.0	• improved the description of the User Interfaces • completed the alloy code and added the results • added another definition • better description of the Product Functions • Mapping of GOALS
Version 2.0	• Changing class diagram • Elimination of Sensor in the class diagram. The system connects with an API to an external system. The system request only the data to calculate the algorithm for the score.

1.5 Reference Documents

- Repository Data4Policy `https://github.com/UNDP-India/Data4Policy`
- Telagana's information `https://en.wikipedia.org/wiki/Telangana`
- Smart Farming `https://www.iotforall.com/smart-farming-future-of-agriculture`

- A comprehensive review of Data Mining techniques in smart agriculture https://www.sciencedirect.com/science/article/pii/S1881836619301533

1.6 Document Structure

The document is composed by 5 chapters, as describe below:

1. **Introduction:** it is described the purpose of the project and the goals of DREAM. Moreover, it defines the scope of the project, and it is included the analysis of the world and the analysis of the shared phenomena.

2. **Overall Description:** it is described the product perspective with all the scenarios and further details on the shared phenomena and the domain model. Few diagrams are included to describe better the product. In this section it is included also, the most important requirements and the domain assumptions we have done to create the system.

3. **Specific Requirements:** represents the body of the document. It describes in details the requirements and non-functional requirements. First of all there is the description of the system trough external interface requirements where it is described the user interfaces, hardware and software interfaces. Secondly, there is the Functional requirement using use cases description and sequence diagrams. In order to reach the goals described in section 1 there are also all the requirements needed and they are described through sequence diagrams and state charts. Then there are the mapping on goals where goals, domain assumption and requirements are linked. Lastly there is the description of non-functional requirements trough Design Constraints and Software System Attributes.

4. **Alloy:** In this section is included Alloy in order to perform and verify specifications described in section 3.

5. **Effort:** Show the effort done by each member of the group.

2 Overall Description

2.1 Product perspective

In this subsection it is reported a high-level class diagram. It contains only few essential attributes for the various classes and does not include every class that will be necessary to define the Model of the system.

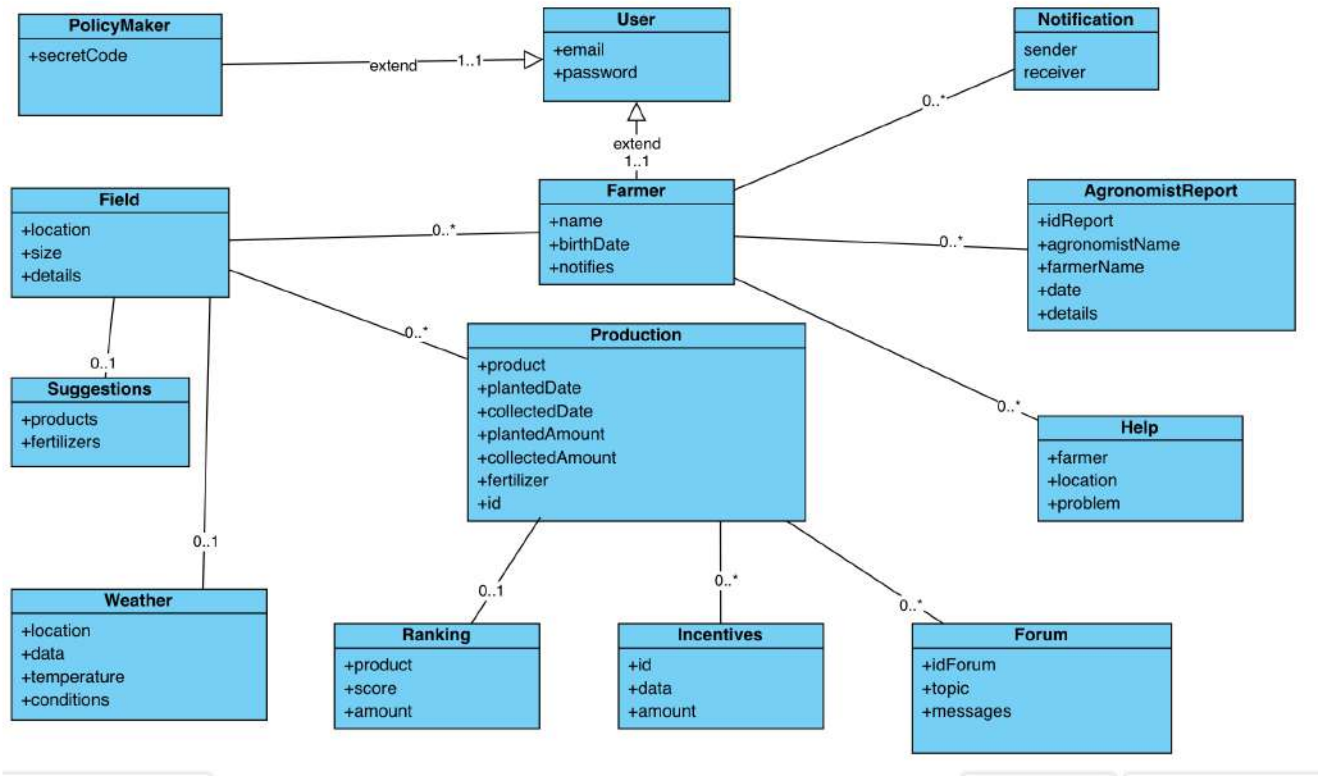


Figure 2: Class Diagram of Dream

2.1.1 Scenarios

- **Registration of a farmer:**

  The farmer Bob wants to register into the system. He accesses the system and selects the option to register as a farmer. The system asks him for his information. He provides all the data and confirms the operation. After his confirmation, the system asks him to check a message inside his mail, to confirm his registration through a link. Bob checks his email and finds a message from Dream, containing the link. He clicks the link and confirms the registration.

- **The farmer sees weather conditions:**

  Bob wants to see the weather conditions for the week, in order to decide whether or not to water the artichoke field. He enters in the the system and logs in with his email and password, then selects the Weather button. He enters in a new page and he selects the date and the location of the field, so that the system shows him the weather condition for that date in that location.

- **The farmer adds a new field:**

  Bob has bought a new field, so he wants to add the field in the system. He is already registered, so he logs in and he enters into the first page and selects the Field button. The system shows a new page with a table which rows contain information of each land own by Bob. He selects the button Add and the system shows a pop up. The system asks him for the information of the field as location, size and details of terrain type (these details are not required). He provides all the data and confirms the operation. After the confirmation Bob sees the new land into the table of the land page.

- **The farmer adds a Production:**

  Bob wants to provide to the system information about his cultivation of courgettes. Once he logs into the system, and the Main Page appears, he clicks on Products. The systems shows a table with the products he had inserted before, with the name and the field in which that product is cultivated. Then he clicks on the Add button, and a popup appears on the screen. He fills the text fields of this popup with the name of the product he is adding, the land in which he is cultivating it, the amount of planted seeds, with the date, the amount of collected product, with the date and the fertilizer used (if used). Then he clicks confirm in order to close the popup and to see the new row with the product in the table.

- **The farmer asks for suggestions:**

  Alice, a farmer, has just added a field in the system, and she wants suggestions on the best plants to grow in that field. After the log in, in the Main Page, she clicks on the Suggestions button. A new page appears with a table with previous suggestions and a button to ask a new one. She clicks on that button and the system shows a popup by which she can select the field for which she wants suggestions (among the ones she added) and select if she needs tips for products, fertilizer or both. She clicks on both and the system shows on the display the products and fertilizer best indicated for that field.

- **The farmer needs help:**

  Bob faced a problem and thinks that he needs help of an agronomist. So he logs into the system and select the “Need for help?” option in the Main Page. A new page appears, in which he can specify the land for which he needs help (or not if the problem is more general) and describes what he needs in the text field. He clicks on the Confirm button, so that the request will send to an agronomist. Now he can wait for a response.

- **The farmer creates a forum:**

  Mark, a young farmer who is a bit inexperienced, wants to chat with other farmers to learn their techniques and shortcuts. Once he logs in the system, and the Main Page appears, he clicks on the Forum button. A new page is displayed, in which there is a list of the lives forum created by other farmer who are chatting. He can join one of them, but he decided to create a new one. He clicks the Button "New Forum", a new Pop up appears. The system asks for the Topic and a a description of the Topic, he clicks the button "Ok", and a new Forum is added in the system. Now he can chat with the farmers who join in that forum.

- **The farmer checks notifications:**

  Bob wants to know notifications. After entering in the system, he selects in the Main Page the Email button. The system shows a new page with all messages received. There it can be either incentives sent by the policy makers or the agronomist’s response to the request for help . He clicks on the notification and reads the message.

- **Registration’s Telagana’s policy maker:**

  Tom is a policy maker who wants use the system. He enters in the DREAM’s page and select the button sign up as policy makers. He inserts his secret code and confirm. The system sends him an email with credentials. Tom checks the email, takes the credentials and he re-open the Main page of DREAM. He logs in and he puts the credentials given by the system and he presses the button Access.

- **Telagana's policy maker checks field information**:

  Tom a policy maker wants to see the information of a particular field, so after logging in, the system shows the home page and Tom presses the Fields button. At this point the system displays a page with a table showing the emails of the farmers and their respective fields. Tom searches for the farmer he is interested in, but wants more information so he presses the field associated with the farmer. This opens a new page with a table with all the productions cultivated in that field. In particular the system shows the production information.

- **A Policy Maker wants to identify those farmer who are performing well:**

  Steven, a policy a maker, wants to identify the farmers who are performing well. To do this he logs into the system and, after the Main Page appears, he clicks on the Rankings button. The system displays a new page, where he can select the product for which he wants to see the best farmers. Steven selects peppers. A table appears on the screen, with farmer's email, the score regarding the production calculated on the section 2.2.

- **A Policy Maker wants to identify those farmer who are performing bad:**

  Steven,a policy maker, after following the steps explained in the previous point, clicks on the Order button, to reverse the order of the table which will shows the farmers in ascending order of score. Therefore there is a column with a button sends Agronomist in order to help farmer who are performing bad .

- **A Policy Maker wants sends incentives to the farmers who are performing well:**

  John, a policy a maker, wants to send incentives to the farmers who are performing well. To do this he follows the steps explained in the Scenario "A Policy Maker wants to identify those farmer who are performing well". Then John clicks on the Notify button for the two first table rows, so for the first and second farmer who performed better. When he clicks on the button of the second row a new page appears. It notifies the policy maker that the farmer selected has already received an incentive for that product. Hence only first farmer will receive a notification which explains that they will receive an incentive via bank transfer and invites them to create forums in order to explains their techniques and give advice.

- **A Policy Maker wants send an agronomist in order to help farmer who are performing bad:** John, a policy maker, wants to send an agronomist to the farmer who are performing bad. He follows the same steps explained in the Scenario "A Policy Maker wants to identify those farmer who are performing bad". Then John clicks on the button "Send Help" in order to send a message to an agronomist who will help the farmer.

- **A Policy Maker wants to see those farmers who received incentives:**

  John, after following the steps explained in the Scenario "A Policy Maker wants to identify those farmer who are performing well", clicks on the Incentives history so that a new page appears, which contains a table with information about each farmer who received an incentive, the date and the amount of the prize.

- **Telagana's policy maker checks agronomist:**

  Tom a policy maker wants to understand if John the agronomist is working well. After logging in the system, the main page appears and he selects the button Agronomist. The system shows a new page where there is a table with all the works of all agronomist, the farmers that they had helped, the land and the date associate to the agronomist. Tom finds John and selects his name, a new page is showed. Tom reads the Report Detail of the agronomist, what he has done and for how long. Moreover the system shows the score before and after the intervention.

2.2 Product functions

DREAM
The goal of this system is to help Telengana's government to design models for food system, analyzing the results of the work of the farmers. To do this it allows the farmer to upload data about their fields and cultivation, and combine them with other data, like weather conditions, amount of water used to irrigate the fields, humidity soil and so on.
Beyond the contribution it makes to policy maker in order to take decisions for the future, it gives advice to farmers, allows them to communicate, and to ask for help, if necessary, to governmental agronomists.

- **Give suggestions to the farmers:** the farmers can select the option for suggestion linked to a specific field which are provided directly by the system. So in this case these advice do not become from other farmers. Once they ask for suggestions (specifying a land) the system will display the best products to cultivate and fertilizers to use for the field the farmer specified.

- **Give help to the farmer:** the farmers, thanks to the system, can ask for help to the agronomists, specifying a field and the kind of problems they faced. An agronomist will receive the request and provide help.

- **Farmer checks weather conditions:** the farmers can check the weather conditions based on the locations of the field in order to decide to irrigate or not a fields.

- **Farmer interacts each other:** the farmers can create a forum, or join to a forum created by other farmers in order to discuss with them about a specific topic, or ask them other suggestions that the system does not provide.

- **Policy maker sees information about farmers:** a policy maker can see all the information about a farmer: all his/her fields' information, his/her productions' information, the weather condition he/she faced during his/her productions, the amount of water used to irrigate his/her fields and the soil humidity levels.

- **Policy maker identifies farmers who perform well:** the system, with a specific algorithm, that takes in account different variables (field location and size, weather condition, farmer's age, amount of collected product and so on), computes the productivity of each farmer, and assigns each of them a score, which is the parameter on which the ranking is generated. This score is calculated for each farmer and for each product he cultivates. So there will be differentiated rankings per product. This allow the policy makers to identify the farmer who are performing well (so that they can receive incentives and help others).

- **Policy maker identifies farmers who perform bad:** in the same way that the system shows the farmer who perform well. The system allow to change the ranking order: from the worst to the better one. In this way the policy's maker can decide the farmer who needs help (so that they can send a message to an agronomist to help him/her)

- **Policy maker checks the agronomists' work:** if an agronomist provides help to a farmer, the policy maker can check in the system the report of that intervention: the farmer and the field in which it happened, the score of that farmer before and after that (hence his or her improvements) and further details.

- **Database Interaction:** DREAM is able to store the data inserted by the User, provided by the Sensors and the Suggestions in a database. It processes and manages the data in order to give to the User only what he/she asks for. This theme will be better analyzed in the Design Document.

- **Ranking Algorithm:** the system, thanks to the data collected and stored in the database compute a ranking based on the Score. The Score is calculated in function of different factors: field's size, weather conditions faced during the period of the production, amount of product cultivated, amount of water used, average level of humidity soil and farmer's age. However the algorithm which calculates the score will be better explained in the Design Document.

2.3 User characteristics

The system involves the following actors:

- **Guest**
  A person who still needs to register to the system in order to use what it offers. If he/she works as policy maker he/she as a secret code to use to register as Policy Maker, otherwise he/she can register as Farmer.

- **User**
  A person who is registered in the system, and can log in using his/her credentials (email and password) in order to use its functionalities. There are two type of User:
  - **Farmer**
    A farmer who is registered and, once he logs in to the system, can interact with it: he can add fields, information about what and how he or she produces, ask for help and suggestions.
  - **Policy Maker**
    A policy maker who is registered and, once he logs in to the system, can interact with it: he can check the works of the farmers, of the agronomist in order to take decisions about the future.

Pay attention: in this document, when Farmer and Policy Maker are mentioned, reference is made to farmers and policy makers who are already registered in the system.

Therefore, the agronomist, mentioned before, do not need to register and log in to the system, because they don't use it directly, but they can share their reports to the policy makers and see the messages sends by the farmers or by the policy makers thanks to other platforms they use, as the system is able to share contents with them. So they are not considered as real actors involved in the system.

2.4 Assumption, dependencies and constraint

2.4.1 Domain Assumptions

Domain Assumption	Description
D1	Personal data given by the farmer during the registration process are assumed to be correct
D2	The farmer is assumed to confirm his/her account by the link sent by email
D3	Data inserted by the Farmer during the log in process are assumed to be correct
D4	Information about the weather provided by the API are assumed to be correct
D5	Information about the field the farmer is adding are assumed to be correct
D6	Information about the cultivated product the farmer is adding are assumed to be correct
D7	The system is assumed to be able to give suggestions about a product related to a specific field to the farmer

D8	The system is assumed to be able to give suggestions about a fertilizer related to a specific field to the farmer
D9	It is assumed that the farmer's request for help is received by the agronomist
D10	A farmer is assumed to join in a forum created by another farmer
D11	A farmer is assumed to write messages in a forum
D12	A farmer is assumed to check his/her notifications
D13	Policy maker's secret code is assumed to be correct
D14	The Policy maker is assumed to check is/her email to discover the login credentials
D15	Data inserted by the Policy Maker during the log in process are assumed to be correct
D16	The score of the farmers calculated by the system is assumed to be correct
D17	The farmer is assumed to have received help by the agronomist
D18	The water irrigation system that provides data about the amount of water used is assumed to work correctly
D19	The sensors that provides data about the humidity soil are supposed to work correctly
D20	It is assumed that the policy maker's request for help the farmer is received by the agronomist

2.4.2 Dependencies

- Dream will use an external API to provide data about weather conditions, interacting with an external platform such as Weather.com.
- Dream will use an external API to get the data collected by the humidity sensors of the soil.
- Dream will get the data collected by the water irrigation system.
- Dream will use an external API to communicate with the AgronomistsPlatform
- Dream will use an algorithm to calculate bad and good farmer depending on the farmers' resistance to adverse weather conditions, water used to cultivate, the type of soil, amount of product collected, fertilizer used

2.4.3 Constraints

- Each field is own by only a farmer
- It can't be more than a production in a field in the same period of time.
- The farmer can insert a new Production in the system when he has already collected the product.
- User email is unique, two user can't have the same email.

3 Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interface

In this subsection are shown the main interfaces of the system. The FirstPage, which is the first page that DREAM shows, where a Guest can register into the system or a User can log in.

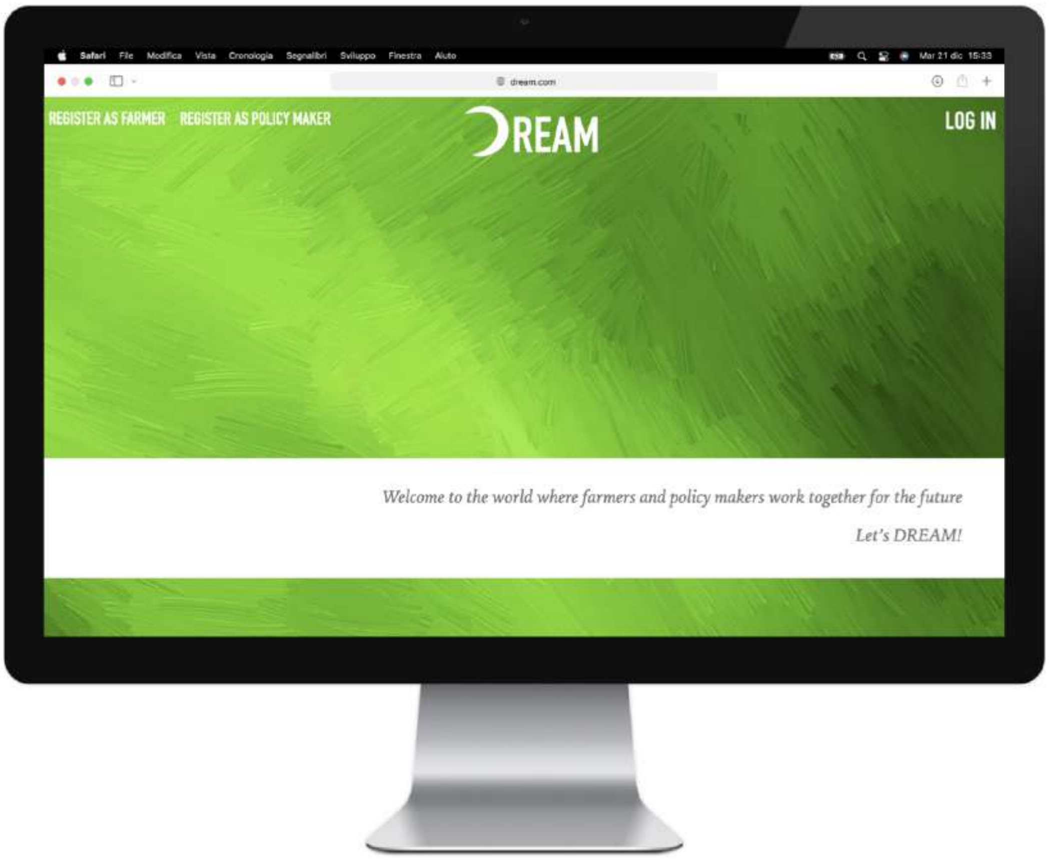

REGISTER AS FARMER
REGISTER AS POLICY MAKER
DREAM
LOG IN
Welcome to the world where farmers and policy makers work together for the future
Let's DREAM!

Figure 3: FirstPage of DREAM

The FarmerPage is the main page of the farmer, where he/she can decide what system's features he/she wants to use clicking on one of the buttons shown: Weather button, Field button, Production button, Suggestion button, Help button and Notifications button.

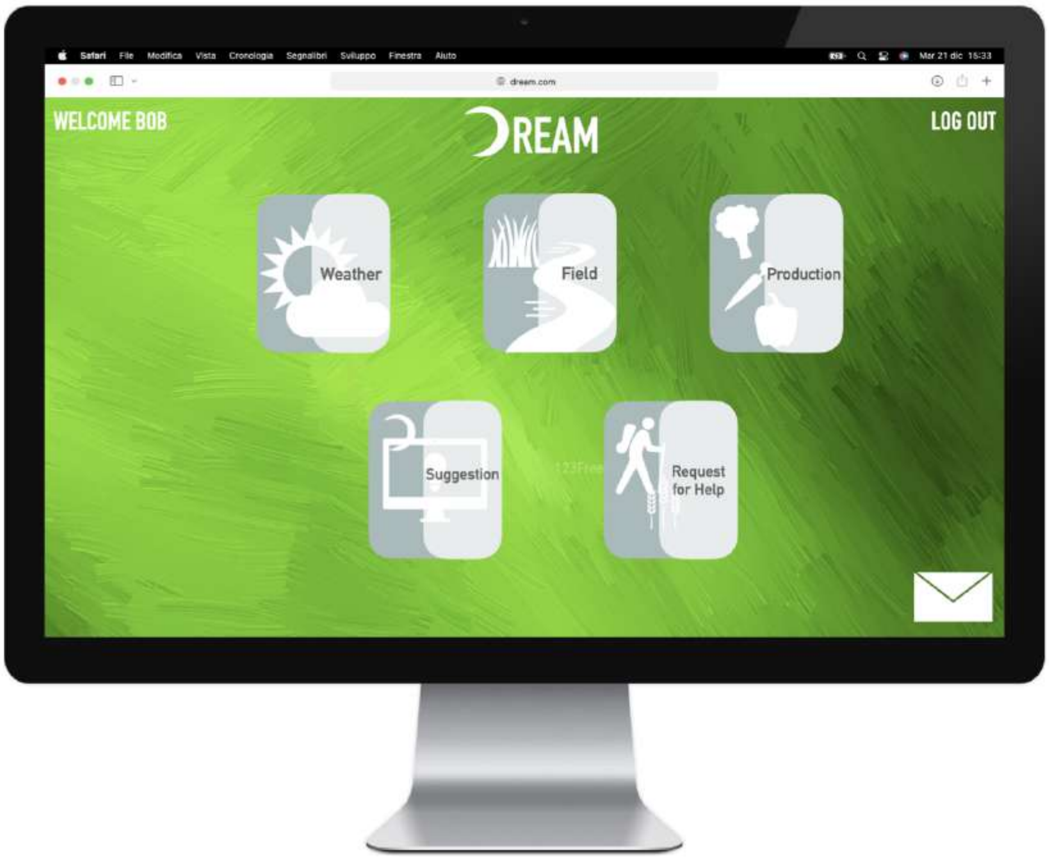


Figure 4: FarmerPage of DREAM

The PolicyMakerPage is the main page of the PolicyMaker, where he/she can decide what system's features he/she wants to use clicking on one of the buttons shown: Fields button, Ranking button, AgronomistsReport button.

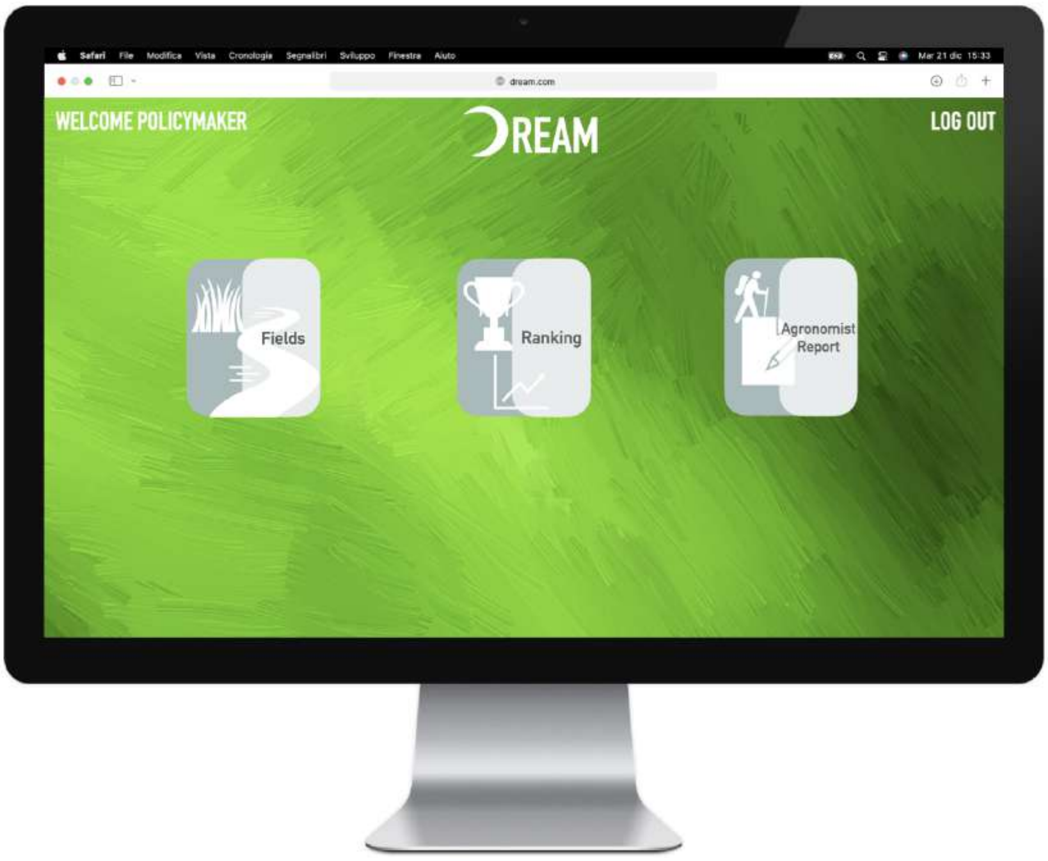


Figure 5: PolicyMakerPage of DREAM

All the other user interfaces will be described more in detail in the Design Document. In this subsection we simply list them in order to help the reader to better understand the use cases and the sequence diagrams of the subsubsection 3.3.2. Farmer's interfaces:

- WeatherPage, where the Farmer can check the weather forecasts
- FieldPage, where the Farmer can check his/her fields, and add a new one
- ProductionPage, where the Farmer can check his/her productions, and add a new one
- SuggestionPage, where the Farmer can asks for suggestions to the system
- RequestForHelpPage, where the Farmer can asks for help to the agronomists
- ForumPage, where the Farmer can join a forum, or create a new a one
- NotificationsPage, where the Farmer can checks his/her notifications and through which he can select a notification and read it in detail in the SelectedNotificationPage

PolicyMaker's interfaces:

- FieldsPage, where the PolicyMaker can sees the table of farmers and their fields, and through which can access to the ProductionPage in order to see the productions of a Field and the productions' information

- RankingPage, where the PolicyMaker can see the rankings of farmers, selecting a product. He/She can sends notificaitons of incentives to the farmers who are performing well and requests for help to the agronomists for the farmer who are performing bad. Through the RakingPage, the PolicyMaker can access to the IncentivesHistoryPage, in order to see the history of incentives already sent to the farmers for the production of the product selcected in the RankingPage
- AgrnomistsReport, where the PolicyMaker can checks the reports of the agronomists who had helped the farmers and select one of them to see it in detail in the SelectedAgronomistsReportPage

3.1.2 Hardware Interfaces

Since DREAM is a Web system only a computer or a device is needed. The first one has to have a browser rendering, the second must to connect to the network in order to connect with the server and exchange data.

3.1.3 Software Interfaces

External services and data accessible via API are needed.
In fact, there are databases that collect data with reference to information from the fields, useful for defining the capabilities of Farmers. There are:

- weather forecasts;
- water irrigation system data;
- humidity soil sensor data.

3.2 Functional Requirements

3.2.1 Use case diagrams

- Farmer

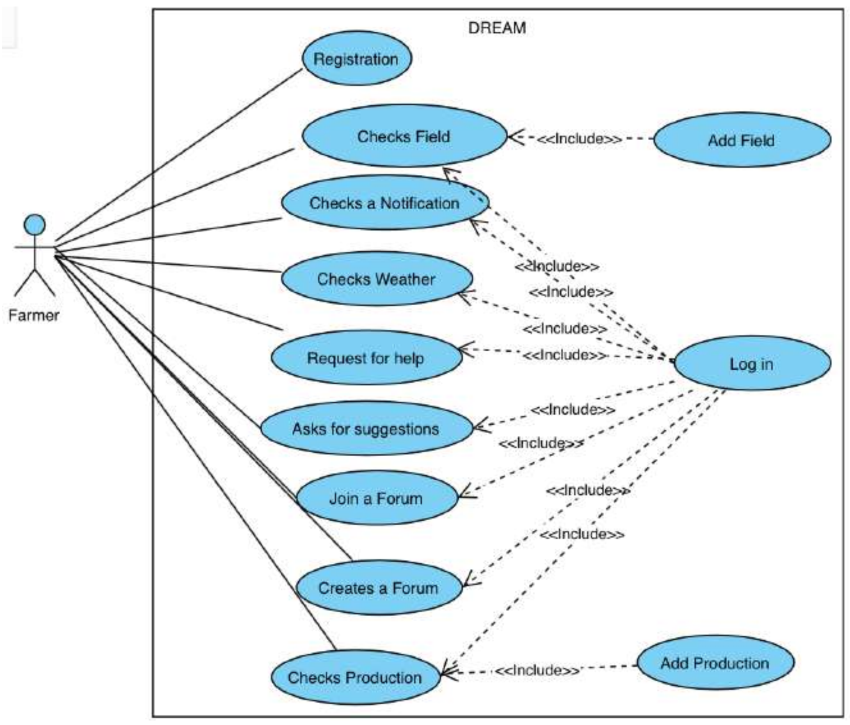


Figure 6: Farmer's use cases

- Policy Maker’s use cases

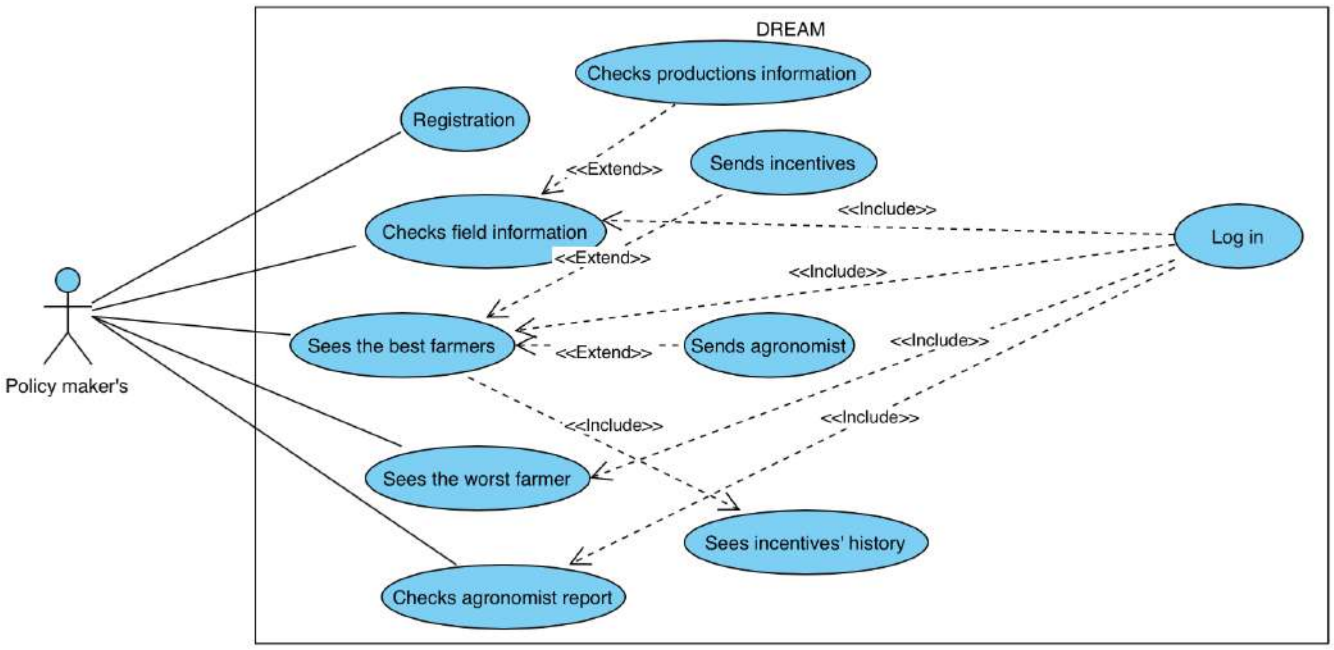


Figure 7: Policy Maker’s use cases

3.2.2 Use cases

Name	Farmer Registration
Actor	Guest
Entry conditions	The Guest is on the FirstPage of the system
Event flow	1. The Guest selects the option to Register as Farmer 2. The system asks for his/her information 3. The Guest enters all his/her information and confirms 4. The system asked the Guest to check the email where there is the link to confirm the registration process 5. The Guest checks the email 6. The Guest checks the received email and clicks on the confirmation link 7. The system stores the information
Exit conditions	The Guest successfully registered on DREAM and now he/she can login
Exceptions	• The Guest's email is already existed in the system • The information are not valid during step 2 If one of above situations happens, the system will return an error message where it is written the error and return the registration form page

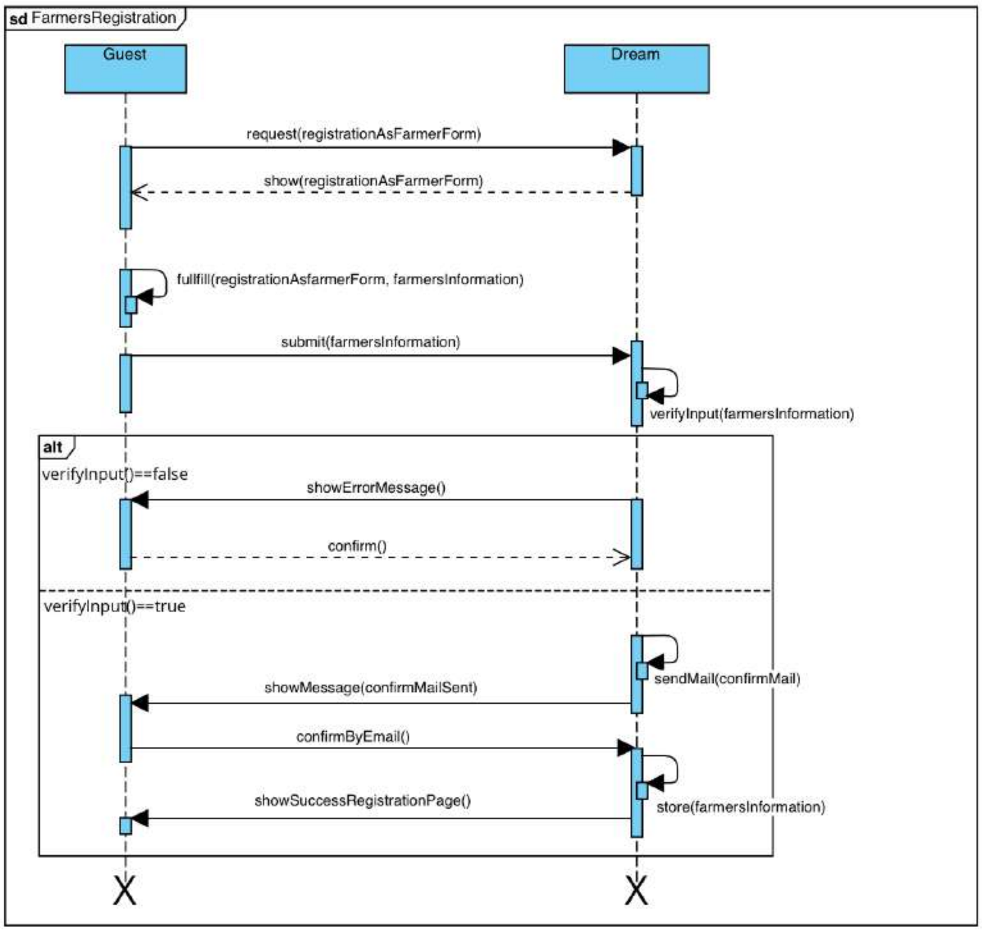

sd FarmersRegistration
Guest
Dream
request(registrationAsFarmerForm)
show(registrationAsFarmerForm)
fullfill(registrationAsfarmerForm, farmersInformation)
submit(farmersInformation)
verifyInput(farmersInformation)
alt
verifyInput()==false
showErrorMessage()
confirm()
verifyInput()==true
sendMail(confirmMail)
showMessage(confirmMailSent)
confirmByEmail()
store(farmersInformation)
showSuccessRegistrationPage()

Figure 8: Sequence Diagram concerning the registration of the Farmer

Name	Policy Maker Registration
Actor	Guest
Entry conditions	The Guest is on the FirstPage of the system
Event flow	1. The Guest selects the option to Register as a Policy Maker 2. The system asks for the secret code and confirms 3. The system checks the validity of the secret code 4. The system sends an email with his/her credentials to the Guest 5. The Guest checks the email with the credentials
Exit conditions	The Guest successfully registered on DREAM and now he/she can login
Exceptions	• The Guest's secret code is not valid • The system shows an error message and asks to reinsert the secret code The system will return an error message where it is written the error and returns the registration form page

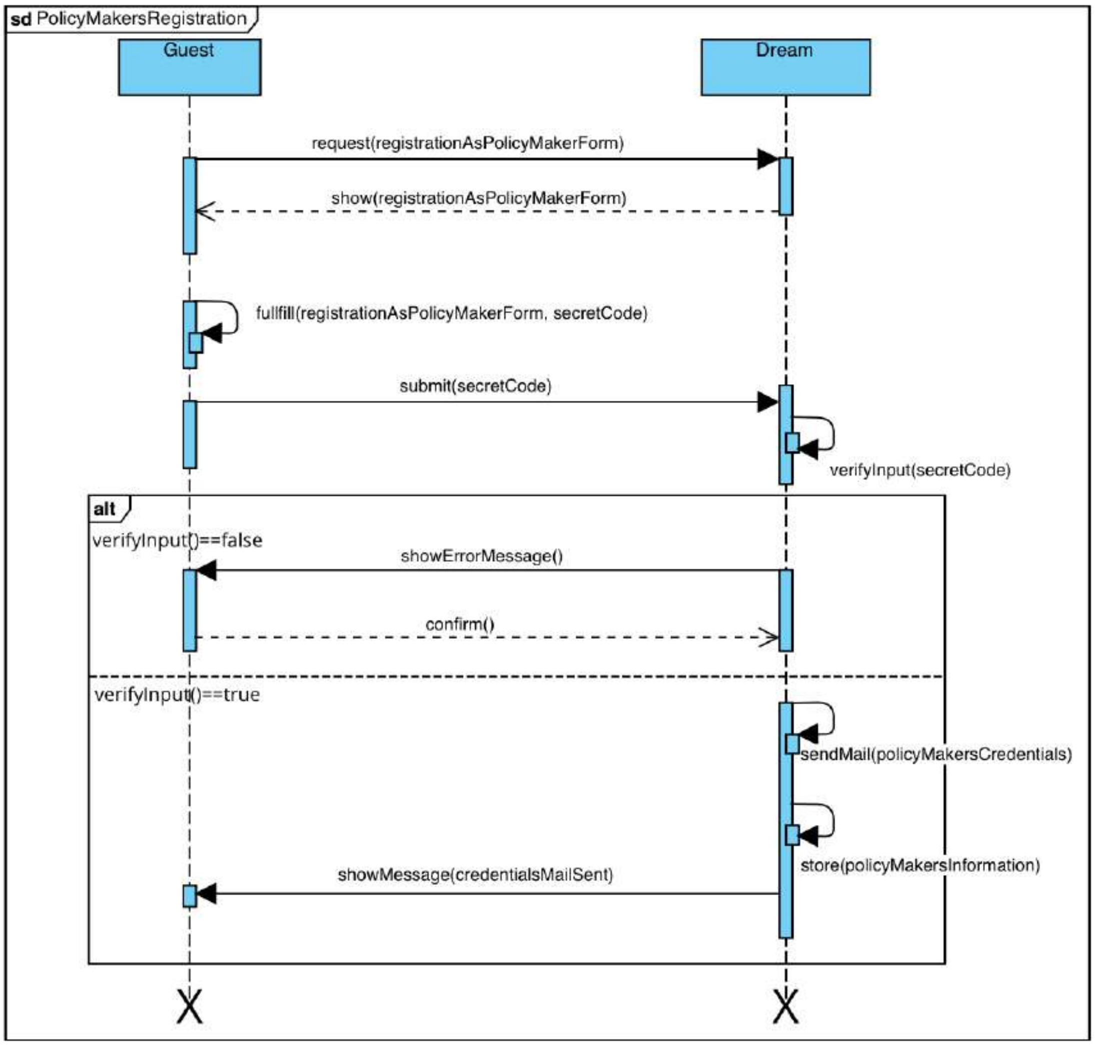

sd PolicyMakersRegistration
Guest
Dream
request(registrationAsPolicyMakerForm)
show(registrationAsPolicyMakerForm)
fullfill(registrationAsPolicyMakerForm, secretCode)
submit(secretCode)
verifyInput(secretCode)
alt
verifyInput()==false
showErrorMessage()
confirm()
verifyInput()==true
sendMail(policyMakersCredentials)
store(policyMakersInformation)
showMessage(credentialsMailSent)

Figure 9: Sequence Diagram concerning the registration of the Policy Maker

Name	Farmer checks weather conditions
Actor	Farmer
Entry conditions	The Farmer is already logged in the system The Farmer is on the FarmerPage of the system
Event flow	1. The Farmer selects the option to log in 2. The system asks for his/her credentials: email and password 3. The Farmer puts the credentials and confirms 4. The system shows the page of Farmer 5. The Farmer clicks the button of the weather 6. The system shows a WeatherPage and asks for period and location. 7. The Farmer puts the position and the period that he/she wants to checks 8. The system shows the weather and the temperature.
Exit conditions	The system shows the weather conditions and the Farmer can go back on the main page.
Exceptions	• The Farmer's email and password are not correct during step 3 • The system shows an error message and asks again the credentials • The Farmer does not put a correct position or correct period of time The system shows an error message and ask again to insert the information

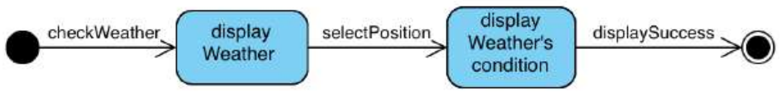


Figure 10: State Charts : checks Weather conditions

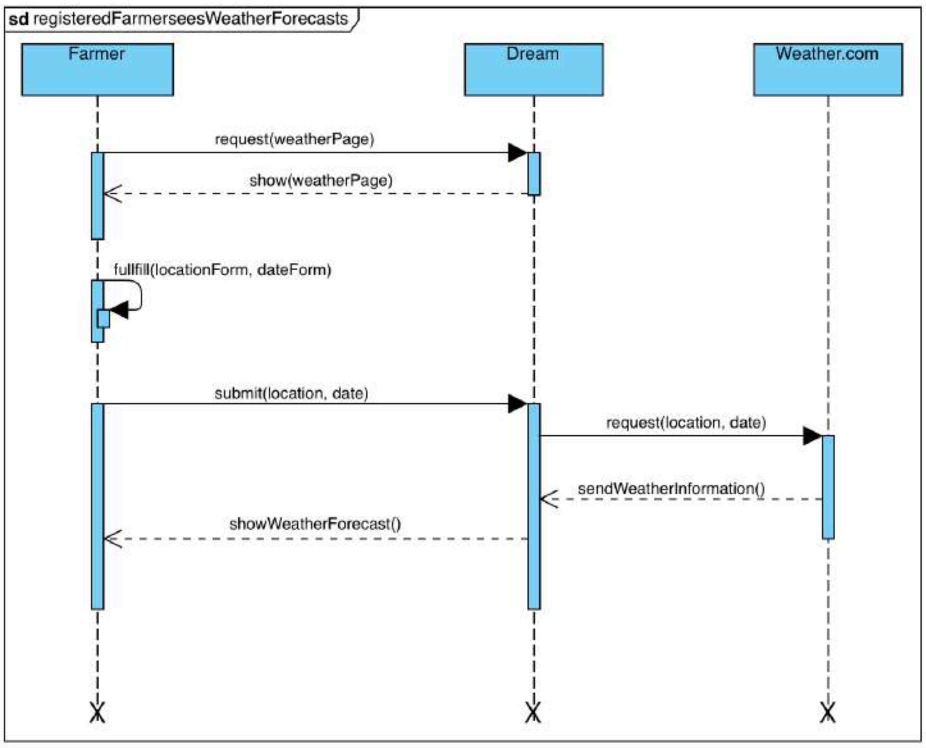


Figure 11: Sequence Diagram:Farmer checks weather conditions

Name	Farmer checks her/his fields
Actor	Farmer
Entry conditions	The Farmer is already logged in the system The Farmer is on the FarmerPage of the system
Event flow	1. The Farmer selects the Field button 2. The system shows the FieldPage with a table with all the fields owned by him/her
Exit conditions	The system shows the table and the Farmer can go back on the FarmerPage
Exceptions	

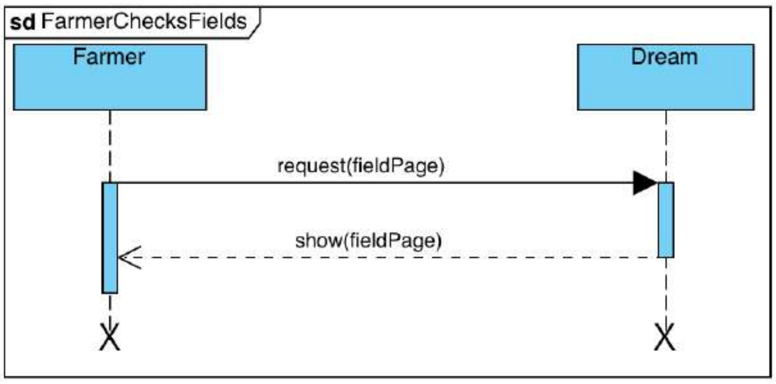


Figure 12: Sequence Diagram: Farmer checks fields

Name	Farmer adds a new field
Actor	Farmer
Entry conditions	The Farmer is already logged in the system The Farmer is on the FarmerPage of the system
Event flow	1. The Farmer selects the Field button 2. The system shows the FieldPage with a table with all the fields owned by the Farmer 3. The Farmer selects the button Add 4. The system shows a new pop-up 5. The system asks him/her the field's information 6. The Farmer puts field's information 7. The Farmer select the button Ok 8. The popup disappears and the fields' table is updated
Exit conditions	The new field is successfully added
Exceptions	• The Farmer doesn't put the correct information of the field • The Farmer puts an already existing field in the system The system shows an error message and asks again the information

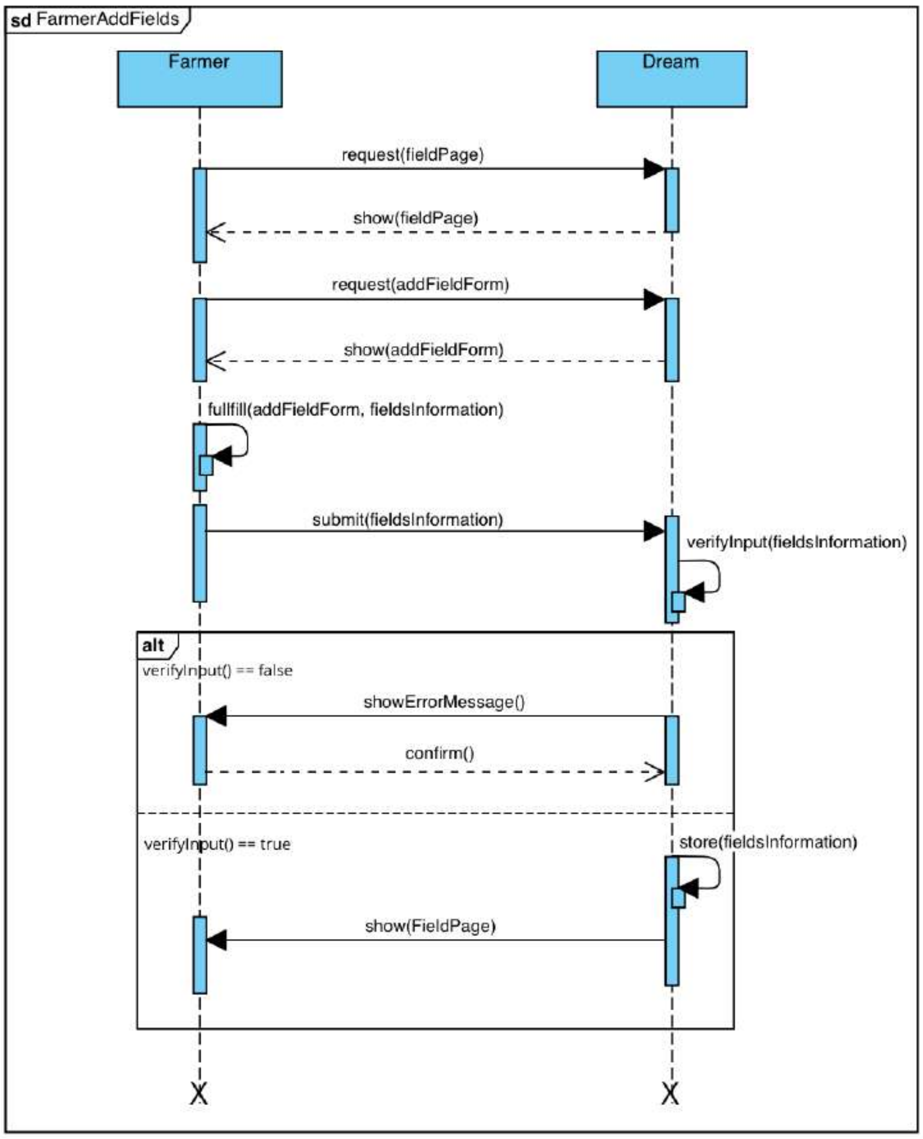


Figure 13: Sequence Diagram: Farmer adds a new field

Name	Farmer checks her/his productions
Actor	Farmer
Entry conditions	The Farmer is already logged in the system The Farmer is on the FarmerPage of the system
Event flow	1. The Farmer selects the Production button 2. The system shows the ProductionPage with a table with all the production he/she has already added
Exit conditions	The system shows the table and the Farmer can go back on the main page
Exceptions	

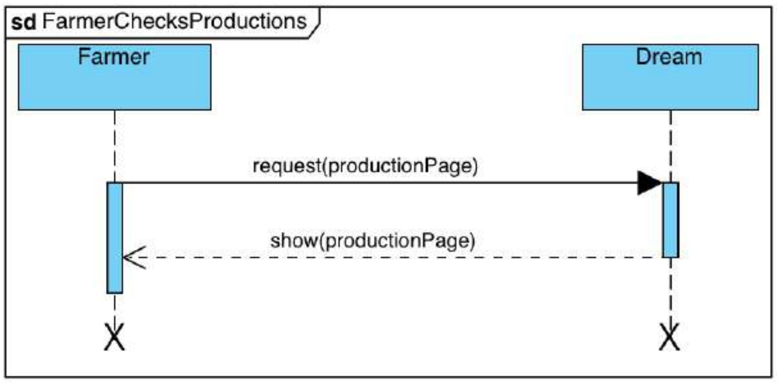


Figure 14: Sequence Diagram: Farmer checks his/her productions

Name	Farmer adds a new production
Actor	Farmer
Entry conditions	The Farmer is already logged in the system The Farmer is on the FarmerPage of the system
Event flow	1. The Farmer selects the Productions button 2. The system shows the ProductionPage with a table with all the productions and productions' information 3. The Farmer selects the Add button 4. The system shows a new pop up 5. The system asks for production's information 6. The Farmer puts the production's information 5. The Farmer selects the button Ok 8. The popup disappears and the productions' table is updated
Exit conditions	The new production is successfully added
Exceptions	• The Farmer adds a production in a field that have already a production in that period • The Farmer misses some information In both cases the system shows an alert

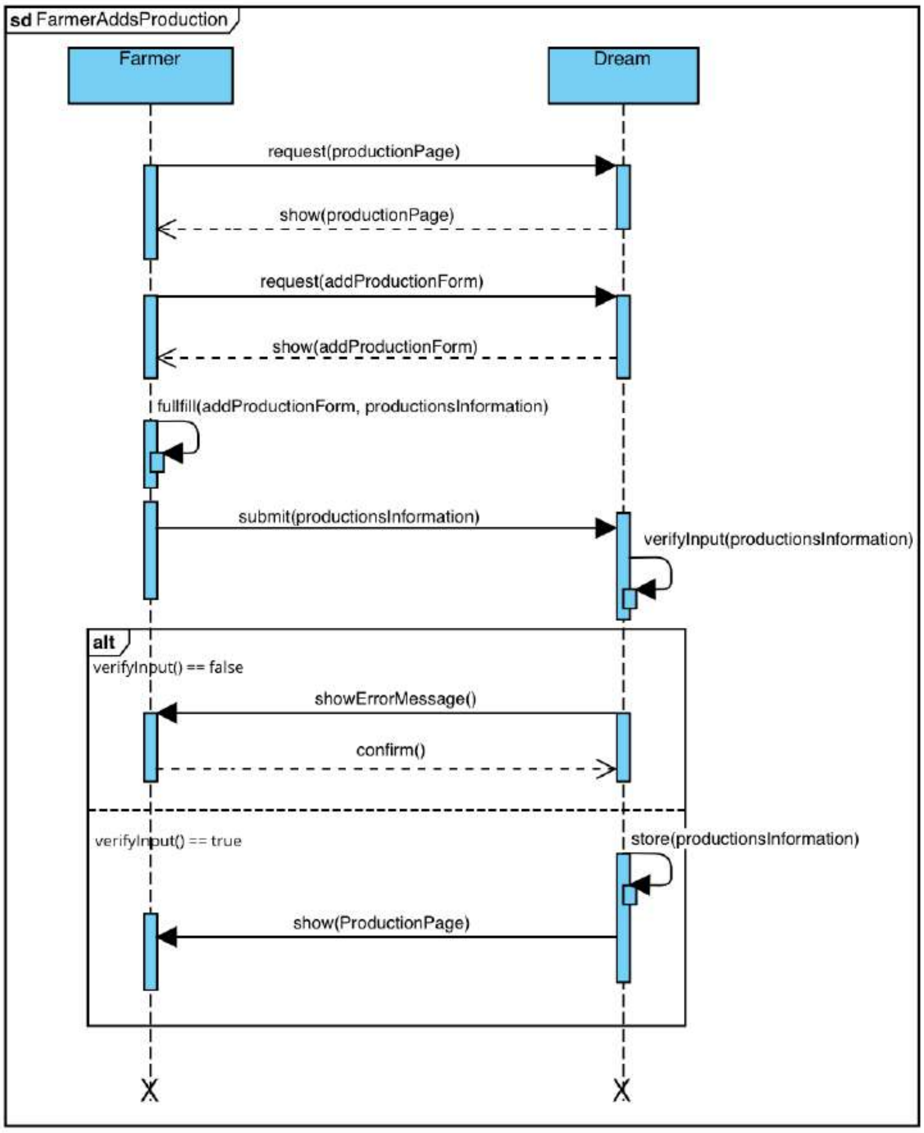


Figure 15: Sequence Diagram: Farmer adds a production

Name	Farmer asks for suggestions about a product
Actor	Farmer
Entry conditions	The Farmer is already logged in the system The Farmer is on the FarmerPage of the system
Event flow	1. The Farmer selects the Suggestions button 2. The system shows the SuggestionPage with the previous suggestions 3. The Farmer selects the NewSuggestion button 4. The system shows a new pop up 5. The system asks to select the field and the products or the fertilizer 6. The Farmer selects “product” 7. The Farmer selects the button Ok 8. The system displays the best products for the specified field
Exit conditions	The system shows the suggestions successfully and the Farmer can go back on the main page

Exceptions	• The Farmer asks for a field not included in the system • The Farmer does not select if he need suggestions for a product or fertilizer In both cases the system shows an alert

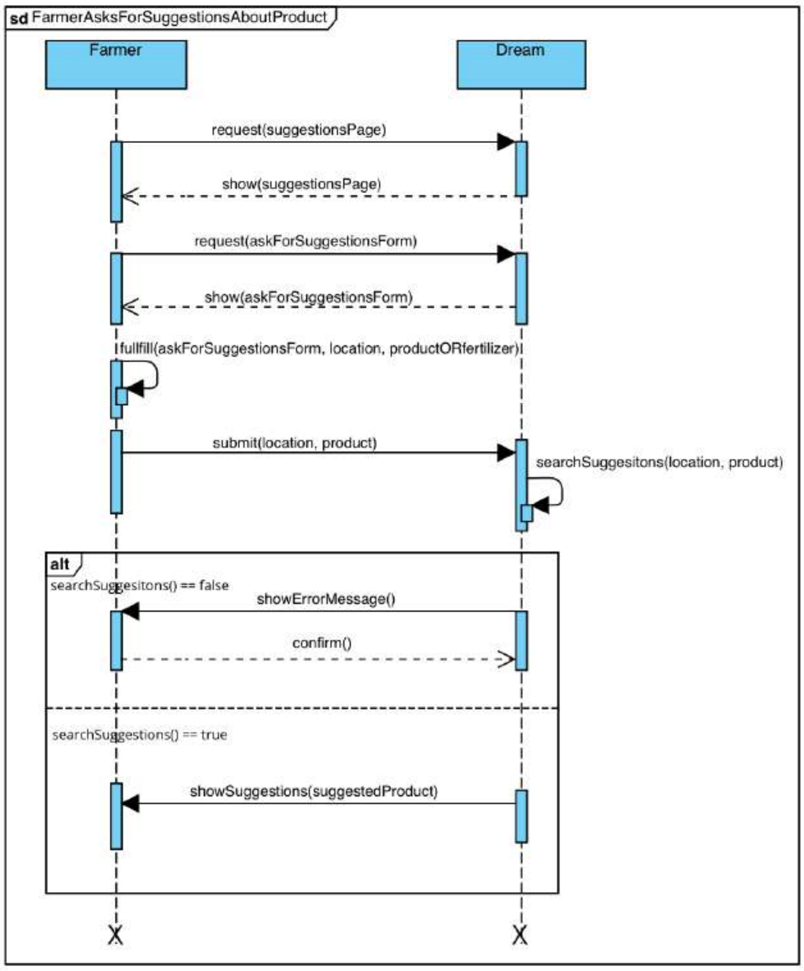

sd FarmerAsksForSuggestionsAboutProduct
Farmer
Dream
request(suggestionsPage)
show(suggestionsPage)
request(askForSuggestionsForm)
show(askForSuggestionsForm)
fullfill(askForSuggestionsForm, location, productORfertilizer)
submit(location, product)
searchSuggesitons(location, product)
alt
searchSuggesitons() == false
showErrorMessage()
confirm()
searchSuggestions() == true
showSuggestions(suggestedProduct)

Figure 16: Sequence Diagram: Farmer asks for suggestions about a product

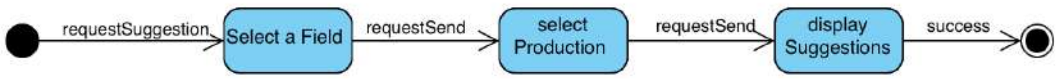


Figure 17: State Charts : Farmer ask for suggestions about a product

Name	Farmer asks for suggestions about a fertilizer
Actor	Farmer
Entry conditions	The Farmer is already logged in the system The Farmer is on the FarmerPage of the system
Event flow	1. The Farmer selects the Suggestion button 2. The system shows the SuggestionPage with the previous suggestions 3. The Farmer selects the NewSuggestion button 4. The system shows a new pop up 5. The system asks to select the field and the products or the fertilizer 6. The Farmer selects "fertilizer" 7. The Farmer selects the button Ok 8. The system displays the best fertilizers for the specified field
Exit conditions	The system shows the suggestions successfully and the Farmer can go back on the main page
Exceptions	• The Farmer asks for a field not included in the system • The Farmer does not select if he need suggestions for a product or fertilizer In both cases the system shows an alert

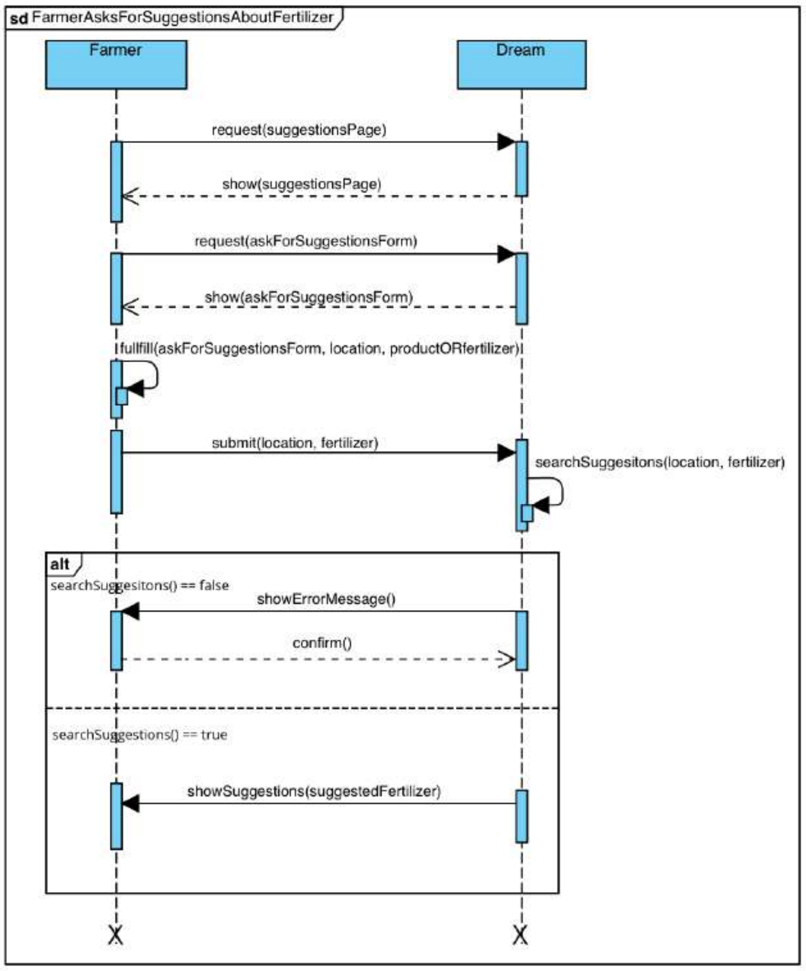

sd FarmerAsksForSuggestionsAboutFertilizer
Farmer
Dream
request(suggestionsPage)
show(suggestionsPage)
request(askForSuggestionsForm)
show(askForSuggestionsForm)
fullfill(askForSuggestionsForm, location, productORfertilizer)
submit(location, fertilizer)
searchSuggesitons(location, fertilizer)
alt
searchSuggesitons() == false
showErrorMessage()
confirm()
searchSuggestions() == true
showSuggestions(suggestedFertilizer)

Figure 18: Sequence Diagram: Farmer asks for suggestions about a fertilizer

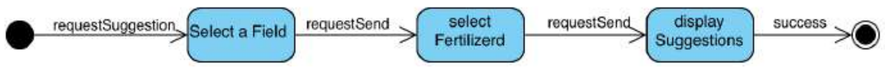


Figure 19: State Charts : Farmer asks for suggestions about a fertilizer

Name	Farmer asks for help
Actor	Farmer
Entry conditions	The Farmer is already logged in the system The Farmer is on the FarmerPage of the system
Event flow	1. The Farmer selects the "RequestForHelp" button 2. The system shows the RequestForHelpPage 3. The system asks to specify the field and the problem in the text field 4. The Farmer select the field and write information about the problems he/she faced 5. The Farmer press the button Send 6. The system displays the FarmerPage
Exit conditions	The system sends the request for help successfully
Exceptions	• The Farmer asks for a field not included in the system • The Farmer doesn't put the text in the text field In both cases the system shows an alert

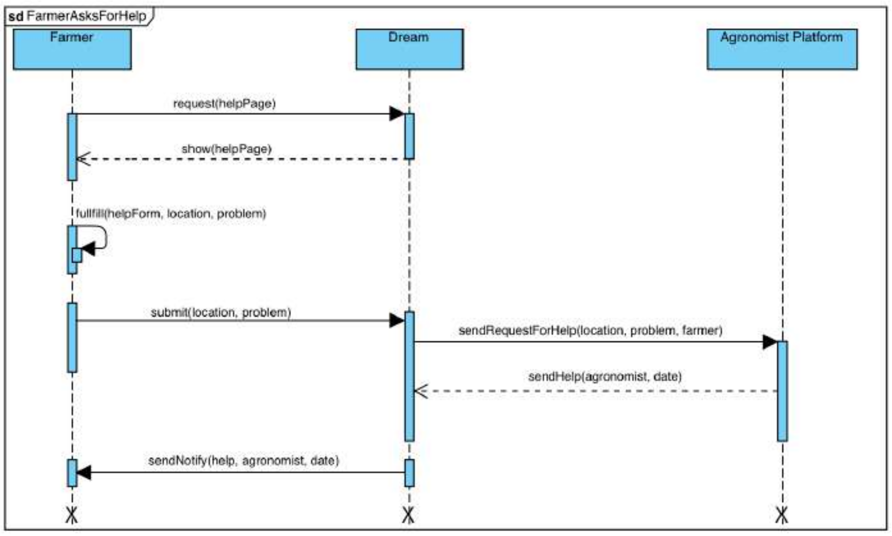

sd FarmerAsksForHelp
Farmer
Dream
Agronomist Platform
request(helpPage)
show(helpPage)
fullfill(helpForm, location, problem)
submit(location, problem)
sendRequestForHelp(location, problem, farmer)
sendHelp(agronomist, date)
sendNotify(help, agronomist, date)

Figure 20: Sequence Diagram: Farmer asks for help

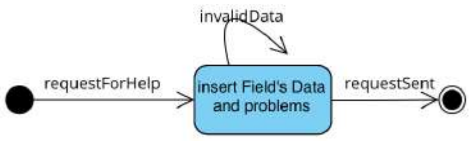


Figure 21: State Charts : Farmer asks for help

Name	Farmer creates a forum
Actor	Farmer
Entry conditions	The Farmer is already logged in the system The Farmer is on the FarmerPage of the system
Event flow	1. The Farmer selects the "Forum" button 2. The system shows the ForumPage with all the forums and the button “New-Forum” 3. The Farmer clicks on the “NewForum” button 4. The system shows a new popup 5. The system asks for the topic and the description of the forum 6. The Farmer inserts the topic and the description of the forum 7. The Farmer press Ok 8. The system shows the NewForumPage
Exit conditions	The new forum is created successfully
Exceptions	• The Farmer asks for a topic already created in another forum • The Farmer doesn’t insert the the topic • The Farmer doesn’t insert the the description In both cases the system shows an alert

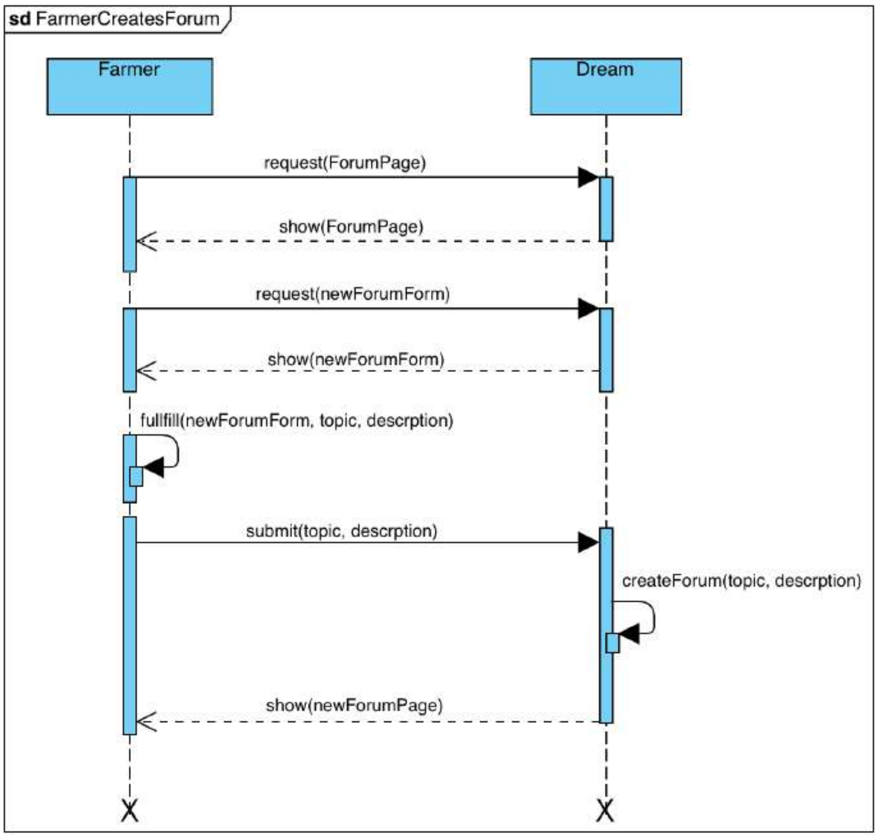


Figure 22: Sequence Diagram: Farmer creates a new forum

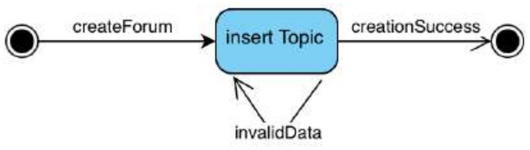


Figure 23: State Charts : Farmer creates a new forum

Name	Farmer joins a forum
Actor	Farmer
Entry conditions	The Farmer is already logged in the system The Farmer is on the FarmerPage of the system
Event flow	1. The Farmer selects the "Forum" button 2. The system shows the ForumPage with all the forums and the button “new-Forum” 3. The Farmer selects one of the existing forums and clicks on the "Join" button 4. The system open the SelectedForumPage
Exit conditions	The Farmer exits the forum
Exceptions	

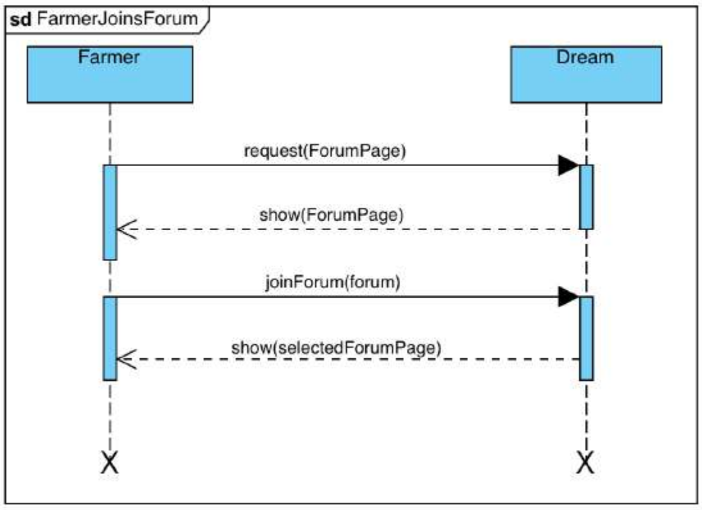


Figure 24: Sequence Diagram: Farmer joins a forum

Name	Farmer checks notifications
Actor	Farmer
Entry conditions	The Farmer is already logged in the system The Farmer is on the FarmerPage of the system
Event flow	1. The Farmer selects the "Notifications" button 2. The system shows the NotificationsPage 3. The Farmer reads the new notifications and select one 4. The system shows a new popup with the notification selected 5. The Farmer reads the notification and close the popup 6. The system shows the NotificationsPage
Exit conditions	The Farmer closes the NotificaitonsPage
Exceptions	

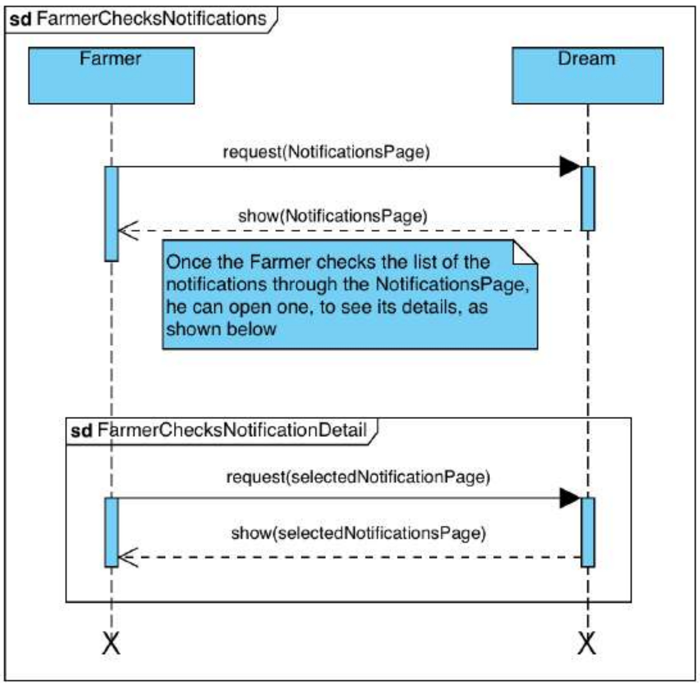

sd FarmerChecksNotifications
Farmer
Dream
request(NotificationsPage)
show(NotificationsPage)
Once the Farmer checks the list of the notifications through the NotificationsPage, he can open one, to see its details, as shown below
sd FarmerChecksNotificationDetail
request(selectedNotificationPage)
show(selectedNotificationsPage)

Figure 25: Sequence Diagram: Farmer checks notifications

Name	Policy Maker checks productions' information
Actor	Policy Maker
Entry conditions	The Policy Maker is already logged in the system The Policy Maker is in the PolicyMakerPage
Event flow	1. The Policy Maker selects the button "Fields" 2. The system shows the FieldsPage with all the fields cultivated by the Farmers 3. The Policy Maker selects a particular fields 4. The system shows a new page with all the Productions and the Productions' information of that field
Exit conditions	The system shows the FieldsPage and the Policy Maker can go back to the PolicyMakerPage
Exceptions	

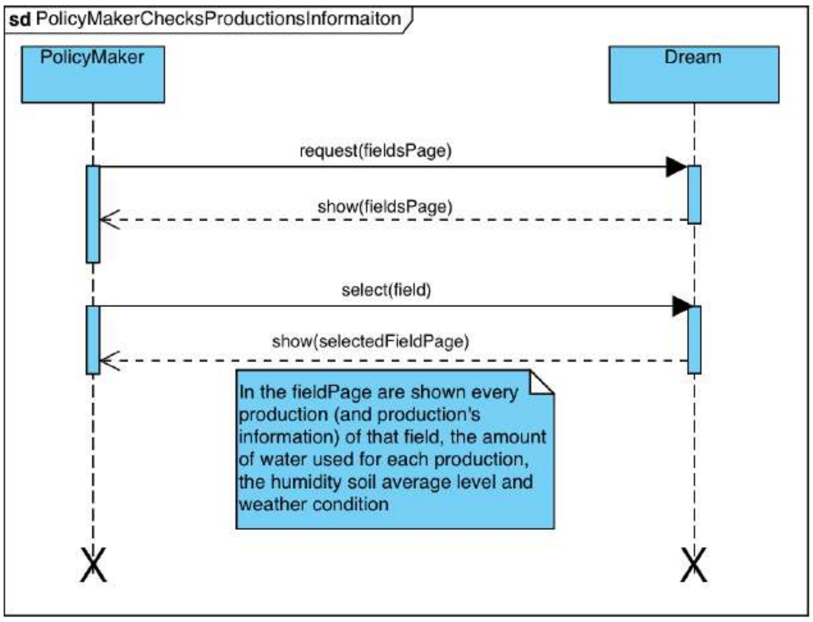

sd PolicyMakerChecksProductionsInformaiton
PolicyMaker
Dream
request(fieldsPage)
show(fieldsPage)
select(field)
show(selectedFieldPage)
In the fieldPage are shown every production (and production's information) of that field, the amount of water used for each production, the humidity soil average level and weather condition

Figure 26: Sequence Diagram: Policy Maker checks Productions

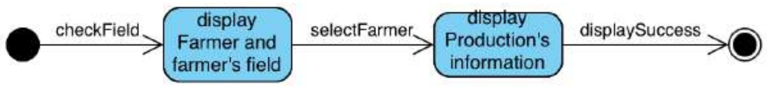

checkField
display Farmer and farmer's field
selectFarmer
display Production's information
displaySuccess

Figure 27: State Charts : Checks Productions

Name	Policy maker sees the best farmers
Actor	Policy Maker
Entry conditions	The Policy Maker is already logged in the system The Policy Maker is in the PolicyMakerPage
Event flow	1. The Policy Maker selects the button "Ranking" 2. The system shows the RankingPage and asks to select a product 3. The Policy Maker selects the particular product for which he/she wants to see the ranking 4. The system shows the table with the Farmers in descending order of score and their score
Exit conditions	The system shows the ranking successfully and the policy maker can go back to the PolicyMakerPage
Exceptions	

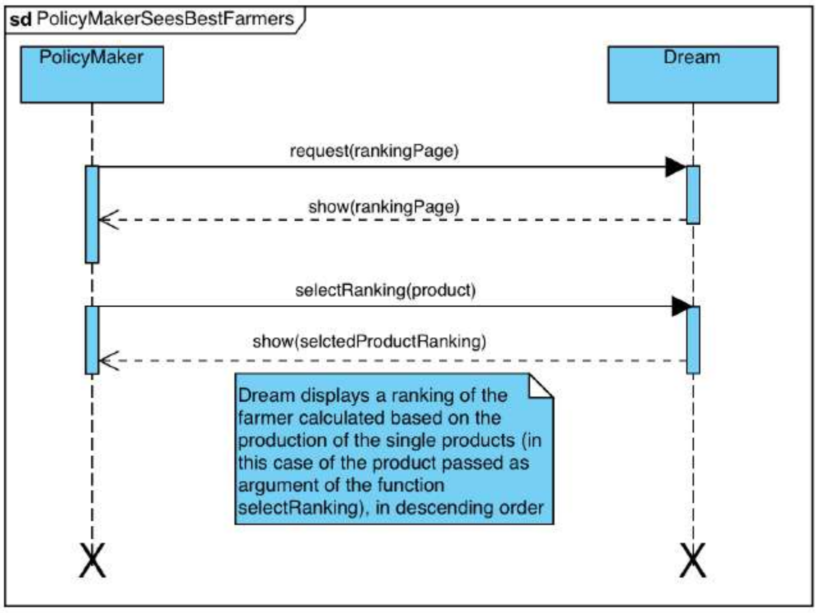

sd PolicyMakerSeesBestFarmers
PolicyMaker
Dream
request(rankingPage)
show(rankingPage)
selectRanking(product)
show(selctedProductRanking)
Dream displays a ranking of the farmer calculated based on the production of the single products (in this case of the product passed as argument of the function selectRanking), in descending order

Figure 28: Sequence Diagram: Policy Maker sees best farmers

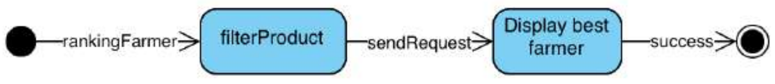


Figure 29: State Charts : Policy Maker sees best farmers

Name	Policy maker sees the worst farmers
Actor	Policy Maker
Entry conditions	The Policy Maker is already logged in the system The Policy Maker is in the PolicyMakerPage

Event flow	1. The Policy Maker selects the button "Ranking" 2. The system shows the RankingPage and asks to select a product 3. The Policy Maker selects the particular product for which he/she wants to see the ranking 4. The system shows the table with the Farmers in descending order of score and their score 5. The policy maker select the Order button to reverse the order 6. The system shows the table in growing order
Exit Conditions	The system shows the ranking successfully and the policy maker can go back to the PolicyMakerPage
Exceptions	

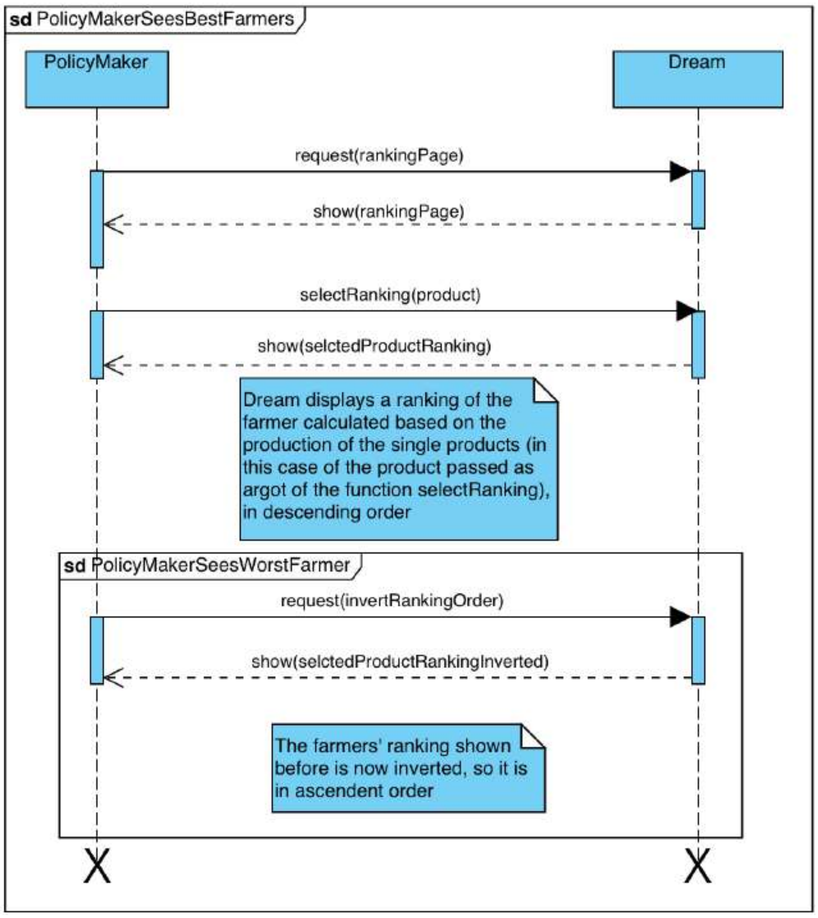

sd PolicyMakerSeesBestFarmers
PolicyMaker
Dream
request(rankingPage)
show(rankingPage)
selectRanking(product)
show(selctedProductRanking)
Dream displays a ranking of the farmer calculated based on the production of the single products (in this case of the product passed as argot of the function selectRanking), in descending order
sd PolicyMakerSeesWorstFarmer
request(invertRankingOrder)
show(selctedProductRankingInverted)
The farmers' ranking shown before is now inverted, so it is in ascendent order

Figure 30: Sequence Diagram: Policy Maker sees worst farmers

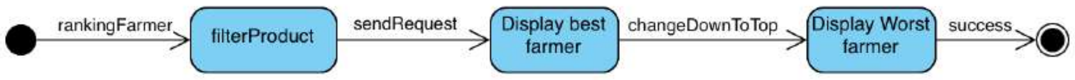


Figure 31: State Charts : Worst Farmer

Name	Policy maker sends agronomist for help farmer who are performing bad
Actor	Policy Maker
Entry conditions	The Policy Maker is already logged in the system The Policy Maker is in the PolicyMakerPage
Event flow	1. The Policy Maker selects the button "Ranking" 2. The system shows the RankingPage and asks to select a product 3. The Policy Maker selects the particular product for which he/she wants to see the ranking 4. The system shows the table with the Farmers in descending order of score and their score 5. The policy maker select the Order button to reverse the order 6. The system shows the table in growing order 7. The Policy Maker clicks on the button "Help" near the farmer to whom he/she wants to send the agronomist's help
Exit Conditions	The system sends the request for help successfully
Exceptions	

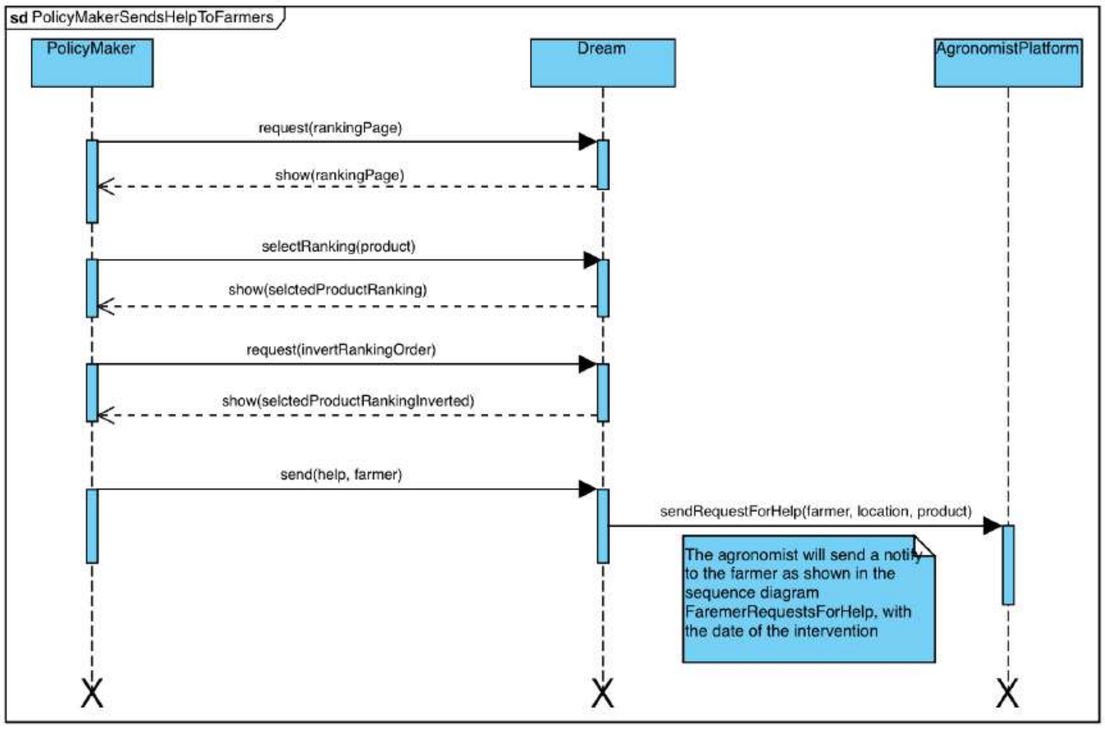

sd PolicyMakerSendsHelpToFarmers
PolicyMaker
Dream
AgronomistPlatform
request(rankingPage)
show(rankingPage)
selectRanking(product)
show(selctedProductRanking)
request(invertRankingOrder)
show(selctedProductRankingInverted)
send(help, farmer)
sendRequestForHelp(farmer, location, product)
The agronomist will send a notify to the farmer as shown in the sequence diagram FaremerRequestsForHelp, with the date of the intervention

Figure 32: Sequence Diagram: Policy Maker sees worst farmers

Name	Policy maker sends notifies of incentives to the farmers who perform well
Actor	Policy Maker
Entry conditions	The Policy Maker is already logged in the system The Policy Maker is in the PolicyMakerPage

Event flow	1. The Policy Maker selects the button "Ranking" 2. The system shows the RankingPage and asks to select a product 3. The Policy Maker selects the particular product for which he/she wants to see the ranking 4. The system shows the table with the Farmers in descending order of score and their score 5. The policy maker selects the button "Notify" near the best Farmer (or Farmers)
Exit conditions	The system sends the notification successfully
Exceptions	• The policy maker selects a Farmer that has already received the incentives • The system displays a pop up with " The Farmer has already received incentive for this product in this period"

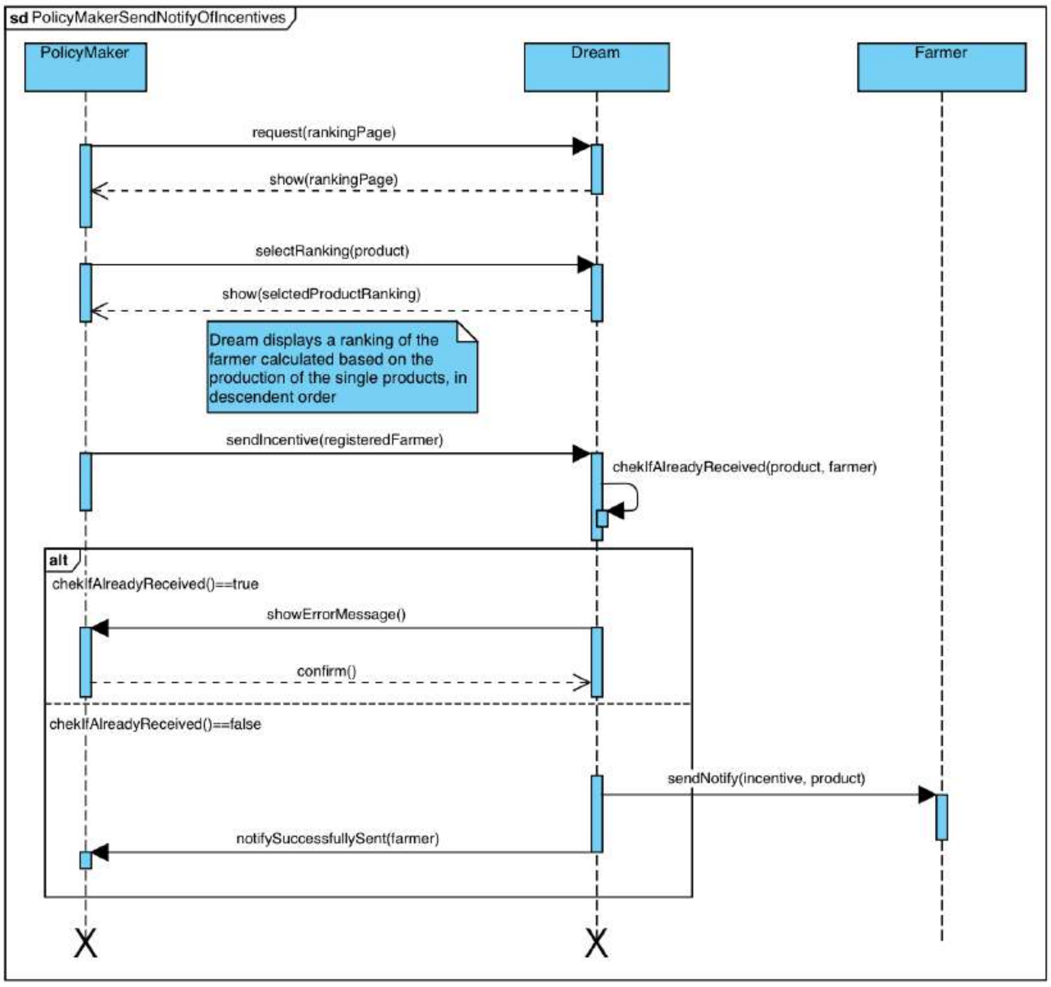

sd PolicyMakerSendNotifyOfIncentives
PolicyMaker
Dream
Farmer
request(rankingPage)
show(rankingPage)
selectRanking(product)
show(selctedProductRanking)
Dream displays a ranking of the farmer calculated based on the production of the single products, in descendent order
sendIncentive(registeredFarmer)
chekIfAlreadyReceived(product, farmer)
alt
chekIfAlreadyReceived()==true
showErrorMessage()
confirm()
chekIfAlreadyReceived()==false
sendNotify(incentive, product)
notifySuccessfullySent(farmer)

Figure 33: Sequence Diagram: Policy Maker sends notifications for incentives

Name	Policy maker sees the incentives history
Actor	Policy Maker The Policy Maker is already logged in the system The Policy Maker is in the PolicyMakerPage

Event flow	1. The Policy Maker selects the button "Ranking" 2. The system shows the RankingPage and asks to select a product 3. The Policy Maker selects the particular product for which he/she wants to see the ranking 4. The Policy Maker selects the button "IncentivesHistory" 5. The systems displays IncentivesHistoryPage with all the incentives which have been sent, the Farmers who received them, the date and the amount of the prize
Exit conditions	The system shows the incentives history successfully and the policy maker can go back to the PolicyMakerPage
Exceptions	

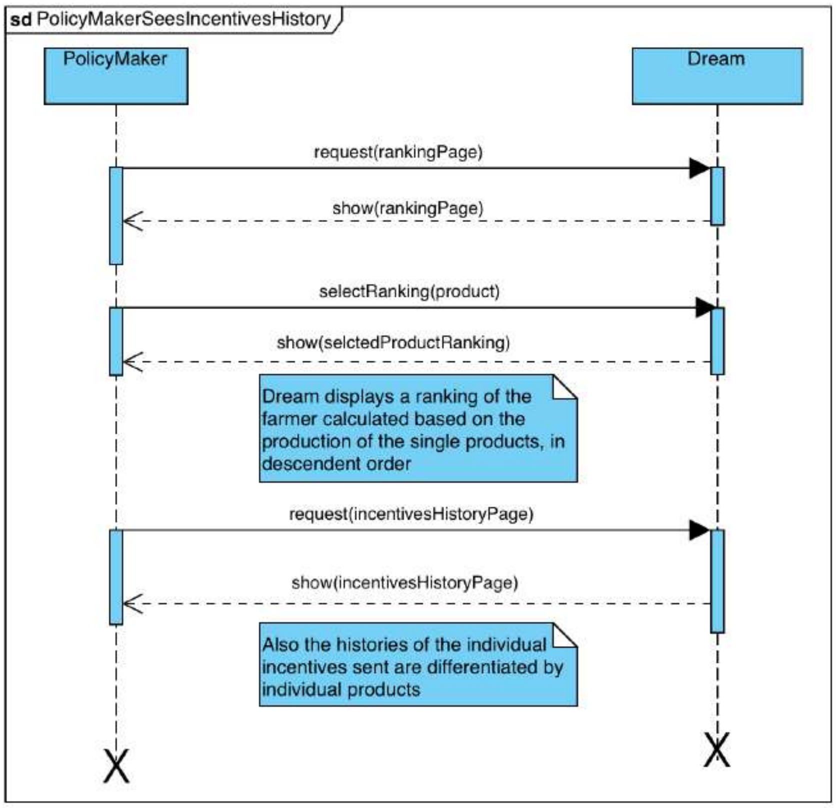

sd PolicyMakerSeesIncentivesHistory
PolicyMaker
Dream
request(rankingPage)
show(rankingPage)
selectRanking(product)
show(selctedProductRanking)
Dream displays a ranking of the farmer calculated based on the production of the single products, in descendent order
request(incentivesHistoryPage)
show(incentivesHistoryPage)
Also the histories of the individual incentives sent are differentiated by individual products

Figure 34: Sequence Diagram: Policy Maker sees incentives history

Name	Policy maker checks agronomists’ report
Actor	Policy Maker
Entry conditions	The Policy Maker is already logged in the system The Policy Maker is in the PolicyMakerPage
Event flow	1. The Policy Maker selects the button "AgronomistsReport" 2. The system displays the AgronomistsReportPage with a table with all the works of the agronomist and the Farmers that they had helped. 3. The Policy Maker selects a particular row to select a particular report 4. The system shows the popup SelectedReportDetail with the report details and the score of the helped Farmer before and after the intervention

Exit conditions	The system shows the agronomist's report successfully and the policy maker can go back to the PolicyMakerPage
Exceptions	

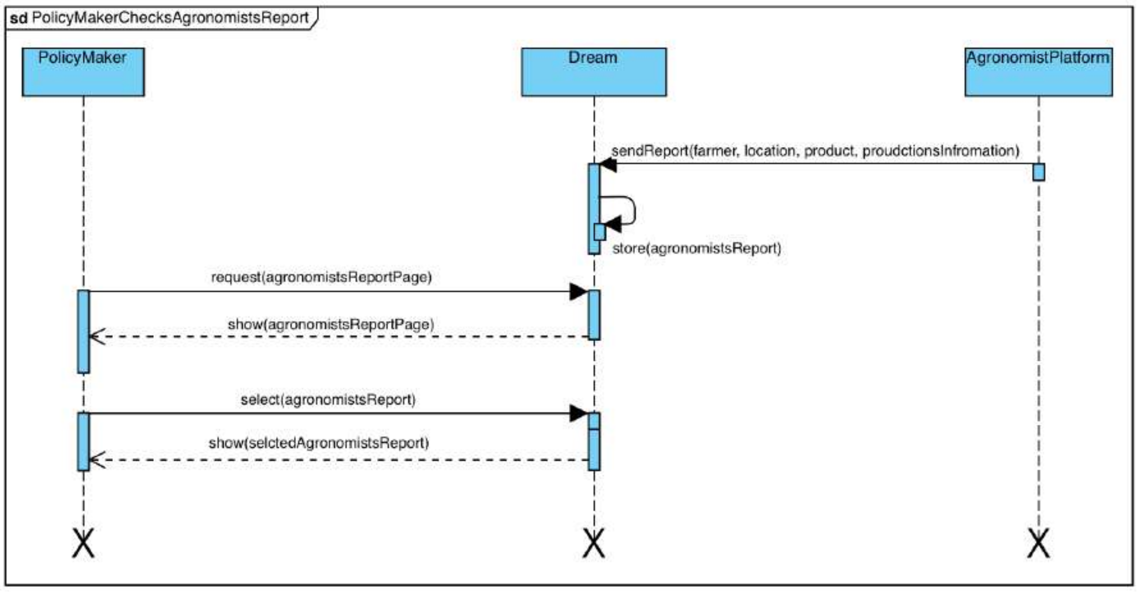

sd PolicyMakerChecksAgronomistsReport
PolicyMaker
Dream
AgronomistPlatform
sendReport(farmer, location, product, proudctionsInfromation)
store(agronomistsReport)
request(agronomistsReportPage)
show(agronomistsReportPage)
select(agronomistsReport)
show(selctedAgronomistsReport)

Figure 35: Sequence Diagram: Policy Maker checks agronomists' reports

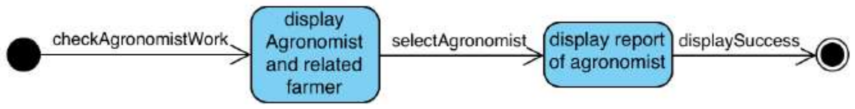

checkAgronomistWork
display Agronomist and related farmer
selectAgronomist
display report of agronomist
displaySuccess

Figure 36: State Charts : Policy Maker checks agronomists' reports

3.3 Functional Requirements

3.3.1 List of Requirements

Requirement	Description
R1	The system shall allow a Guest to register
R2	The system shall not accept an already Farmer email
R3	The system shall be allow the Farmer to insert his/her information (Farmer's information)
R6	The system shall send an email to the Farmer after he/she fills the form to confirm his/her registration
R7	The system shall store the data provided by the Farmer
R8	The system shall allow the Farmer to login to Dream by entering his/her credentials
R9	The system shall check if the password and the email are correct
R10	The system shall display the FarmerPage with the different services it offers to the Farmer

R11	The system shall be able to provide weather forecasts for a certain area and period
R12	The system shall allow the Farmer to insert his/her field's information
R13	The system shall store the data about Farmer's fields
R14	The system shall allow the Farmer to insert information of his/her Productions (Production's information)
R15	The system shall store the Production's Information
R16	The system shall allow the Farmer to see his/her Fields and the Fields' Information
R17	The system shall allow the Farmer to see his/her Productions and the Productions' Information
R18	The system shall allow the Farmer to ask for suggestions about a product for a certain Field
R19	The system shall allow the Farmer to ask for suggestions about a fertilizer for a certain Field
R20	The system shall provide suggestions about a product for a certain field
R21	The system shall provide suggestions about a fertilizer for a certain field
R22	The system shall allow the Farmer to ask for help for a certain Field to the agronomist
R23	The system shall allow the Farmer to check Notifications
R24	The system shall allow the Farmer to create a forum based on a certain topic and description
R25	The system shall allow the Farmer to join a forum created by another Farmer
R26	The system shall allow the Farmer to send messages in a forum
R27	The system shall allow the Farmer to read other Farmers' messages in a forum
R29	The system shall allow the policy maker to insert his/her secret code
R30	The system shall generates the login credentials of the policy maker
R31	The system shall store the login credentials of the policy maker
R32	The system shall send an email to the policy maker after he/she inserts the secret code with the credentials to login
R33	The system shall allow the policy maker to login by inserting his/her credentials
R34	The system shall display the PolicyMakerPage with the different services it offers to the policy maker
R35	The system shall allow the policky maker to check the Farmers and the fields they owns
R36	The system shall allow the policy maker to see Productions' Information for each field
R37	The system shall allow the policy maker to see the weather condition during the period of a certain Production
R38	The system shall allow the policy maker to see the amount of water used by a Farmer for a certain Production
R39	The system shall allow the policy maker to see the average humidity soil level of a field during the period of a certain Production
R40	The system shall allow the policy maker to see the Ranking of the best Farmers for a certain product
R41	The system shall allow the policy maker to see the Ranking of the worst Farmers for a certain product

R42	The system shall allow the policy maker to send notification that a Farmer will receive an incentive
R43	The system shall store the history of the given incentives
R44	The system shall allow the policy maker to see the history of the Farmer who received incentives
R45	The system shall not allow the policy maker to send notifications of incentives to Farmers who have already received one for the Production of that product
R46	The system shall allow the policy maker to send messages to the agronomists in order to help the worst Farmers
R47	The system shall allow the policy maker to check the report of the agronomist who helped Farmer who requested for help
R48	The system shall allow the policy maker to sends agronomist to the worst farmer in order to help them

3.3.2 Mapping on Goals

Goal	Domain assumption	Requirement
G1	D1, D2, D3, D5, D7	R1, R2, R3, R6, R7, R8, R9, R10, R12, R13, R16, R18, R20
G2	D1, D2, D3, D5, D8	R1, R2, R3, R6, R7, R8, R9, R10, R12, R13, R19, R16, R21
G3	D1, D2, D3, D4	R1, R2, R3, R6, R7, R8, R9, R10, R11, R12, R13, R16
G4	D1, D2, D3, D10, D11	R1, R2, R3, R6, R7, R8, R9, R10, R24, R25, 26, 27
G5	D1, D2, D3, D5, D9	R1, R2, R3, R6, R7, R8, R9, R10, R12, R13, R16, R22, R23
G6	D13, D14, D15, D16, D1, D2, D3, D4, D5, D6, D18, D19	R1, R29, R30, R31, R32, R33, R34, R2, R3, R6, R7, R8, R9, R10, R12, R13, R16, R14, R15, R17, R40, R42, R43, R44, R45, R23
G7	D13, D14, D15, D16, D1, D2, D3, D4, D5, D6, D18, D19	R1, R29, R30, R31, R32, R33, R34, R2, R3, R6, R7, R8, R9, R10, R12, R13, R16, R14, R15, R17, R41, R46, R23
G8	D13, D14, D15, D9, D17, D20	R1, R29, R30, R31, R32, R33, R34, R47
G9	D13, D14, D15, D1, D2,D3, D5, D6	R1, R29, R30, R31, R32, R33, R34, R2, R3, R6, R7, R8, R9, R10, R11, R12, R13, R14, R15, R16, R17, R35, R36, R37, R38, R39

4 Design Constraints

4.1 Standard compliance

- The system must manage the data retrieved from the the user in respect to privacy
- The system must manage the data received by the databases
- The data collected in the database must respect the international standard unit of measure.

4.2 Hardware limitations

Dream is a Web system so the system has to have

- 250 Mb of RMA
- 2 MB/s of internet

5 Software System Attributes

5.1 Reliability

DREAM should be available 24/7 in order to allow User to enter and interact with the system. It should be fault tolerance and a second database should be installed in order to not to lose date. However period of down can be accepted for the maintenance.

5.2 Availability

Since DREAM does not have an important value its availability can be also of 90%.

5.3 Security

In order to guarantee secure system, DREAM uses only encrypted communication protocols, for the user server communication the system uses only HTTPS. The sensible data are encrypted and password are not stored in the system. At the end, users are not allowed to enter in the system and to see sensible data.

Moreover, a Farmer cannot see the data of the other Farmers, only Policy Makers are allowed to check all the Fields' and Productions' information inserted by the Farmers.

The SecretCode is essential to verify the identity of the Policy Makers. Each secrect code is associeted with and Email. So for each PolicyMaker in the system there is an encrypted couple SecretCode/E-mail. When a PolicyMaker insert his/her SecretCode during the Registration, DREAM checks if it is in the database and find the corresponding Email. It generates a secure password and sends it to the Policy Maker's Email. Now the PolicyMaker can logs into the system using his/her email and the password the system provided him/her.

5.4 Maintainability

The system must to be organized in micro services in order to allow the system to be changed and upgraded in a simple way.

5.5 Portability

DREAM is developed as a Web system, so a device is sufficient to connect to the internet.

6 Formal Analysis Using Alloy

In this section we use Alloy to simulate specifications and perform property verification of DREAM. The focus is on some static constrains such as :

- Each User have an Unique Email and one Password
- Each PolicyMaker have one Unique Secret Code to enter in the system
- Each Field is different from the other fields so they cannot have the same position, the same farmer or the same productions
- Each production is unique, so two Field cannot have the same productions. Moreover a Production mast have the planted date greater than collected date.
- Each Forum is associated to a topic and a messages that must be unique. The messages are associated to a Farmer in order to identify which farmer has joined the forum
- Each Ranking represents a ranking of single product so all the productions in one ranking have the same product.
- The suggestions are linked to a Field, fertilizers and a products.
- Each request for help is linked to a Farmer.
- Each AgronomistReport is linked to a Farmer
- Weather is unique and they have a date and a location

The Agronomist is not considered because he uses an other platform to connect with DREAM

6.1 Model

```
open util/boolean
open util/integer

-------------------
-- Signatures
-------------------

sig Email, Password {}
sig Name {}
sig SecretCode {}
sig Sensor{}
sig Product{}
sig Score{}
sig Fertilizer{}
sig Weather{
        location : one Location,
        date : one Date}
abstract sig User{
        email: one Email,
        password: one Password
}

sig Farmer extends User{
        fields: some Field,
        suggestions : some Suggestion,
        notifications : some Notify,
        request : some RequestForHelp
}

sig PolicyMaker extends User{
        secretCode: one SecretCode,
}

sig Location{
        latitude: one Int,
        longitude: one Int
}

sig Field{
        location: one Location,
        productions : some Production
}
sig Production{
        collectedDate : one Date,
        plantedDate : one Date,
        sensors : some Sensor,
        product : one Product,
        score : one Score,
        fertilizers : some Fertilizer
}

sig AgronomistReport{
        farmer : one Farmer
}


sig Ranking{
        product : one Product,
        productions : some Production
}

sig Date{
        date : one Int
} {date > 0}

sig Topic{
        farmer: one Farmer
}
sig Text {}
sig Forum{
        topic : one Topic,
        descritpion : one Text,
        messages :  some Message
}

sig Message{
        farmer : one Farmer,
        text : one Text
}

sig Suggestion{
        field: one Field,
        fertilizers : some Fertilizer,
        products : some Product
}

sig Notify{
        from : one PolicyMaker}

sig RequestForHelp{
    field: one Field,
    text : one Text}
---------------
-- Facts
----------------

//Cannot exist an email without a user
fact noUsernoEmail{
        all e: Email | some u: User |
                e in u.email
}

//Cannot exist a password without a user
```

```
fact noUsernoPassword{
        all p: Password | some u: User |
                p in u.password
}

//Two users cannot have the same email
fact allUniqueUsers{
        no disj u1, u2 : User | u1. email = u2.email
}
//Cannot exist a SecretCode without a PolicyMaker
fact noPolicyMakerSecretCode{
        all s: SecretCode | some p : PolicyMaker |
                s in p.secretCode
}

//Two PolicyMaker cannot have the same secretCode
fact allUniqueSecretCode{
        no disj p1, p2 : PolicyMaker | p1.secretCode = p2.secretCode
}

//Cannot exist a Field without a Farmer
fact noFarmerNoField{
        all f:Field | some fa:Farmer |
                f in fa.fields
}

//Cannot exist two farmer with the same Field
fact allUniqueField{
        all f : Field |
                no disj fa1, fa2 : Farmer |
                        f in fa1.fields and f in fa2.fields
}
//Cannot exist a location without a Field or a Weather
fact noLocationnNoField{
        all l: Location| (some f: Field | l in f.location) or (some w : Weather | l in w.
            ↪ location)
}

//Two field cannot have the same position
fact allUniqueLocation{
        no disj f1, f2 : Field | f1.location = f2.location

}
//Cannot exist a Production without a Field
fact noFieldNoProduction{
        all p: Production | some f:Field |
                p in f.productions
}

//Cannot exist two field with the same Production
fact allUniqueProduction{
        all p : Production |
        no disj f1,f2 : Field |
         p in f1.productions and p in f2.productions
}

//Cannot exist a Sensor without a Production
fact noProductionNoSensor{
        all s: Sensor | some p:Production |
                s in p.sensors
}

//Cannot exist two Production with the same Sensor
fact allUniqueSensor{
        all s: Sensor |
        no disj p1, p2 : Production |
                s in p1.sensors and s in p2.sensors
}

//Cannot exist a Topic without a Forum and a Farmer
fact noTopicNoForum{
        all t : Topic | some f : Forum |
                t in f.topic
}
//Two Forum cannot have the same Topic
fact allUniqueTopic{
        all t: Topic |
        no disj f1, f2 : Forum |
                t in f1.topic and t in f2.topic
}

//Cannot exist Message without a Forum
fact noForumnoMessages{
        all m : Message | some f: Forum |
                m in f.messages
}
//Cannot exist the same Message in 2 different Forum
fact allUniqueMessages{
        all m : Message |
                no disj f1, f2 : Forum | m in f1.messages and m in f2.messages
                    ↪
}
//Cannot exist 2 Weather with the same Date and Location
fact allUniqueWeather{
     no disj w1,w2: Weather | w1.location = w2.location and w1.date = w2.date
}

//Cannot exist a Date without a weather or a production
fact noOnlyDate{
        (all d:Date | some w: Weather | d in w.date) or (all d:Date | some p: Production
        | d in p.collectedDate and d in p.plantedDate)
}

//For each Production CollectedDate must be greater than PlantedDate
fact dateOneProduction {
        (all p : Production | p.collectedDate.date> p.plantedDate.date)
}

//A Production cannot be in more than one Ranking
fact prodcutionsInOneRanking{
        all p : Production |
                no disj r1, r2 : Ranking | p in r1.productions and p in r2.productions
}

//It exists only one Ranking for Product
fact noSameProductInRanking{
        all p : Product |
                no disj r1, r2 : Ranking | p in r1.product and p in r2.product }

//Cannot exist a Score without a Production
fact noProductionNoScore{
        all s : Score | some p : Production |
                s in p.score
}

//All the Productions must be in a Ranking
fact allProductionsInRanking{
        all p : Production | one r: Ranking | p in r.productions
}

//All the Productions in a Ranking must have the same product which must be the same of
    ↪ the Ranking
fact rankingProducts{
        all r:Ranking| all p :r.productions | p.product = r.product
}


//Cannot exist a Fertilizer without a Suggestion or a Production
fact noProductionSuggestioNoFertilizer{
        (all f:Fertilizer | some p: Production | f in p.fertilizers) or (all f:Fertilizer
            ↪  |
        some s: Suggestion | f in s.fertilizers)
}
// Cannot exist a suggestion without a Farmer
```

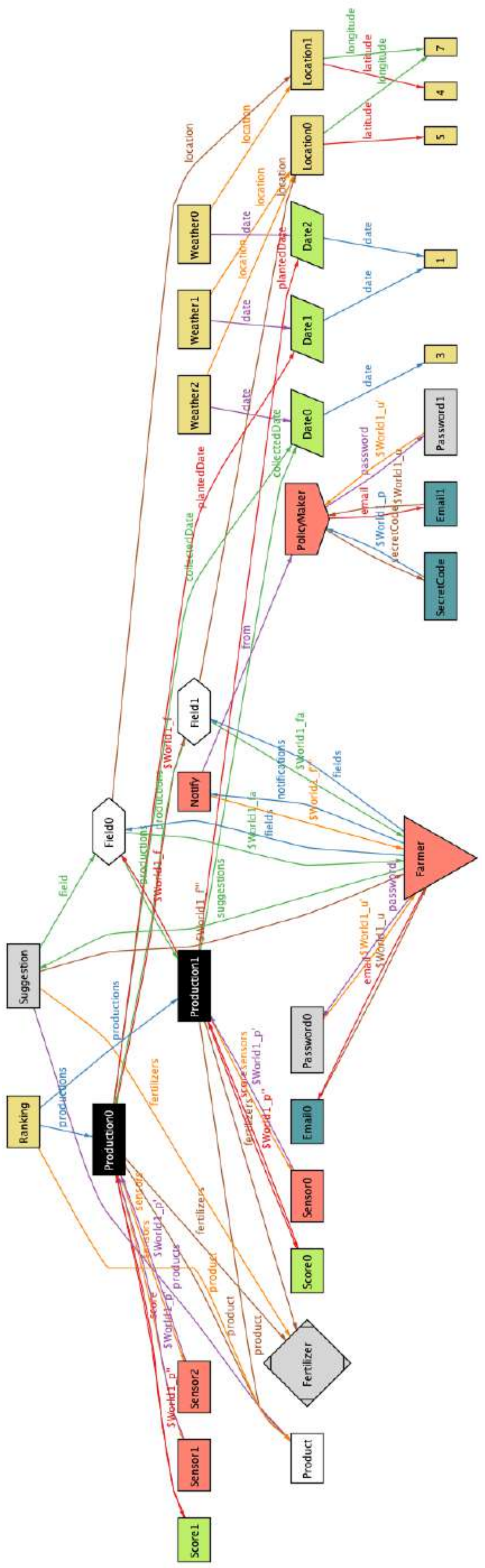
```
fact noFarmerNoSuggestion{
        all s : Suggestion | some f : Farmer |
                s in f.suggestions
}
//Cannot exist a Notification without a Farmer
fact noFarmerNoNotification{
        all n : Notify | some f : Farmer |
                n in f.notifications
}

//Cannot exist a Date without a Weather or a Production
fact noOnlyDate{
        (all d:Date | some w: Weather | d in w.date) or
        (all d:Date | some p: Production | d in p.collectedDate and d in p.plantedDate)
}
//Cannot exist a request for help without a farmer{
    all r: RequestForHelp | some f: Farmer |
        r in f.request}
---------------
-- Assertions and Checks
----------------

assert UniqueEmail{
        no disj u1, u2 : User | u1.email = u2.email
}
check UniqueEmail for 5
//Each field is own by a singol farmer and 2 farmer don't have the same field


assert SingleFarmerforField{
        no disj f1, f2 : Farmer | some fi:  Field | fi in f1.fields and fi in f2.fields
}
check SingleFarmerforField for 4


//Each Sensor is own by a Production
assert uniqueSensor{
        no disj p1, p2 : Production | some s : Sensor | s in p1.sensors and s in p2.
            ↪ sensors
}
check uniqueSensor for 3

//Unique Production in each Field
assert uniqueProduction{
        all p : Production |
        no disj f1,f2 : Field |
         p in f1.productions and p in f2.productions
}
check uniqueProduction for 2
//Ranking have different productions
assert Ranking{
        no disj r1, r2 : Ranking | some p: Production | p in r1.productions and p in  r2.
            ↪ productions
}
check Ranking for 4
```

6.2 Result

As is shown in the images the model is consistent.

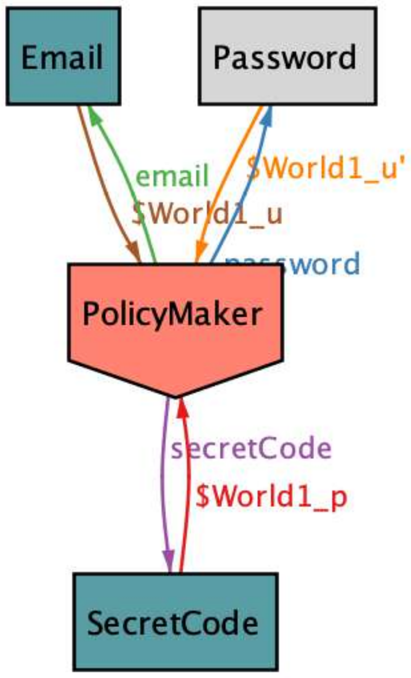
```
pred World1{
#Farmer = 1
#PolicyMaler = 1
#Field = 2}
```

The predicate World1 is consistent and the result is shown in the first figure. There are a Farmer and a PolicyMaker that have different email, two Field that have different Production. There is also the ranking which is linked to a single product,and two different productions are linked to the ranking and they have the same product. Lastly there is also the notify that links the PolicyMaker with the Farmer and the sensor that are linked to different productions.


Location1
Location0
Weather0
Weather1
Weather2
Date0
Date1
Date2
1
3
4
5
7
PolicyMaker
Password1
Email1
SecretCode
Field0
Field1
Notify
Farmer
Suggestion
Ranking
Production0
Production1
Password0
Email0
Sensor0
Sensor1
Sensor2
Score0
Score1
Fertilizer
Product
location
latitude
longitude
date
plantedDate
collectedDate
from
notifications
fields
field
suggestions
productions
fertilizers
sensors
score
product
products
email
password
secretCode

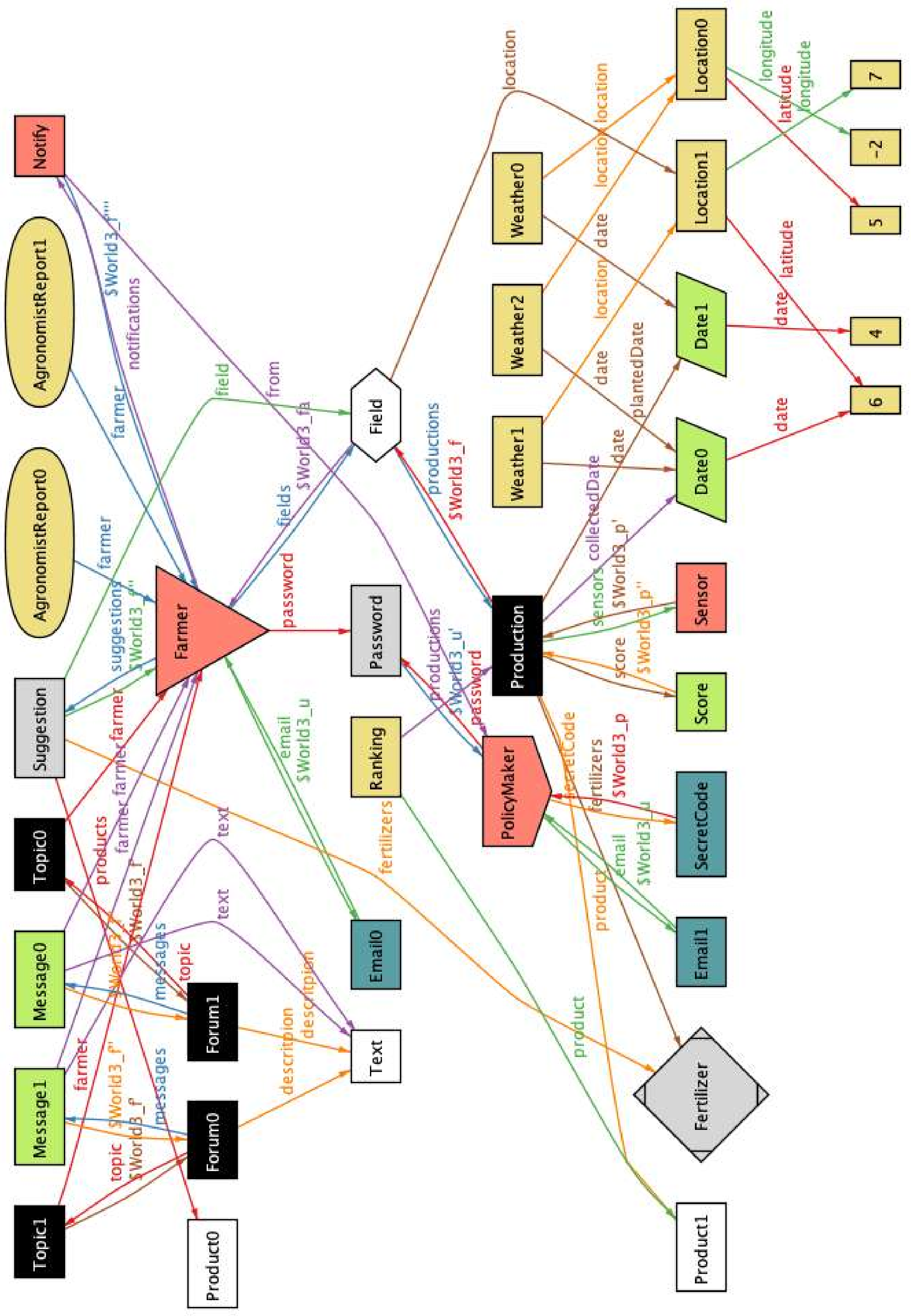
```
pred World2{
#Farmer = 0
#PolicyMaker = 1}
```

The predicate World2 is consistent and shown the PolicyMaker

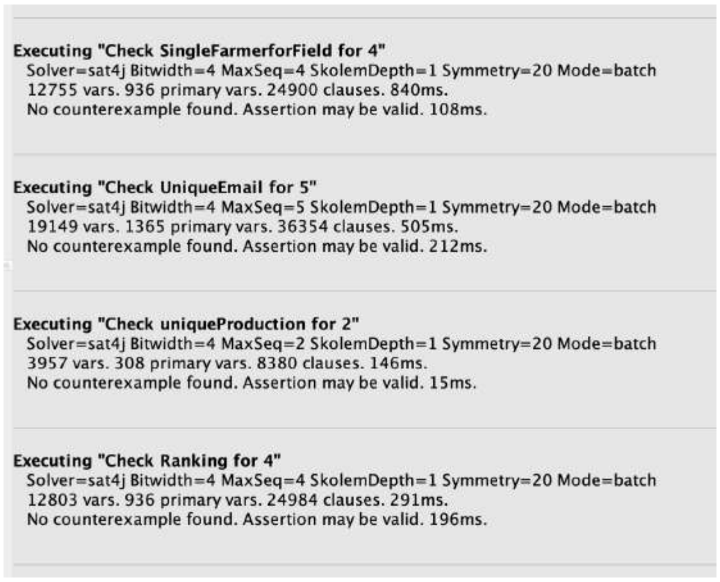


The predicate World3 is consistent, there are shown 2 Agronomist Report linked to the farmer and 2 Forum with respective messages.

```
pred World3{
#AgronomistReport =2
#Forum = 2}
```


Notify
AgronomistReport1
AgronomistReport0
Suggestion
Topic0
Message0
Message1
Topic1
Forum1
Forum0
Product0
Farmer
Field
Password
Ranking
Email0
Text
Weather0
Weather2
Weather1
Production
PolicyMaker
Location0
Location1
Date1
Date0
Sensor
Score
SecretCode
Email1
Fertilizer
Product1
7
-2
5
4
6
farmer
suggestions
notifications
field
from
fields
password
email
productions
location
longitude
latitude
date
plantedDate
collectedDate
sensors
score
secretCode
fertilizers
product
products
topic
messages
text
description

The model is consistent and no counterexamples are found



```
Executing "Check SingleFarmerforField for 4"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20 Mode=batch
  12755 vars. 936 primary vars. 24900 clauses. 840ms.
  No counterexample found. Assertion may be valid. 108ms.

Executing "Check UniqueEmail for 5"
  Solver=sat4j Bitwidth=4 MaxSeq=5 SkolemDepth=1 Symmetry=20 Mode=batch
  19149 vars. 1365 primary vars. 36354 clauses. 505ms.
  No counterexample found. Assertion may be valid. 212ms.

Executing "Check uniqueProduction for 2"
  Solver=sat4j Bitwidth=4 MaxSeq=2 SkolemDepth=1 Symmetry=20 Mode=batch
  3957 vars. 308 primary vars. 8380 clauses. 146ms.
  No counterexample found. Assertion may be valid. 15ms.

Executing "Check Ranking for 4"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20 Mode=batch
  12803 vars. 936 primary vars. 24984 clauses. 291ms.
  No counterexample found. Assertion may be valid. 196ms.
```

Figure 37: Results of Check

7 Effort Spent

Member: Alessandro Pindozzi

Task	Hours
General Reasoning	10
Introduction	7
Product Perspective	2.5
Product Functions	3
User Characteristics	1.5
Assumption, dependencies and constraint	3
External Interface Requirements	3
Functional Requirements: • User cases Diagrams • Use cases • Sequence Diagrams • Statecharts Diagrams	7: • 0 • 2.5 • 4.5 • 0
Functional Requirements	7
Design Constraints	1
Software System Attributes	2
Alloy	7
Document Organization	4

Member: Diletta Quarticelli

Task	Hours
General Reasoning	8
Introduction	7
Product Perspective	3.5
Product Functions	3
User Characteristics	1
Assumption, dependencies and constraint	3.5
External Interface Requirements	3
Functional Requirements: • User cases Diagrams • Use cases • Sequence Diagrams • Statecharts Diagrams	9: • 3 • 6 • 0 • 2
Functional Requirements	5
Design Constraints	3
Software System Attributes	3
Alloy	8
Document Organization	3