POLITECNICO DI MILANO

SOFTWARE ENGINEERING II PROJECT

DREAM

Requirements Analysis and Specifications Document

Authors:
Mattia SABELLA
Salvatore SCOZZARI
Francesca DE DONATO

Professor:
Damian TAMBURRI

December 23, 2021

version 3.0

Contents

List of Figures

List of Tables

1 Introduction

1.1 Purpose

This document describes in detail the system in terms of functional and non-functional requirements, includes exhaustive descriptions about typical use cases and is used as a contractual basis between the customer and the developer. This document is addressed to the developers of the S2B as well as the actors that will take part in the system. Hence this document is the result of the requirements elicitation and the analysis activities paired with a specific description aimed to precise the behavior of the system.

1.2 Scope

The aim of the system is to provide an an anticipatory governance model for food systems, helping to strengthen the data-driven policy making process of a Country or State/Region. The system involves multiple actors, from policy makers to farmers and agronomists and allows them to have much information and data regarding many of their areas of interest all gathered and easily disposable.

The system will help:

- • the farmers, to improve their production, by an efficient exchange of suggestions with each other and the possibility to interact with a dedicated agronomist, who is responsible of their area of interest.
- • the agronomists, to manage efficiently their area of interest, by visualizing data concerning weather forecasts and farmers' performance, helping them to keep their daily plan of visits to the farmers updated
- • the State's policy makers, to keep track of the farmers' performance, in order to give special incentives to the best ones and to help the ones who are performing particularly bad, and to evaluate the agronomist's steering initiatives.

Now that we have a first description of what DREAMS aims to achieve, we can have a look at the goals that have been chosen in order to accomplish these functionalities.

1.2.1 Goals

G1 Allow Farmers to monitor their production and statistics

G2 Allow Farmers to specify any problem and update about their production

G3 Allow Farmers to interact with other Farmers

G4 Allow Farmers to interact with Agronomists and viceversa

G5 Allow Agronomists to organize and simplify their work according to their area of interest

G6 Allow Agronomists to visualize best/worst performing farmers in their area of interest

G7 Allow Telangana's Policy Makers to visualize best/worst performing farmers

G8 Allow Telangana's Policy Makers to analyze agronomists' work

1.2.2 World and Shared Phenomena

WP1 Farmer decides to improve his production of a particular crop field.

- **SP1A** Farmer defines in the system data about his production field.
- **SP1B** Farmer visualizes data relevant to his production and location.

WP2 Farmer installs useful sensors on a field he owns.

- **SP2A** Farmer defines in the system every sensor he installed.

WP3 Sensors gather data from the field.

- **SP3A** Sensors send data through a communication infrastructure to the system.

WP4 Farmer faces a problem or notices peculiar aspects of his production.

- **SP4A** Farmer inserts in the system updates about his problem and peculiar aspects he faced.

WP5 Farmer takes a picture of the field.

- **SP5A** Farmer inserts in the system the photo he shot.

WP6 Farmer wants to talk with other farmers.

- **SP6A** Farmer creates/participates to discussion forums with other farmers.
- **SP6B** Farmer reports off topics in the topic of the discussion forum.
- **SP6C** The system received more than a certain number of reports for a post.

- **SP6D** The system removes the post.

WP7 Farmer wants to share his knoledge about a crop type.

- **SP7A** Farmer inserts in the system suggestions about a crop type.
- **SP7B** The system forwards the suggestions to all farmer that own the targeted crop.
- **SP7C** Farmer receives a notification about a new suggestion that has been received.

WP8 The field has completed his production.

WP9 Farmer harvests his production field.

- **SP9A** Farmer declares in the system that a production field has been harvested.
- **SP9B** The system builds a performance index for the farmer.
- **SP9C** The system builds a performance index for the Agronomist that helped the farmer.

WP10 Farmer wants to contact privately another Farmer or an Agronomist.

- **SP10A** Farmer sends an help request through the system to a Farmer or an Agronomist.
- **SP10B** Farmer or Agronomist replies.
- **SP10C** Farmer or Agronomist blacklists the contact that sent the help request.
- **SP10D** Farmer or Agronomist removes a contact from his blacklist.

WP11 Farmer has not yet been visited twice by an Agronomist during the current year.

- **SP11A** The system notifies Agronomists whose area of interest covers the above mentioned Farmer.

WP12 Agronomist reviews farmers' work according to his area of interest.

- **SP12A** Agronomist sees farmers in the system.
- **SP12B** Agronomist sees in the system production data of a Farmer.
- **SP12C** Agronomist publishes a suggestion for the Farmer.

WP13 Agronomist wants to memorize which farms he has to visit for the next day.

- **SP13A** The system builds a new daily plan for Agronomist.
- **SP13B** The Agronomist inserts all farms to be visited in the daily plan.

WP14 Agronomists visited all farms he has to visit according to his plan.

- **SP14A** Agronomists confirms his daily plan.

WP15 Telangana's Policy Maker decides to monitor farmers' performances.

SP15A Telangana's Policy Maker asks the system to display farmers and their performances.

WP16 Telangana's Policy Maker decides to monitor agronomist strategies.

SP16A Telangana's Policy Maker asks the system to display steering initiatives carried out by agronomists with the help of good farmers.

WP17 Telangana's government releases incentives for good farmers .

1.3 Glossary

1.3.1 Definitions

- **Unauthenticated User:** An user that is not authenticated; he is not allowed to use all the DREAM functions except for Login and Registration.
- **User:** An authenticated user; he can use the DREAM functions relevant for his role.
- **Farmer:** An user that owns a "Farmer" account; he can access to the DREAM functions useful for Farmers.
- **Agronomist:** An user that owns an "Agronomist" account; he can access to the DREAM functions useful for Agronomists.
- **Telangana's Policy Maker:** An user that owns a "Telangana's Policy Maker" account; he can access to the DREAM functions useful for Telangana's Policy Makers
- **Production Field:** A virtualized element that corresponds to an existing production field for a Farmer. It is mainly defined by its crop and the owner of the field..
- **Performance Index (Farmers):** A numerical index useful for ranking farmers whose formulation is based on the average of the combination of three parameters per timeline: number of crops planted for production field, number of crops harvested for production field, weather conditions during the production field cultivation. A possible formulation is: E[cropsPlanted(productionField, timeOfPlanting)/cropsHarvested(productionField, timeOfHarvesting) * weatherAverageIndexConditions(timeOfHarvesting-timeOfPlanting)]. Where E means "Expected Value" based on a two parameters function and weatherAverageIndexConditions is an index quantifying how bad/good weather conditions were in the cultivation temporal window.
- **Performance Index (Agronomists):** A numerical index useful for ranking agronomists whose formulation is based on the average of the combination of two parameters per timeline: number of Farmer visited, improvement of Farmers.

- **Geographical Location:** A geographical location in which at most one Farmer can work/owns production fields.
- **Area of Interest:** A geographical area in which one or more Agronomists can work. Each area groups different geographical locations; each Agronomist is associated with one and only one area of interest.
- **Topic:** A virtual sub-place of discussion consiting of a particular argument.
- **Comment:** A virtual opinion of a Farmer that populates a topic.
- **Off-Topic:** The act of talking of useless things that do not fit into the bounds of the main context of the topic.
- **Help Request:** A request for help sent by a Farmer to an Agronomist or a Farmer.
- **Help Reply:** A reply to an help request sent by an Agronomist or a Farmer to a Farmer.
- **Blacklist:** A virtual list owned by a contact. The list is populated by blocked contacts (contacts that can't send messages to the owner of the list).
- **Message:** Help Request or Help Reply.
- **Inbox:** A virtual inbox containing messages for a contact.
- **Daily Plan:**A virtual memo-like plan owned by an Agronomist. It contains farm(er)s to visit and it is created day by day.
- **Report:** A virtual reporting to a potentially offending post or topic.
- **Private Suggestion:** A suggestion written by an Agronomist for a Farmer.
- **Crop Based Suggestion/Farmer Suggestion:** A suggestion written by a Farmer. It is based on a particular crop type..
- **Personalized Suggestion :** Synonim for Crop Based Suggestion.
- **Farmer Statistics:** Statistics built by the system for the Farmer. They are built day by day and they involve several aspects like planted crops in a particular day and harvested crops in another day. They are represented through cartesian graphs with timeline on x axis.
- **Agronomist Statistics** Statistics built by the system for the Agronomist. They are built day by day and they involve several aspects like visted farm(er)s in a particular day, improvement of farmers detected in another day. They are represented through cartesian graphs with timeline on x axis.

1.3.2 Acronyms

- **RASD:** Requirement Analysis and Specification Document
- **IoT:** Internet of Things (devices)

1.3.3 Abbreviations

- **TPM:** Telangana's Policy Maker
- **DA:** Domain assumption
- **G:** goal
- **WP:** World Phenomena
- **SP:** Shared Phenomena
- **R:** Requirement

1.4 Document Structure

The document is structured in a double linked way in order to provide an easier and quicker navigation in particular for the parts where several abbreviations are used and the entire text would not fit in the layout. The aim is to give a description that interconnects the most interesting parts of the document that are also related in a theoretical point of view: **World and Machine**, **Goals and Requirements** and **Use Cases.**

Moreover the document is structured as now briefly described:

1. **Introduction:** gives a first description of the problem and describes the purpose of the DREAM system. Goals are also highlighted to enforce the previous shallow description; the section ends with the glossary.
2. **Overall Description:** starts with the product perspective where first the system is highlighted from the outside and then from the inside with a high-level description of its structure. State diagrams are then used to clarify the behavior of the most critical objects identified in the modeling process and then product functions are ready to be precisely described. The section ends with the identification of the important phenomena for the problem that are now clearly described with the World and Machine paradigm.
3. **Specific Requirements:** in this section, requirements are precisely described starting with the ones of the interfaces that DREAM uses to provide its services to the external world. Functional requirements, where the satisfaction of the goals is proved thanks to the requirements, and the domain assumptions previously defined, are the core of this section. Use cases are also important in particular to highlight their strict relation with the requirements also highlighted with the traceability matrix.
4. **Formal Analysis Using Alloy:** includes the model that is described formally thanks to the alloy language. This section highlights the most critical aspects of the entire problem and proves their satisfaction in specific worlds generated for this purpose.
5. **Effort Spent:** this section has been used to keep track of the hours spent to complete this document. The first table defines the hours spent together while taking the most important decisions, the seconds instead contain the individual hours.

6. **References:** includes all the references used to define the document.

2 Overall Description

2.1 Product Perspective

Thanks to the general introduction and the scope definition from the previous sections, we are now able to look at our system first from the outside and then from the inside. To deal with this description we are going to see the external interfaces the system has to interact with and then the definition of the model in order to have a feasible structure with them; at the end of the section, **state diagrams** are used to emphasize the dynamic behavior of the most critical classes identified for the model.

2.1.1 Model Structure

The static analysis now continues to define the internal structure of the system, in particular with a high-level class diagram (Figure 1) that considers the most important objects and their relations in order to achieve the functionalities described by the **goals**.

The main objects in the UML class diagram are:

- **User:** the system has to track three types of users (e.g Farmers, Agronomists, TPM). This distinction is essential to define the three types of actors that interact with the system. These users have attributes in common, even though they have distinct roles, different registration process and home pages for example. This class keeps track of information related to the login and search process (name, surname, email, password, avatar).
- **Farmer:** identifies a farmer with all the data he provides in its registration. This class represents a client of the application and through that it is possible to insert data production fields, visualize them, send requests to other users, interact in forum or send suggestions. Farmers can be classified according to certain parameters.
- **Agronomist:** identifies an agronomist with all the data provides in its registration. This class needs a license that identifies uniquely the user who is in particular an agronomist. Also Agronomists are clients for application that have the role of answering the requests of the farmers, visualizing their statistics, sending suggestions and visiting farmers according to a specific daily plan.
- **TPM:** identifies a Telangana Policy Makers with all the data provides in its registration. This class needs a license thtat identifies uniquely the user who is in particular a TPM. Also TPMs are clients of the application but they visualize both agronomists and farmers statistics to monitor their work and the progession of the production.
- **Production Field:** identifies all the data related to the cultivation of a certain crop and is linked only with the farmers. It refers only to a single production field, so only one crop. The statistics of the farmers are built upon these data, for example the number of planted seeds is crucial to build the performance index of the farmers and indirectly to build the

performance index of the agronomist. The data stored in this class, are visible in the pages of the farmers and only by the agronomists related to the same geographical area.

- **Crop:** identifies a specific known crop. This class is useful to define exactly a production field, but it exists to define suggestions, because every suggestion is linked with a crop. In the next sessions it is possible to see how the suggestions are visualizable only related to a certain crop
- **Farmer Statistics:** stores all the data computed from the production fields class (e.g performance index). Every time a production field cycle ends, the amount harvested is inserted by the farmer and only then, the statistics are computed and stored. It is essential to represent a chart that visualizes the progression or the regression of the production
- **IoT Sensor:** all production fields have a set of IoT sensors that monitor and send measurements as updates to the system in order to give more information to the state and development of the production field. Every Iot sensor has an IP to be identified.
- **Discussion Forum:** this is a singletone class because the system has a single forum that is a collection of topics and comments.
- **Topic:** identifies every topic published by a farmer and it is described by a title and an author. A topic is a collection of comments.
- **Comment:** represents each comment pubblished by a farmer for a certain topic. It has the description, so the comment itself and the commenter. A comment can be written only by a farmer
- **Request/Response Message:** identifies the interaction between agronomist and farmers. A farmer sends a message request to an agronomist and the agronomist can send a reply to that farmer. All this kind of interactions are stored in this class.
- **Suggestion:** identifies another type of direct interaction between farmers and agronomist. There exists two types of suggestions that are related to other two classes (e.g *crop suggestion*, *private suggestion*). The crop suggestion is related to only one crop, so in this way every user that has this type of cultivation, can see these suggestions. The private suggestion is written by an agronomist and addressed only to one farmer.
- **Geographical location:** identifies the precise location of the farm for each farmer. This is essential to pair a farmer in a certain *area of interest*. Every area of interest is managed by the agronomist belonging to that *area of interest*. A location is identified with an address and then translated into geographical coordinates.
- **Area of interest:** identifies the geographical area where the agronomist has to work. The area is composed by a large set of locations so the agronomist has to take care of a group of farmers. Every area has its name.

- **Agronomist Statistics:** represents the performance of each agronomist because they are evaluated and supervised by TPMs. These performances are based on the sum of the performance index of each farmer related, on the visits completed and on the suggestions sent.
- **Daily Plan:** identifies the collection of daily plans for each agronomist. This class stores each visit to the farmers with the date. The date is useful to specify which is the daily plan because for each day has one istance of daily plan.
- **Telangana's weather data** it is the collection of data related to the weather forecasts. Each weather forecast is personalized for each geographical location. This class is importante because the weather forecasts must be visible for the farmers and for the agronomists.

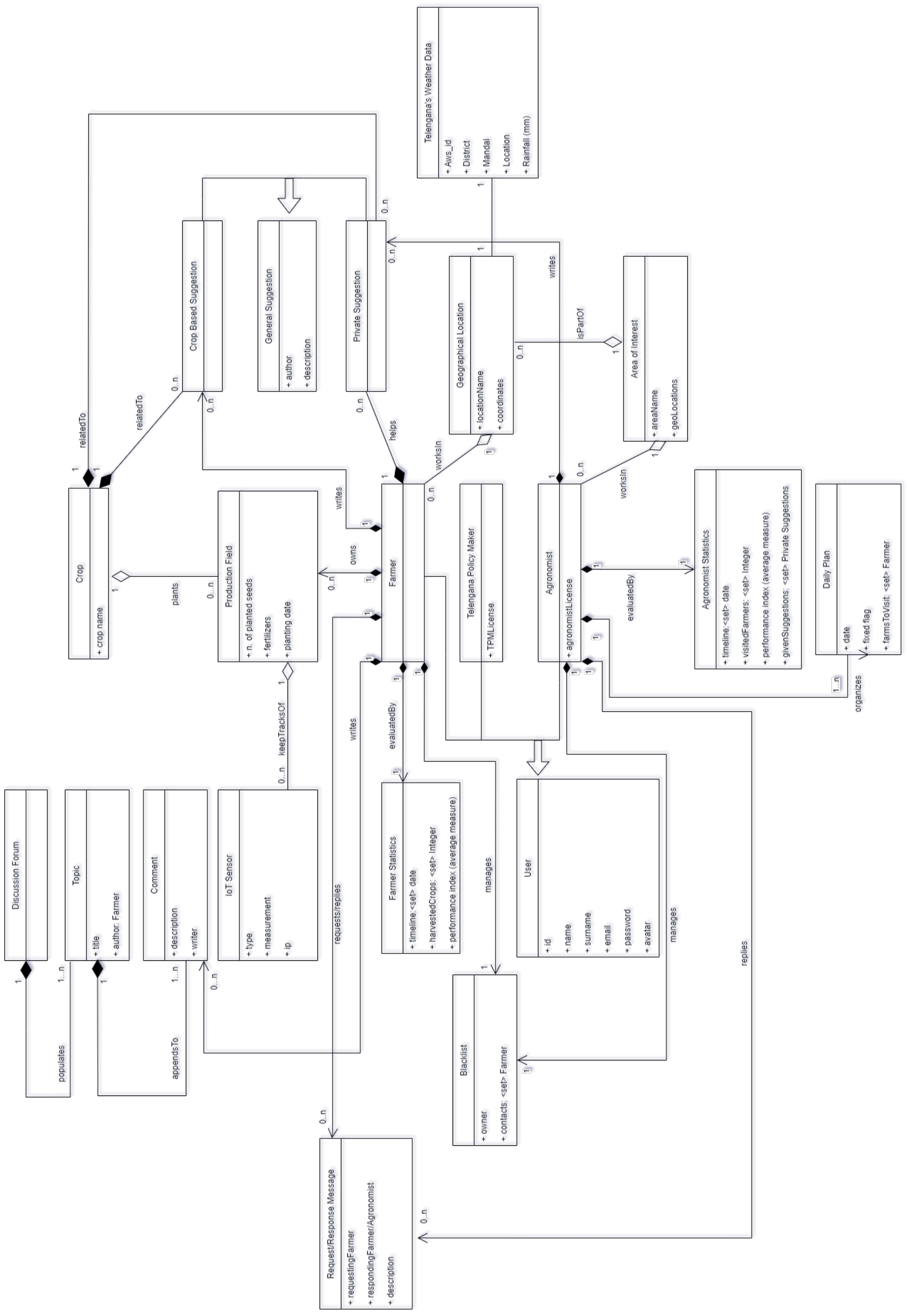


Figure 1: High-level model structure

2.1.2 State Diagrams

Considering now the main functionalities of the system, it is important to highlight the events that make its objects change their state. State diagrams are used to describe the most critical aspects of the objects previously described in the UML class diagram and how the systems manages them (Figure 1).

Data Production Field At the beginning the farmer compiles a form where he inserts the type of production (i.e the crop), the amount planted, the fertilizer, the date and the possible Iot devices associated. This event defines the beginning of a new production, so the system receives the data in the *unprocessed* state, then it checks the correctness of the fields filled and at the end they are *stored* in the data base. Here, the farmer can do different actions like doing *updates* periodically inserting a comment, photo and a date. Also the *IoT devices* declared weekly sends their data to the system and then stored in the database. At the end of a production cycle for the crops specified, the farmer inserts the ultimate data production, so *the amount effectively produced.* The system computes the *index performance index* as the ratio between amount produced and planted. It stores the index in the database as *statistics* and the diagram ends.

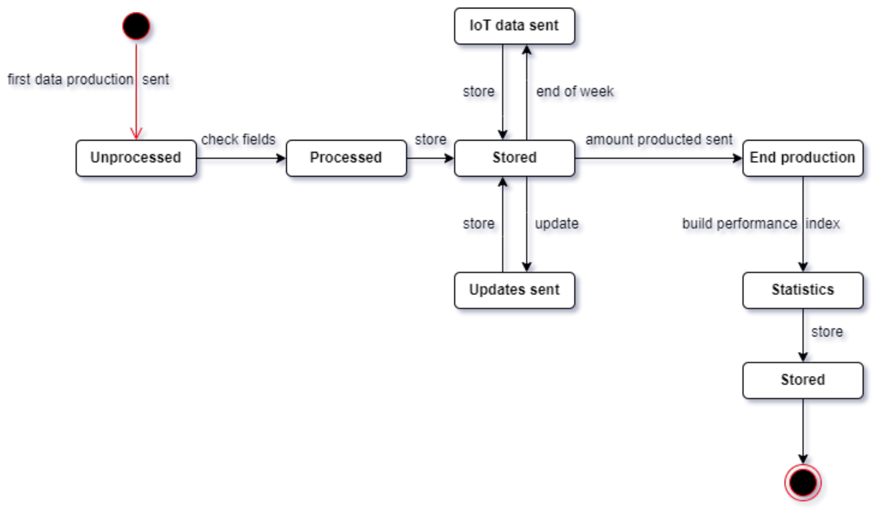


Figure 2: Data Production state diagram

Request One of the main functionalities of the system is the intercation between farmers and agronomists through a direct *request* for help. It is implemented like a chat window where it is possible to send messages, specifing the user selected and content of the message, and receive answers with little particular exceptions. In this situation the system search a farmer search an agronomist or another farmer and send *request* data. The system doesn't send the request immediately to the possible receiver because it must check if he/she is present in a *blacklist*. Every Agronomist and Farmer can block spammer users. If the check is completed succesfully, the request is *stored* in the database, the system sends a *notify* to the receiver *marking it as un read.*

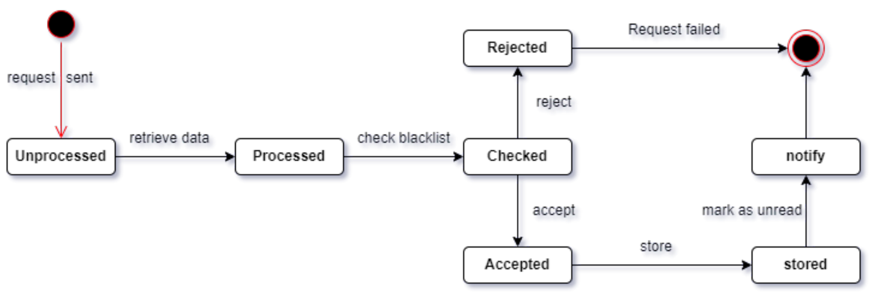


Figure 3: Request state diagram

Forum This state diagram emphasizes the forum functionality where farmers can interact surfing different topics and joining discussion. Each Farmer can publish a *new topic* once per day to avoid spam, furthermore the system wants to discard off topic or recurring discussions. Every time a new topic is sent (where the title and the content is declared), the system checks the *day limit* of the publisher, then it is *accepted* and the topic is *stored* in the database and visible to every users. Now we define a visible topic as *regular.* Every user can report a topic, then the system checks the number of *report* reached, if it is overtaken, the system *deletes* the topic, otherwise the topic is still regular and visible to the users. When a user wants to *comment* a topic, the system inserts the answer it in the database.

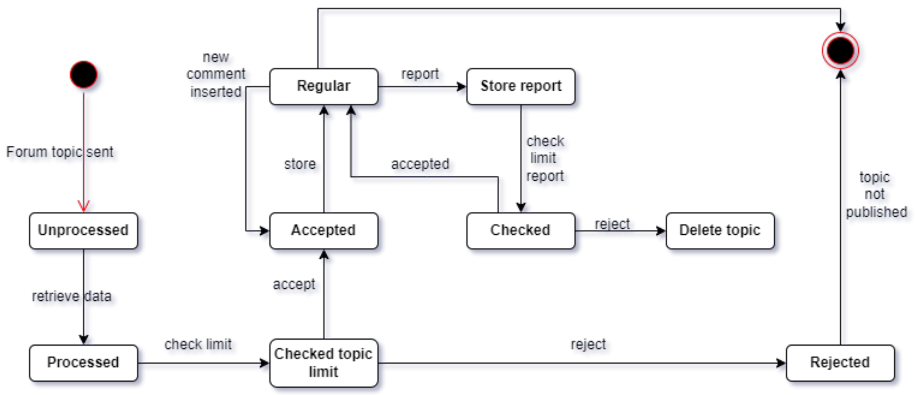


Figure 4: Forum state diagram

Daily Plan This state diagram emphasizes the *daily plan* functionality that describes how the agronomist's daily plan is managed. At the beginning of a *new day* the system shows the *new empty daily plan* tab for each agronomist. Now the user can modify the daily plan as he wants, he *adds visits* and then he can *update* the plan *deleting* some existing ones or adding other visits. The page displays the *confirm* button and only when the agronomist ends, so he has definitively decided which farm he must visit or he just visited, the user confirm the completation of the plan and it cannot be modified anymore and stored in the database. If a plan of a certain day is not confirmed, it will appear in the page in the future days, the user can modify past daily plans and confirm them. At the end of each month if some plans are not confirmed by the user, they will be confirmed definitely by the system. Each time a plan is confirmed, the visits to the farmers definied will be marked.

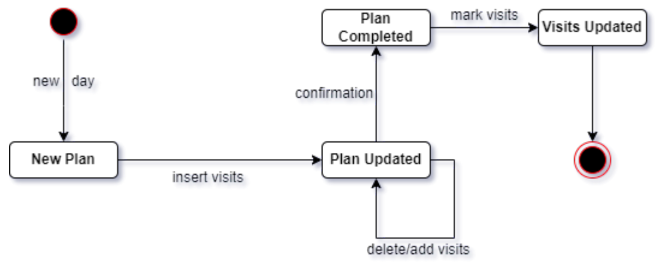


Figure 5: Daily Plan diagram

2.2 Product Functions

DREAM is an application that, on the one hand, provides a better cooperation between farmers, agronomists and TPMs, through direct requests or forums, as a social application, on the other hand the system monitors the development of the production through the computation of the statistics in order to improve the results. So one of the crucial function of the system is the collection of **data production**, and how the data are managed and visualized. Thanks to the geolocalization of the farmers and the assignment of the agronomists to a certain area is possible to simplify the visits to farms through the **daily plan** function. **Forum**, **request** and **personalized suggestions** functions are responsable for users interaction and knowledege exchange.

Before starting with the description it is important to highlight that in order to benefit of DREAM functionalities the customer **must** be logged in the system as he can be recognized.

2.2.1 Data Production Field Function

Farmers are allowed to insert data production thanks to DREAM: the managing process takes place entirely inside the application. It starts when the farmer plants a specified amount of a new crop and ends when the production is completed and it is known the amount of that crops has been produced. So in this way all coltivations cycle of a farm are well tracked constantly. Each farmer can also insert **updates** periodically through a text or photo about the status of the plants. Furthermore Iot devices send weekly data abount system irrigation and the humidity of the soil. This is usefull to keep Agronomists and Farmers up to date about the progession.

The data production functionality is provided to the user by an interface that allows him to insert all the information needed to store, manage and visualize data. The fields a user fills are reported here with a label that describes if they are mandatory or not.

- **Crop** `[MANDATORY]`**:** users must select a crop between a vaste selection. The system doesn't permit user to insert whatever text he or she wants to insert to avoid useless errors. This field is crucial to identify a specific coltivation and to receive in the future personalized suggestions for the crop inserted. It is possible to insert the same crop of an existing coltivation because maybe it has been planted in another moment or in different way.
- **Amount** `[MANDATORY]`**:** this numerical field is marked as mandatory because, as we have previously said, the system manages the data to build statistics, so this is essential to compute the build performance index of that specific coltivation for that farmer. Obviously this input must be well managed and checked because no negative values are accepted.
- **Fertilizer** `[OPTIONAL]`**:** users need to give additional information providing also the type of the fertilizer to add more information on the method of the cultivation. It is not mandatory because it may be possible to avoid using fertilizer.
- **Date** `[MANDATORY]`**:** user must insert the start date of the production, this date must be equal to that of the planting date. The system doesn't fill this field automatically providing the users to add the new data production field whenever he wants.
- **IoT sensors** `[OPTIONAL]`**:** users should be able to monitor the humidity of the soil or the irrigation system through Iot sensors which, if they are available, send their values weekly directly to DREAM in order to **update** the status of the cultivation. This field must be filled with the IP address of the IoT devices.

All these fields are crucial to initialize a new data production field cycle and monitor it. Once all these data are sent and stored to the database, they are visible on the farmer data page to give also usefull information to agronomists to know how the farmer is working. Only the agronomists of the area which the farmer belongs, can visualize these data. *Crop* is used to retrieve all the suggestions released about the crop specified. *Amount* is a value used during the phase of end production to compute the performance index related only to that production cycle. The performance index is built on the dependencies of amunt planted, amount harvested and weather forecasts (e.g rainfall value).

2.2.2 Request Function

The *Request function* is used to create an interaction between Farmers and Agronomists, but in particular the idea is that to give the possibility to Farmers for *solving problems* or asking curiosities directly to someone more competent in a specific field. Before sending the request through a *message*, the system allows Farmers to *search* whoever he wants and select the Farmer or the Agronomist he wants to get in touch with. So the Farmer can search another known and competent Farmer or maybe the Agronomist of its specific area. He must know the name and the surname of the user. When the user has been select, a new window appears and a field must be filled:

- **Description** `[MANDATORY]` **:** This is a text area where the Farmer inserts the problem or whatever he wants to ask to the receiver. The field must not be empty, otherwise the message is not be sent.

After the request has been sent, the addressee will receive a new *notification* and the new request appears in the home page. The receiver (e.g Farmer, Agronomist) can check the requests, read the message and reply thorugh the same method. Farmer and Agronomist in their *home page* visualizes all the requests received and sent. The request function is nt available for that users inserted in the **blacklist**

2.2.3 Forum Function

Farmers usually don't get in touch in the real life because of the distance, working solitary in the farms. It is crucial an horizontally sharing of knowledge especially who plants the same cultivations or who deals with the same climate and soil conditions for example. The system shares a space dedicated only for Farmer through the Forum function. Each Farmer can enter the Forum visualizing topics and comments related to them, creating a new Topic, or partecipating to an existing topic through new comments. In this way DREAM connects all the Farmers registrated to the platform. When the Farmer adds a new topic must fill some fields:

- **Title** `[MANDATORY]` **:** This field is a text area and it usefull to identify briefly the topic discussion and what is inserted appears in the list of topics available in the forum home page.
- **Question** `[MANDATORY]` **:** When a new topic has been created, the discussion has to start with a question point, something that attracts answers and new comments in order to create a comparison. This field is a text area.

Every time a Farmer selects a topic, he visualizes all the comments related and he can add a new comment with a simple text area. After this, the new comment is visualized by all farmers. All topics or comments that are not ineherent will be reported by the users, so at a certain point, when the limit number of reportation has been reached, the system deletes them. This is important to avoid spam and delete useless topics.

2.2.4 Daily Plan Function

Agronomists have to monitor the progession of the farmers' production, helping them with appropiate suggestions, answering their requests and especially visiting them directly in their farms. The agronomist must visits all of the farmers of their area of interest at least twice a year. The daily plan function keeps track of the visits for each agronomist, organizing in an appropriate manner agronomist daily work. Every day a new daily plan is published in the home page of the agronomist where he can interact with, managing, adding or deleting appointments. When the agronomist add an appointment, he must fill some fields:

- **Search** `[MANDATORY]` **:** The agronomist when add a new appointment, must declare the farmer that is visited. He searches through name and surname and he visualizes only farmers on its area of interest.
- **Note** `[OPTIONAL]` **:** This fields is a textarea that adds some information to the visit defined.It helps the agronomist to remember something in particular because it appears in the home page.

A daily plan can be confirmed whenever the agronomists want (also after some days), but in this way, the system do not allow agronomist to manage that daily plan for that day anymore. After the confirm, it will be stored and the system updates the visits. At the end of each month, every daily plan not confirmed, will be definitively stored by the system. The agronomist is notified when a farmer has not been visited.

2.2.5 Suggestion Function

The Agronomist can't interact directly with the farmers but he answers only to the request received, but now with the suggestion function is possible. Each statistic, update of the farmer can be seen and analyzed by the agronomist who is responsible of suggesting them correction, or best practies to improve their production, for example. This kind of suggestion is addressed only to farmer specified, so the system calls it **private suggestion**. When an agronomist wants to give a suggestion to a certain farmer, after he selects the option on the production field page of the farmer, he must fill some fields:

- **Crop** `[OPTIONAL]` **:** The agronomist must specify which crop the suggestion is related to. He can choose also to write the suggestion wthout specifying a crop.
- **Description** `[MANDATORY]` **:** It is a text area where the agronomist describes the suggestion to give.

There exists another type of suggestion called **crop suggestion**. In this case, the suggestion is written by a farmer towards all the other farmers, but this is visualized in the production field of each farmer. In this situation the crop field is mandatory because the suggestion is visualized by all farmers who planted that specific crop.

2.3 User Characteristics

DREAM in Telangana has three different figure that are very important to distinguish in order to provide correctly the functionalities previously described in subsection 2.2.

- **FARMER**

- Can register to the application.
- Always inserts correct data while registering.
- Can login in the application.
- Specify one and only one geographical area of working.
- Can insert in the system data about their production.
- Can insert updates related to a production field in the system.
- Can visualize data relevant (e.g build performance index, charts) to its production.
- Can access to the forum, reading all topics, adding new one or comment an existing one.
- Can retrieve suggestions from another Farmer or from an Agronomist.

• **AGRONOMIST**

- Can register to the application.
- Always inserts correct data while registering and insert their vaild license.
- Can login in the application.
- Specifies at least one area of interest.
- Can visualize data concerning farmers that belong to its area of interest.
- Can visualize best and worst performing farmers.
- Can receive requests from a farmer.
- Can include a farmer in a black list to avoid spam requests.
- Can send suggestions to the farmer.
- They have a daily plan about their visit to the farmer.
- Can update their daily plan.
- Can confirm a daily plan.

• **TELANGANA'S POLICY MAKER**

- Can register to the application.
- Always inserts correct data while registering and insert their vaild license.
- Can login in the application.
- Can visualize farmers sorted by their performance.
- Identify those farmers who need to be helped.
- Can visualize Agronomist statistics.

2.4 Domain Assumptions

Domain assumptions are used to define clearly the world in which DREAM works. Thanks to them, we are able to add constraints that define the bounds of the environment.

We assume that these assumptions hold true in the domain of the system:

DA1 Data concerning meteorological short-term and long-term forecasts are available from third party components.

DA2 Areas inserted by Agronomists and Farmers correspond to existing geographical locations.

DA3 Agronomists visit all farms at least twice a year.

DA4 Agronomists visit under-performing farms more than twice a year.

DA5 Information provided by the farmers about their work are coherent with their production.

DA6 Sensors devoted to obtain territory's measurement work and provide good approximations.

DA7 The water irrigation system works and provides correct measurements.

DA8 Farmer's request for help are well formulated and understandable.

DA9 Agronomists actually visited farms when they declare it in the daily plan.

DA10 Agronomist visits the same farms which has been declared in the daily plan.

DA11 The person who visits the related farm is verified to be the same agronomist declared in the system.

DA12 Suggestions related to production are well formulated, understandable and corenert with the corresponding request.

DA13 Suggestions related to production are actually coming from a competent farmer/agronomist.

DA14 Agronomists and Telengana's policy maker own a valid license.

DA15 Geographical area of interest are divided somehow equally among agronomists.

DA16 Each Farmer works only on a single location.

DA17 Sensors forwards error-free data through the communication network.

DA18 The farmers registered truly have a farm in the location specified.

DA19 Registered user are associated to existing and valid anagraphical identities.

DA20 IoT Irrigation system sensors send periodically and correctly data to the system about the amount of water used.

DA21 IoT territory sensors send periodically and correctly data about the measurement of the humidity of the soils.

DA22 Each photo and comment uploaded by the farmer is coherent with that specific crop published.

3 Specific Requirements

This section is devoted to a specific description of every kind of requirement our system has to deal with in order to achieve all the functionalities described.

3.1 External Interface Requirements

3.1.1 Customer Interfaces

The following interfaces are meant to give a first description of how the functionalities will be offered in practice to the customers of DREAM. A precise description of how each mockup precisely reflects one functionality of DREAM is given in the Design Document. It is sufficient for now to start with their illustration as we can understand the relation with the goals (section 1.2.1) and use cases (section 3.3.3) described in this document.

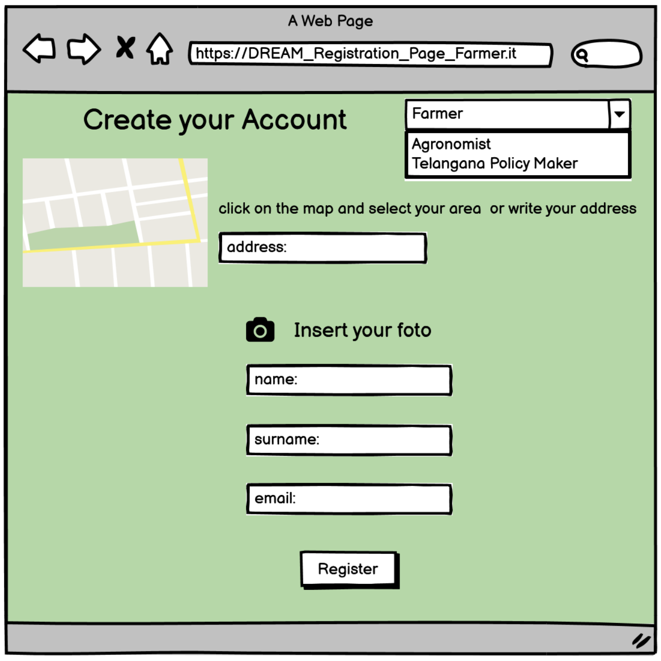


Figure 6: Registration Farmer

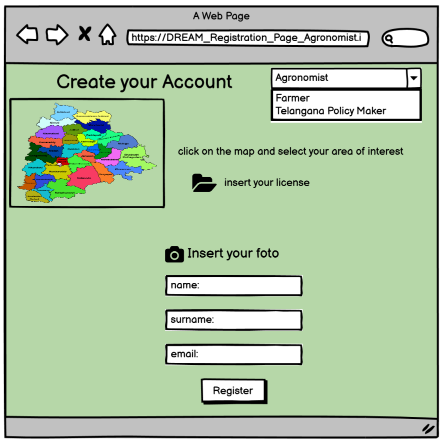


Figure 7: Registration Agronomist

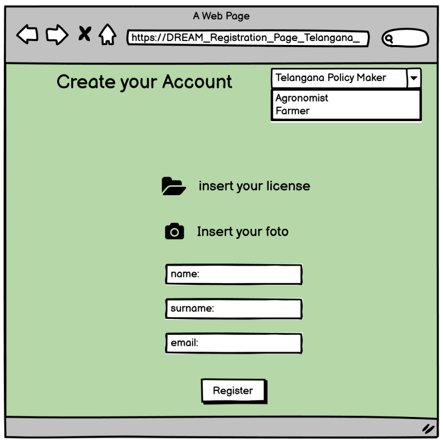


Figure 8: Registration TPM

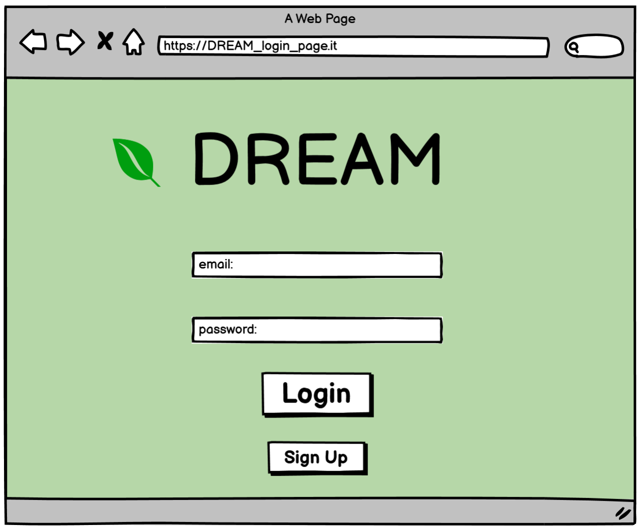
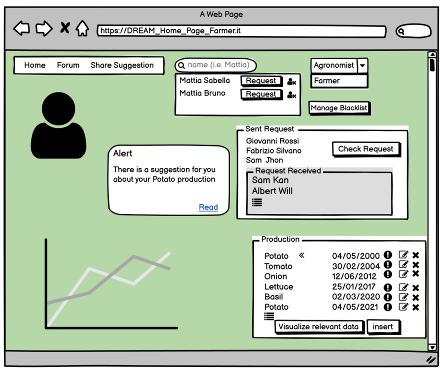
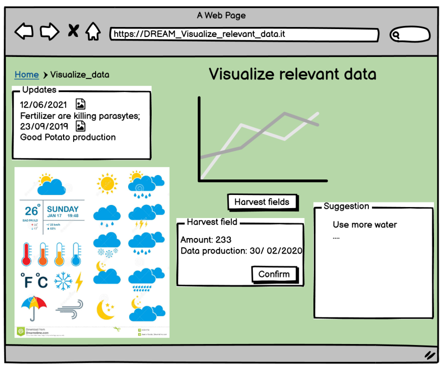


Figure 9: Login Page



```
Potato   «   04/05/2000
Tomato       30/02/2004
Onion        12/06/2012
Lettuce      25/01/2017
Basil        02/03/2020
Potato       04/05/2021
```

Figure 10: Home Page Farmer



12/06/2021
Fertilizer are killing parasytes;
23/09/2019
Good Potato production

Amount: 233
Data production: 30/ 02/2020

Use more water
....

Figure 11: Farmer Production Field Page

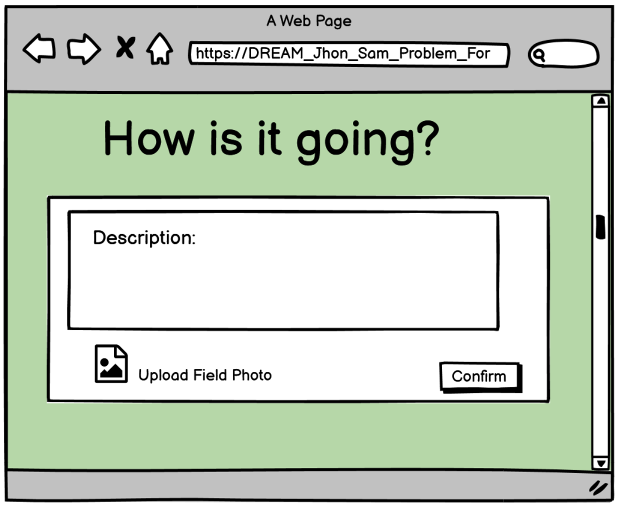


Figure 12: Updates page

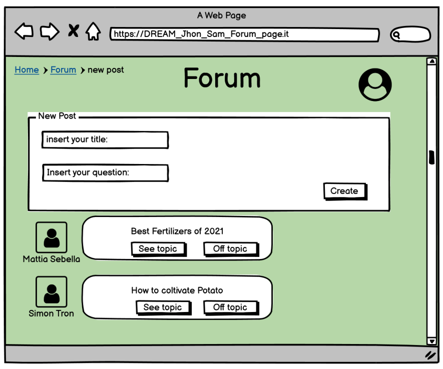
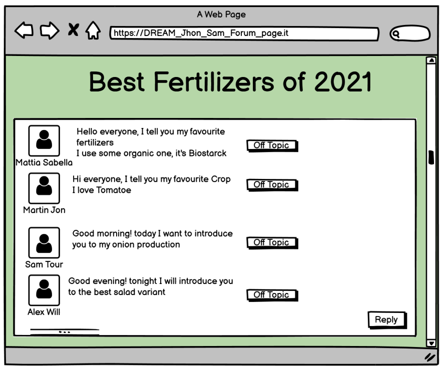


Figure 13: Forum Page



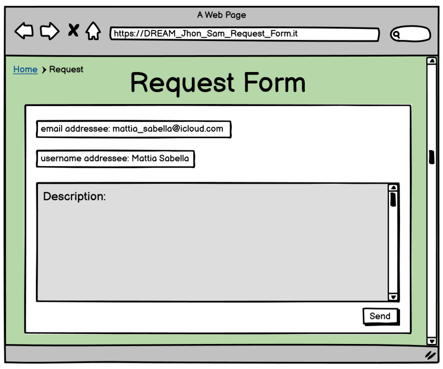
Mattia Sabella
Hello everyone, I tell you my favourite fertilizers
I use some organic one, it's Biostarck

Martin Jon
Hi everyone, I tell you my favourite Crop
I love Tomatoe

Sam Tour
Good morning! today I want to introduce you to my onion production

Alex Will
Good evening! tonight I will introduce you to the best salad variant

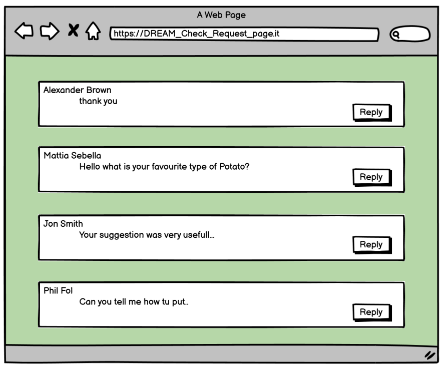
Figure 14: Topic Page



email addressee: mattia_sabella@icloud.com

username addressee: Mattia Sabella

Figure 15: Request Form Page



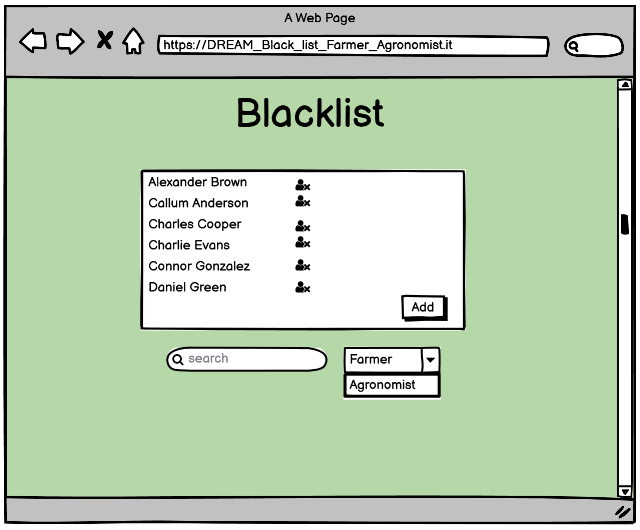
Alexander Brown
thank you

Mattia Sebella
Hello what is your favourite type of Potato?

Jon Smith
Your suggestion was very usefull...

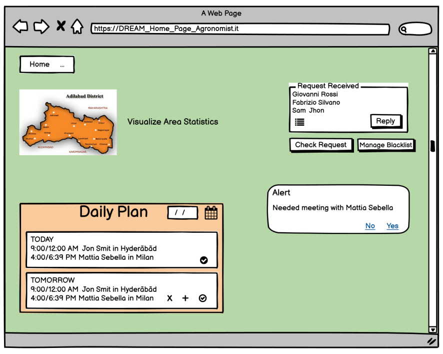
Phil Fol
Can you tell me how tu put..

Figure 16: Requests List Pge



Alexander Brown
Callum Anderson
Charles Cooper
Charlie Evans
Connor Gonzalez
Daniel Green

Figure 17: Blacklist page



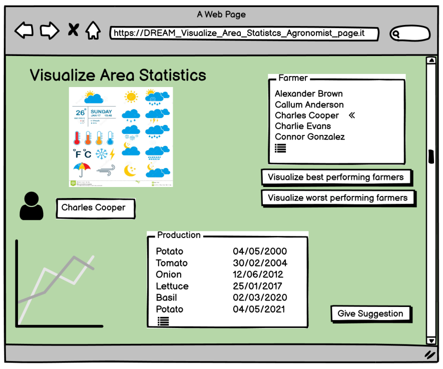
Giovanni Rossi
Fabrizio Silvano
Sam Jhon

Alert
Needed meeting with Mattia Sebella

TODAY
9:00/12:00 AM Jon Smit in Hyderābād
4:00/6:39 PM Mattia Sebella in Milan

TOMORROW
9:00/12:00 AM Jon Smit in Hyderābād
4:00/6:39 PM Mattia Sebella in Milan

Figure 18: Agronomist Home Page


A Web Page
https://DREAM_Visualize_Area_Statistcs_Agronomist_page.it
Visualize Area Statistics
Farmer
Charles Cooper
Visualize best performing farmers
Visualize worst performing farmers
Production
Give Suggestion

Alexander Brown
Callum Anderson
Charles Cooper
Charlie Evans
Connor Gonzalez

Potato	04/05/2000
Tomato	30/02/2004
Onion	12/06/2012
Lettuce	25/01/2017
Basil	02/03/2020
Potato	04/05/2021

Figure 19: Statistics Agronomist Page

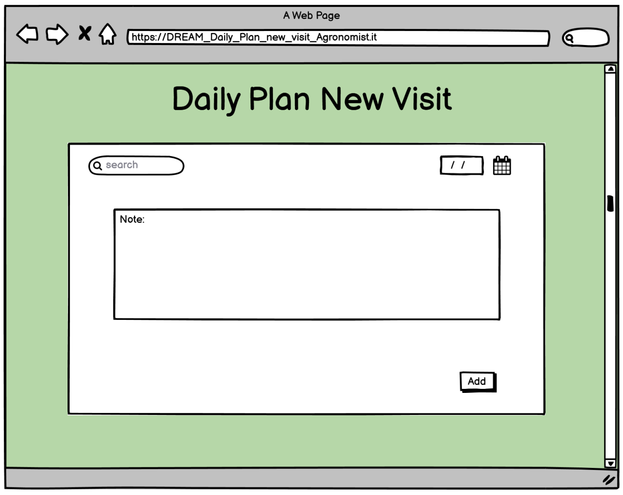


Figure 20: Daily Plan page

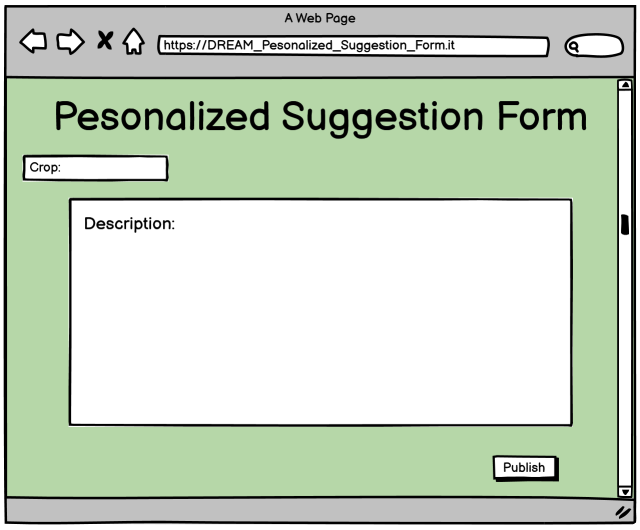

A Web Page
https://DREAM_Pesonalized_Suggestion_Form.it
Pesonalized Suggestion Form
Crop:
Description:
Publish

Figure 21: Suggestion Form page

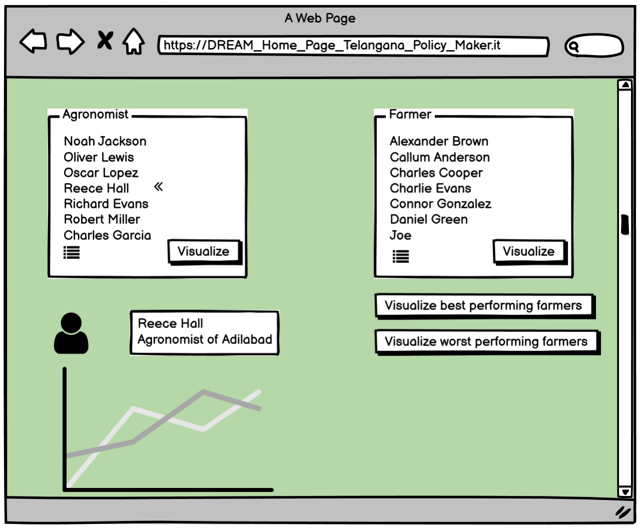


Figure 22: TPM Home Page

3.1.2 Hardware Interfaces

Hardware interfaces to consider are the sensors that must send all the information (eg meteorological info) useful for their production to the farmers. Furthermore a camera useful for Farmers to take pictures of their production to share with the agronomists, to show the progress of their production.

3.1.3 Software Interfaces

The interfaces used by DREAM is an update browser and MySql. We will need to manage all the information with different logical models in order to store the different kind of data that the application has to deal with.

3.1.4 Communication Interfaces

Important communications interface to be considered is that system must allow Telangana Policy Maker, Farmer and Agronomist to visualize/insert data, so we weed a communication standards such as HTTP, and FTP to upload license of Agronomist or Telangana Policy Maker, and the pictures uploaded by the Farmer. Moreover Farmer can talk to the Agronomist or to other Farmer, so all messages will be encrypted with an asymmetric key.

3.2 Functional Requirements

This subsection aims to give a *complete* description of the *functional requirements* of our system by defining them together with the domain assumptions that satisfy the identified **goals**.

3.2.1 Requirements

R1 Users must be authenticated as Farmer, or Agronomist, or Telengana's Policy Maker.

R2 Farmer, Agronomists and Telengana's Policy Makers register to the application through mandatory fields.

R3 In phase of registration, Farmers must specify one and only one geographical area of working.

R4 Agronomists and Telengana's Policy Makers who registered inserted in the application their valid license.

R5 Agronomist must specify in the system at least one area of interest.

R6 Agronomists will access farmer's data and statistics only from that ones in their area of interest.

R7 Private requests between a farmer and an agronomist can't be accessed by other users.

R8 Suggestions related to a particular crop field can be accessed by every farmer who owns that crop field.

R9 Private suggestions coming from an Agronomist can be accessed only by the Farmer who received them.

R10 Agronomists can access to the list of request from farmers.

R11 Agronomists can reply to farmer's request through a chat window.

R12 The system must keep track of how many times a farm has been visited.

R13 Monthly the system notifies the agronomist with the list of farms which has not been visited at least twice.

R14 The system must allow the Agronomist to keep updated his daily plan.

R15 After a proper time window, the daily plan declared by an Agronomist becomes fixed and immutable.

R16 Topics in discussion forums must be initialized by a farmer.

R17 The system allows farmers to access public discussions.

R18 Off-Topic discussions can be reported by farmers to be deleted.

R19 When a topic is reported a certain number of times, the topic will be deleted.

R20 The system allows agronomists to block spam requests by farmers by including them in a blacklist.

R21 The system allows farmers to insert production data through mandatory fields of a form.

R22 Farmers can visualize statistics about their production data built by the system.

R23 Statistics build over farmer's production, performance etc are based on three parameters: amount of crop harvested, amount of crop planted, weather conditions.

R24 The system allows farmers to search agronomists/farmers they want to contact.

R25 Farmers can send requests directly to agronomists and other farmers.

R26 Farmers visualize Agronomists' replies in an inbox window.

R27 Farmers and Agronomists visualize other Farmers requests in an inbox window.

R28 Allow Agronomists to visualize weather forecasts in their area of interest.

R29 Allow Farmers to visualize weather forecasts in their geographical area.

R30 The system classifies best performing farmers based on their performance index.

R31 The system allows Telengana Policy Makers to visualize best and worst performing farmers according to their performance indexes.

R32 The system allows Telengana's Policy Makers to visualize Agronomists statistics.

R33 Farmers can share pictures of their production field through the system.

R34 Farmer and Agronomists can manage a blacklist containing contacts of other Farmers.

R35 The system allows agronomists and farmers to reply to other farmer's help request.

R36 The system allows Agronomist to visualize best and worst performing farmers according to their performance indexes.

3.2.2 Mapping

Goal	G1
Domains	DA1, DA2, DA5, DA6, DA7, DA16, DA17, DA18, DA19, DA20, DA21, DA22
Requirments	R1, R2, R3, R21, R22, R23, R29, R33

Table 1: *G1 Mapping*

Goal	G2
Domains	DA5,DA18, DA19,DA22
Requirments	R1, R2, R3, R21, R33

Table 2: *G2 Mapping*

Goal	G3
Domains	DA8, DA12, DA13, DA18, DA19
Requirments	R1, R2, R8, R16, R17, R18, R19, R24, R25, R27, R34, R35

Table 3: *G3 Mapping*

Goal	G4
Domains	DA3, DA4, DA8, DA11, DA12, DA13, DA14, DA19
Requirments	R1,R2, R4, R7, R9, R10, R20, R24, R25, R27, R34, R35

Table 4: *G4 Mapping*

Goal	G5
Domains	DA1, DA2, DA3, DA4, DA9,DA10, DA11,DA14, DA15, DA19
Requirments	R1,R2, R4,R5,R6, R12, R13, R14, R15, R28

Table 5: *G5 Mapping*

Goal	G6
Domains	DA1, DA2, DA5, DA14, DA15, DA19
Requirments	R1, R2, R5, R6, R21, R22, R23, R28, R30, R36

Table 6: *G6 Mapping*

Goal	G7
Domains	DA1, DA5, DA14, DA19
Requirments	R1, R2, R3, R4, R21, R22, R23, R30, R31

Table 7: *G7 Mapping*

Goal	G8
Domains	DA2, DA3, DA4, DA11, DA14, DA15, DA19
Requirments	R1, R2, R4, R5, R12, R30, R32

Table 8: *G8 Mapping*

3.3 Use Cases Identification

In the following subsection the usage of the system is going to be described first with a general description using scenarios and then in a more specific way using use case diagrams.

3.3.1 Scenarios

Consider these scenarios in order to clarify the usage of the system and the further description with use case diagrams.

SCENARIO 1 FARMER MANAGES HIS FIELDS
Odoacre is a newbie farmer, and he owns so many lands (one of them is full of potatoes) that he does not know how to properly manage them. Odoacre discovered DREAM, a service that can help him to achieve his goal of improving, then after registration he puts in the system data about his production fields. For each declared field Odoacre waits 2 seconds (or less...) until the system calculates for him statistics about weather forecasts of his geographical location of his lands. Odoacre now has more knowledge on how to plan his production for the next week according to weather forecats, but still he is a newbie and wants to know something else and more specific about his type of production, furthermore, he faces a problem: his potato land does not grow properly because of parasytes. Odoacre notices that he can actually declare his problem in the system and upload a photo of his field; it does so, and he hopes that someone some day visualizes what he uploaded and then helps him. The next day Odoacre opens again DREAM and sees personalized suggestions coming from experienced farmers and agronomists about his potato land, which kind of fertilizers and pesticides he should use and even which kind of crops he should plant. Odoacre "harvests" theese golden information and puts in practice what he learned, and sooner or later he will notice an improvement on his production. Indeed, after a month or so, he checks his potato field and notices that finally, twentynine out of thirty potatoes are fully grown and they are ready to be gathered. Orlando then updates DREAM parameters of his production field, and he declares that his monthly production for the potato land is ended, and the field is now empty and ready for the next growth. DREAM stores his declaration, resets all the parameters and immediately updates the performance index of the Farmer according to how many potatoes he harvested and how many of them he planted in that month.

SCENARIO 2 FARMER SHARES SUGGESTIONS
Glauco is an experienced farmer who masters the art of agriculture, and he wants to share his knowledge with other farmers. He keeps track of his failures and successes and writes a text full of personalized suggestions useful for manage production fields according to the type of production, the weather etc... Since Glauco uses DREAM a lot, he shares information contained in his text through the system, which analyzes it and forwards specific suggestions to specific fields of other farmers in need. Odoacre (who now needs help with his carrots land) and Gastone (who faces the

same problems of Odoacre) see in their Carrots Field Panel the suggestions of Glauco, since they were forwarded by the system itself.

SCENARIO 3 FARMER COMMUNICATES THROUGH DISCUSSION FORUMS
Melchiorre is a farmer who plants crops which are really unpopular. Since 90 percent of experienced farmers does not think that someone would plant uncommon crops (like Achuqcha), no one of them even prepared personalized suggestions about these kind of batch. Melchiorre wants to improve is production of Achuqcha, but he does not see any personalized suggestion about his production field! Melchiorre won't give up, he discovered that DREAM offers the possibility of create discussion forums with other farmers, and thinks that maybe in this way multiple farmers will notice him. He creates the discussion forum, writes the first post describing all his needs with the Achuqcha field, and waits for eventual replies. Telemaco, Tebaldo and Teodosio are three farmers that plant Achuqcha since many years, and they use DREAM a lot. They periodically checks the discussion forums present in the system, and they see a topic named "I need help with Achuqcha!" created by Melchiorre. They open it, they read the first post and immediately they formulate and send a reply, each one coming from a different farmer (on different computers...). Melchiorre is so anxious that he refreshes the discussion forum page so many times, and finally he notice three useful replies coming from Telemaco, Tebaldo and Teodosio. As expected, multiple farmers which believed to be the only ones in the world that plant Achuqcha replied to him, and helped him with his production field; Melchiorre is now happy and can improve his production.

SCENARIO 4 FARMER AND AGRONOMISTS COMMUNICATES THROUGH DIRECT MESSAGES
Olivia is a farmer who plants the strange (but delicious) crop of Kiwano. Unlike Melchiorre, Olivia does not want that eventual replies to her requests of help are shared with other Farmers, she is a greedy person, so she avoids the use of public discussion forums. Still, Olivia needs help with his production field of Kiwano, as no one has still inserted in the system personalized suggestions about this kind of crop. Olivia discovers that DREAM offers the possibility to send direct and private email-like messages to a Farmer or to an Agronomist, so she formulates his request and she send it to two persons, Attilio (a farmer who plants Kiwano too) and Nicodemo (an Agronomist). On the other side, Attilio and Nicodemo check their inbox in DREAM and notice the message of Olivia; (individually) they write a reply full of suggestions and they forward it to Olivia. Olivia sees the reply on her inbox in DREAM, reads it and finally she can improve her production of Kiwano.

SCENARIO 5 AGRONOMIST VISUALIZES AREA STATISTICS
Amilcare is an agronomist whose working area of interest is the district of Adilabad in Telengana. Since Amilcare uses DREAM a lot, he often uses the "Visualize Area Statistics" function of the system to see weather forecasts of his area of interest. He can also see the performance indexes of farmers working on his same geographical area, and he notices that among them Frida is performing particularly well (bad), so he decides to investi-

gate by expanding information about her production fields. Amilcare can now see which crops Frida is planting and how her production is going; he finds out that Frida faced a problem with her field of corn, furthermore he checks the photo she published about her land, so he thinks that her field of strawberries can be improved by changing fertilizer (even if Frida didnt face any problem about this production field). So Amilcare formulates two suggestions for the two fields and uploades them into the system, that in turn store and forwards them to Frida.

SCENARIO 6 AGRONOMIST MANAGES HIS DAILY PLAN
Gualtiero is an agronomist who uses DREAM and organizes is work with the help of a daily plan. He knows that he has to visit every (or at least a minimum quantity of) farm present in his area of interest, with special attention to in-need farms. Since farms are a lot and his memory is not so good, he decides to check what farms he should visit according to the daily plan automatically created by the system. He discovers that he should visit the farm of Egidio because it has been visited only once for the current year, and the minimum is two, furthermore he decides to visit also the farm of Brigitta even if it has been visited twice, because she keeps facing problems with her field of zucchini. Once visits are completed and Gualtiero returns to home, he opens DREAM and wants to make sure that the system "knows" that he followed the daily plan; suddendly, Gualtiero remembers that during his daily trip to farms he did a deviation to check the farm of Germana, even if it was not listed in the current daily plan (maybe in the next one). So he declares in the system all visited farms and deviations and then confirms his updates; the system stores the changes, marks the daily plan as "ended" and makes it immutable.

SCENARIO 7 TPM VISUALIZES AGRONOMISTS
Sigismondo is a Telengana Policy Maker and wants to understand whether the steering initiatives carried out by agronomists produce significant results. Sigismondo discovers that DREAM allows him to list all agronomists and see their improvement strategies. He wants to check the work of the agronomist Ezechiele, so he expands the selection about him and reads all the steering initiatives that he carried out. Sigismondo now knows how Ezechiele is performing, and he saves this information for legal purposes.

SCENARIO 8 TPM VISUALIZES FARMERS' PERFORMANCES
Domitilla is a Telengana Policy Maker that wants to check the overall performance indexes of all farmers registered in DREAM. She discovers that the system allows her to list all farmers and shee their performance indexes. Domitilla now knows every performance index for every farmer, and she saves this information for legal purposes.

SCENARIO 9 USER AUTENTICATES TO DREAM
Gardenia, Giacinto and Gelsomina are respectively an Agronomist, a Telengana's Policy Maker and a Farmer that have heard of DREAM, so they want to register to it. They open the sign up page and they fill the mandatory forms; in particular: the system requires that Gardenia and Giacinto upload a valid Agronomist/TPM license as they will provide legal and serious information to third parties/people, and this require

some kind of certifications. Furthermore, the system requires that Gardenia and Gelsomina specify their area of interest/geographical location, as these information will be useful to make DREAM work for them. Finally, registration can be completed, and the three of them can Login whenever they want to use DREAM or Logout whenever they want to stop the navigation.

SCENARIO 10 ACTIONS AGAINST SAUCY FARMER
Orlando, a farmer, is furious because he does not like the reasonable suggestions provided by other farmers and agronomists, he wants more, he wants the golden secret rule of farming. The best thing he wrongly thinks to do is to spam useless help requests to farmers and agronomists, sometimes even offending them. The farmer Rinaldo and the agronomist Astolfo are tired of Orlando's spam messages, so they decide to blacklist him as a spammer; Orlando can't no more disturb Rinaldo and Astolfo through spam messages. Orlando won't give up, so he loses his mind again and now creates useless discussion topics and replies with off topic posts, he is mad. Angelica, a farmer who notices Orlando's spam and impropriate post, reports it, and so do Medoro, Ferraù, and the other farmers. Sooner or later the minimum number of reports per post will be received, hence the spam post of Orlando will be removed, and he will receive a warning. Furthermore, Orlando can't create another discussion topic, as he reached the daily topic limit of one. Rinaldo however misses his friend, and he really hopes that Orlando will recover his good manners soon; when it will happen he will remove from the blacklist his friend, so that the both of them will exchange messages again.

3.3.2 Use Case Diagram

The following diagram is a high-level description of the possible interactions of actors with the system and highlights the different use case in which actors are involved.

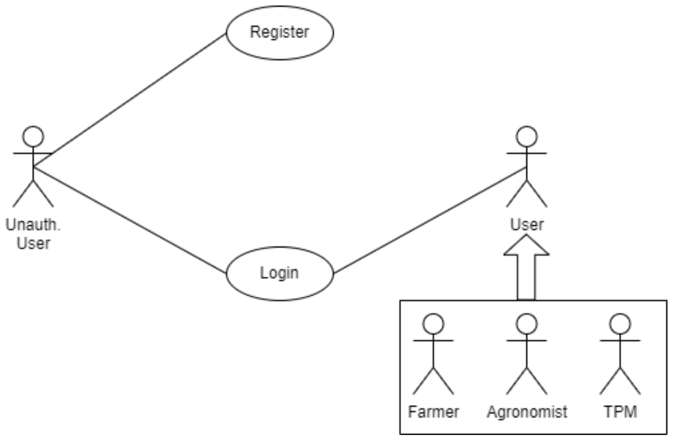


Figure 23: Registration and Login Use Case Diagram

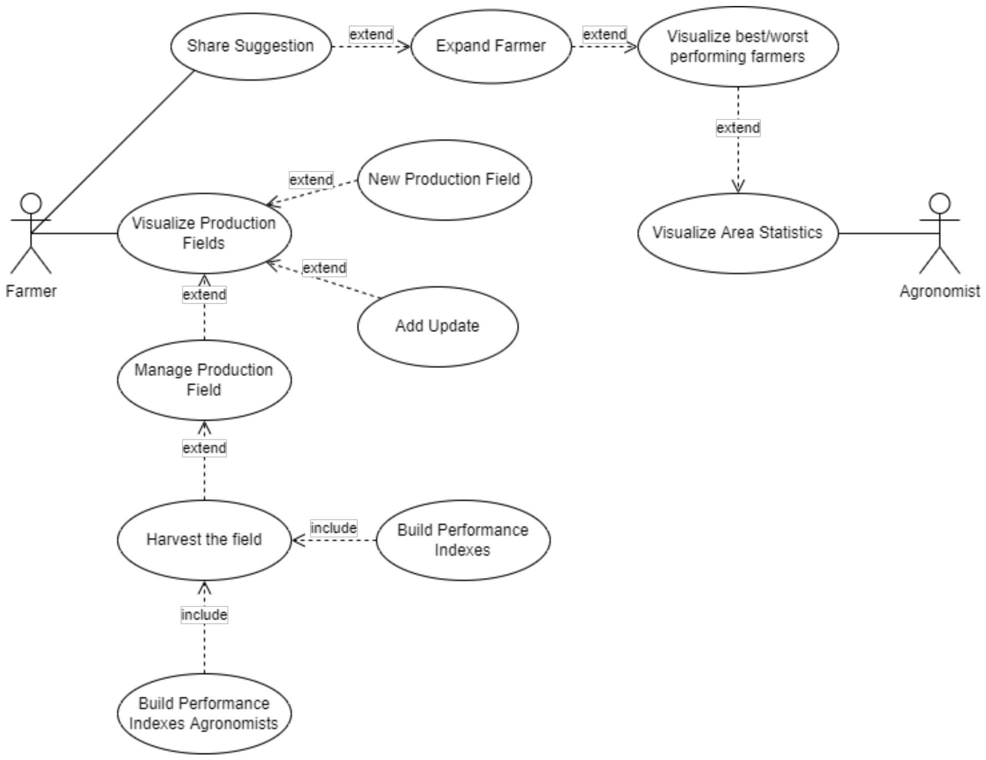


Figure 24: Production Area Use Case Diagram

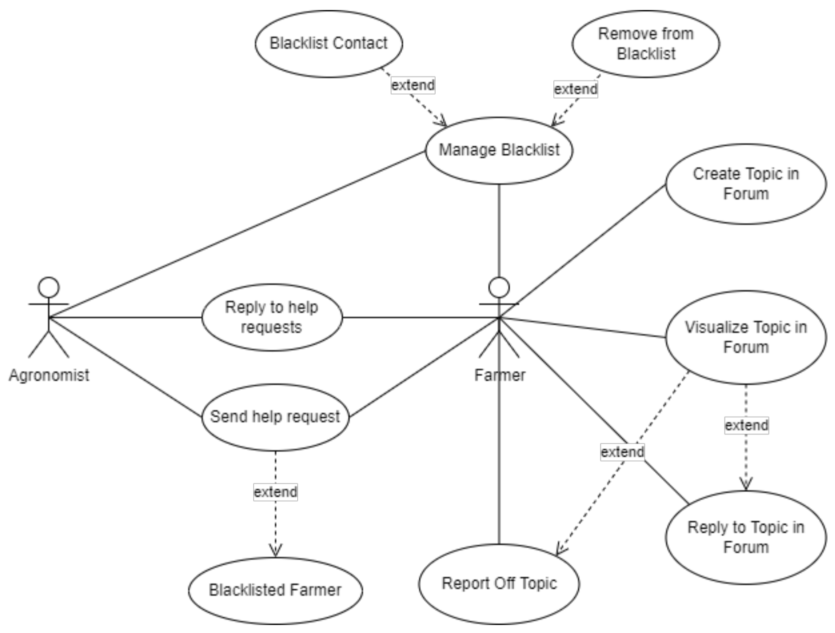


Figure 25: Farmer Communication Use Case Diagram

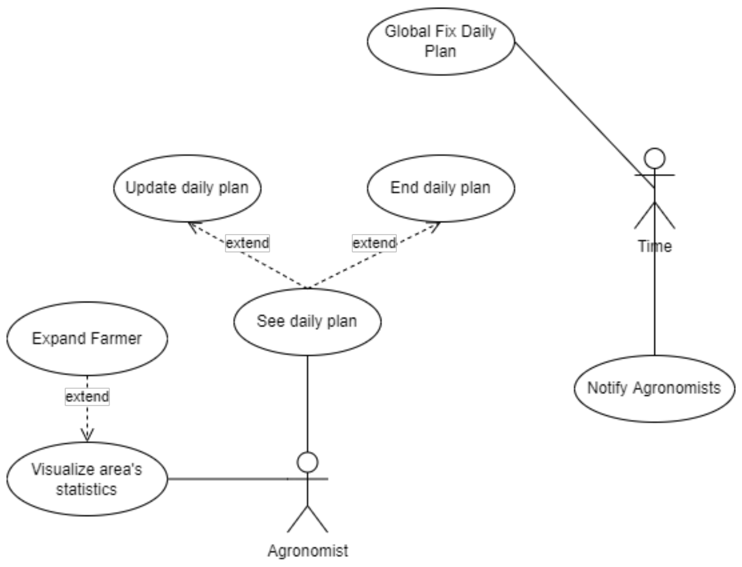


Figure 26: Agronomist Daily Plan Use Case Diagram

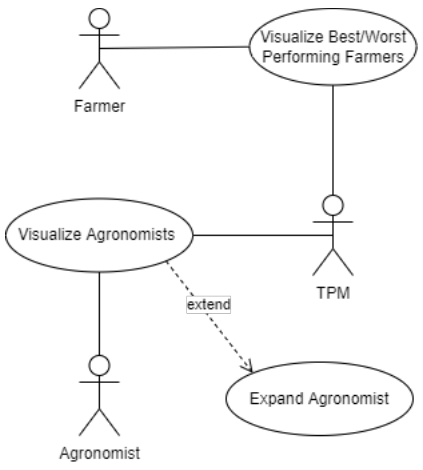


Figure 27: TPM Use Case Diagram

3.3.3 Use Cases Description

1. **Registration**

Name	**Registration**
Actors	Unauth. User
Entry conditions	The web application has started
Flow of events	(a) Unauth. User selects the "Register" option (b) The system asks the type of registration ("Farmer", "Agronomist", "Telangana Policy Maker") (c) Alternative flow A) i. Unauth. User selects "Agronomist" ii. The system asks for a valid Agronomist License iii. Unauth. User uploads his license iv. The system checks if the license is valid v. The system display the map of Telangana vi. Unauth. User inserts his area of interest (d) Alternative Flow B) i. Unauth. User selects "Telangana Policy Maker" ii. The system asks for a valid Telangana Policy Maker License iii. Unauth. User uploads his license iv. The system checks if the license is valid (e) Alternative Flow C) i. Unauth. User selects "Farmer" ii. The system display the map of Telangana iii. Unauth. User inserts his location (f) The system displays mandatory registration fields(i.e: name,password,foto...) useful for the creation of the new User (g) Unauth. User fills the mandatory fields and selects "Sign Up" (h) The system preprocesses input data and stores it into the database
Exit conditions	Unauth. User has now an User account and the User account contains valid data into the database according to its role.

Exceptions	• If an existing user with the same credentials exist, the system displays an error message to Unauth. User • If the preprocessed data in step 6 is not valid, an error message is displayed to Unauth. User • If the GPS Area Mapping system fails, an error message is displayed to Unauth. User • If the License is not valid, an error message is displayed to Unauth. User

Table 9: *User Registration* use case description

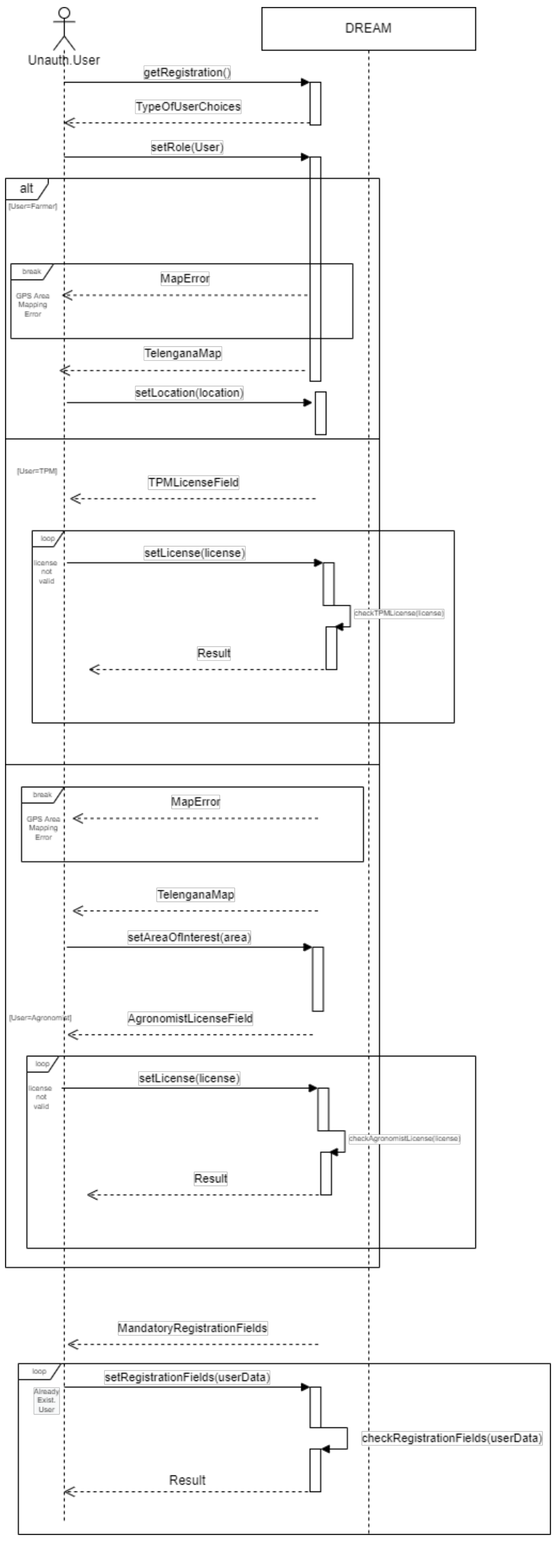


Figure 28: User Registration sequence diagram

2. **User Login**

Name	**User Login**
Actors	Unauth. User, User
Entry conditions	The web application has started
Flow of events	(a) Unauth. User selects the "Login" option (b) The system displays mandatory fields useful for the authentication of the Unauth. User (c) Unauth. User fills the fields and selects "Sign in" (d) The system checks the existence of such User (e) The system autenticates Unauth. User as User with his specific role
Exit conditions	Unauth. User has been autenticated as User and can proceed with the navigation in his homepage
Exceptions	• At any moment, the Unauth. User can stop the use case • If the User does not exists, the system displays an error message

Table 10: *User Login* use case description

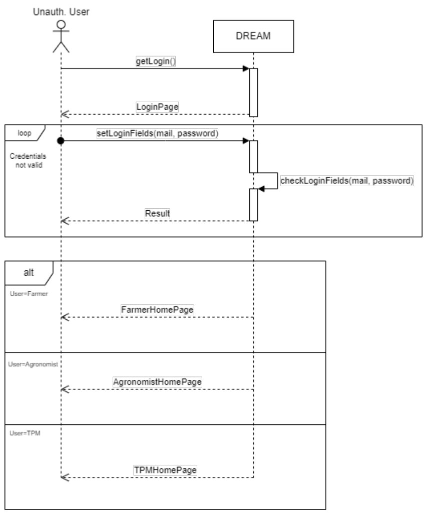


Figure 29: User Login sequence diagram

3. **New Production Field**

Name	**New Production Field**
Actors	Farmer
Entry conditions	(a) Farmer is logged in the application
	(b) Farmer visualizes the Home Page page
Flow of events	(a) Farmer selects "Insert" in Production window
	(b) The system displays mandatory fields to specify the type of production
	(c) Farmer types production data into mandatory fields and presses "Confirm"
	(d) The system preprocesses input data and stores it into the database
Exit conditions	The new production field is correctly stored
Exceptions	• Invalid Production Data Exception

Table 11: *Add Production Field* use case description

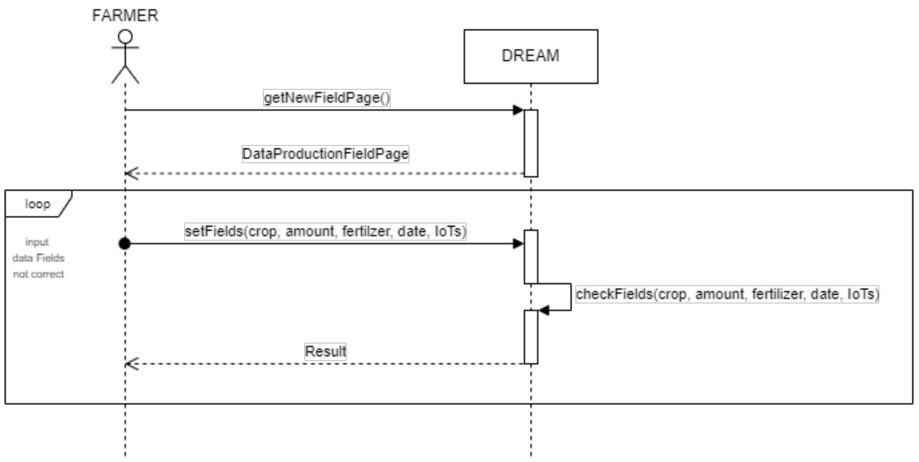

FARMER
DREAM
getNewFieldPage()
DataProductionFieldPage
loop
input data Fields not correct
setFields(crop, amount, fertilzer, date, IoTs)
checkFields(crop, amount, fertilizer, date, IoTs)
Result

Figure 30: Insert Data Production Field sequence diagram

4. **Manage Production Field**

Name	**Manage Production Field**
Actors	Farmer
Entry conditions	(a) Farmer must be authenticated (b) Farmer is visualizing his production fields
Flow of events	(a) Farmer selects a production field (b) The system expands the production field and the relative options (c) Alternative Flow A) i. Farmer selects "Edit Production Field" ii. The system let Farmer to edit the production field editable data (i.e. planted crops, type, fertilizer..) iii. Farmer confirms the edit iv. The system stores the modified production field in the database (d) Alternative Flow B) i. Farmer selects "Remove Production Field" ii. The system asks the farmer if the decision should be taken iii. Farmer confirms the decision iv. The system removes the production field from the database
Exit conditions	All the production fields have been managed
Exceptions	• Invalid Production Data Exception

Table 12: *Manage Production Field* use case description

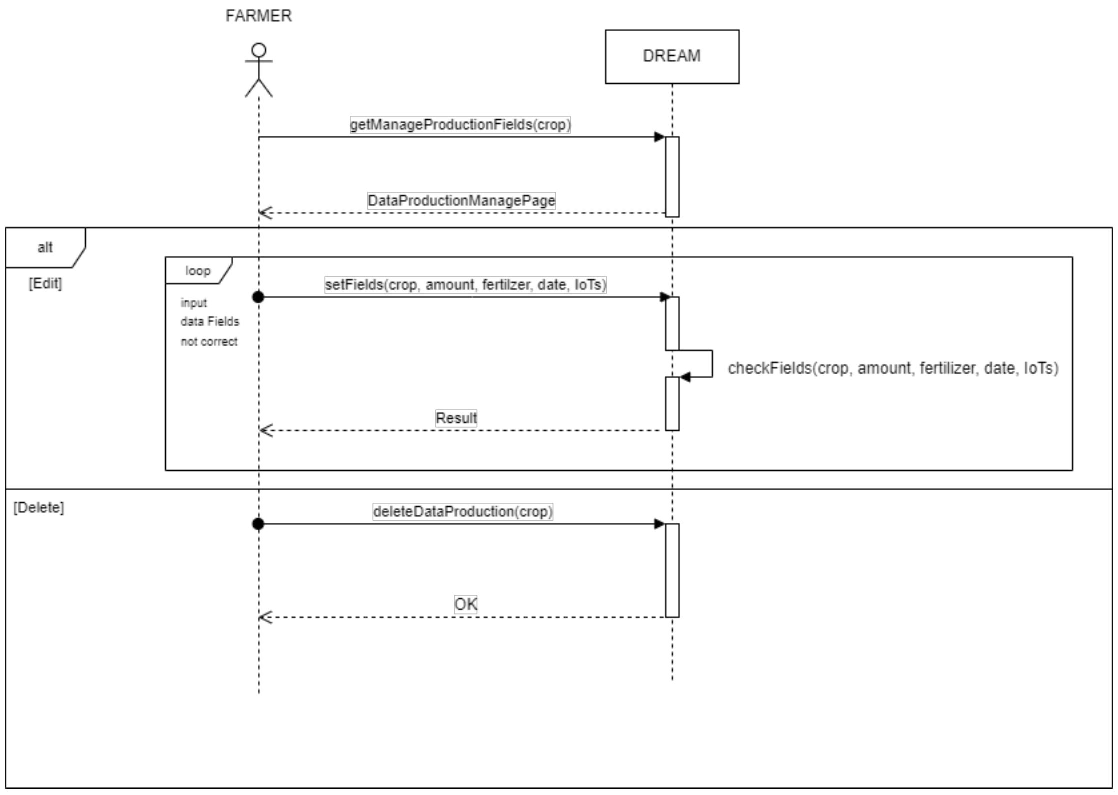

FARMER
DREAM
getManageProductionFields(crop)
DataProductionManagePage
alt
[Edit]
loop
input data Fields not correct
setFields(crop, amount, fertilzer, date, IoTs)
checkFields(crop, amount, fertilizer, date, IoTs)
Result
[Delete]
deleteDataProduction(crop)
OK

Figure 31: Manage Production Field sequence diagram

5. **Add Update**

Name	**Add Update**
Actors	Farmer
Entry conditions	(a) Farmer must be authenticated (b) Farmer is in the Home Page page

Flow of events	(a) Farmer selects "Add Update" option (b) The system displays fields to be filled to describe the problem/update or to insert a photo about that production field (c) Farmer fills the fields (d) Farmer presses "Confirm" (e) The system stores the problem/update in the database and binds it with the selected production field
Exit conditions	The update is correctly stored and bounded with the production field
Exceptions	• Format Photo errore exception

Table 13: *Add Update* use case description

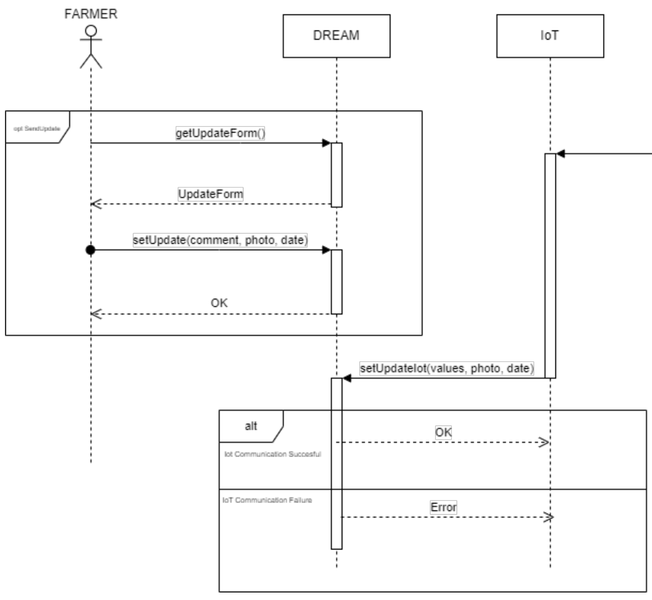

FARMER
DREAM
IoT
TIME
opt SendUpdate
getUpdateForm()
UpdateForm
setUpdate(comment, photo, date)
OK
1 week
setUpdateIot(values, photo, date)
alt
Iot Communication Succesful
OK
IoT Communication Failure
Error

Figure 32: Add Update sequence diagram

6. **Harvest the field**

Name	**Harvest the field**
Actors	Farmer
Entry conditions	• Farmer must be authenticated • Farmer is visualizing his production fields • Farmer is in the Visualize Relevant Data page
Flow of events	(a) Farmer selects "Harvest the field" option (b) The system asks the amount of crops harvested and the ending data (c) Farmer fills the field with a number (d) Farmer selects "Confirm" (e) The system removes the production field (f) The system stores the above mentioned popped information in the database and uses them to update the statistics
Exit conditions	The production field is removed now. Statistics have been updated according to the pair (crop planted, crops harvested)
Exceptions	Format value inserted not accepted

Table 14: *Harvest the field* use case description

7. **Build Performance Indexes**

Name	**Build Performance Indexes**
Actors	Farmer
Entry conditions	• Farmer is logged in the application
Flow of events	(a) The Farmer starts the use case when he harvests a production field (b) The system then builds and updates statistics and performance indexes about the Farmer and the updated Production Field

Exit conditions	New statistics are built according to the harvested production
Exceptions	

Table 15: *Build Performance Indexes* use case description

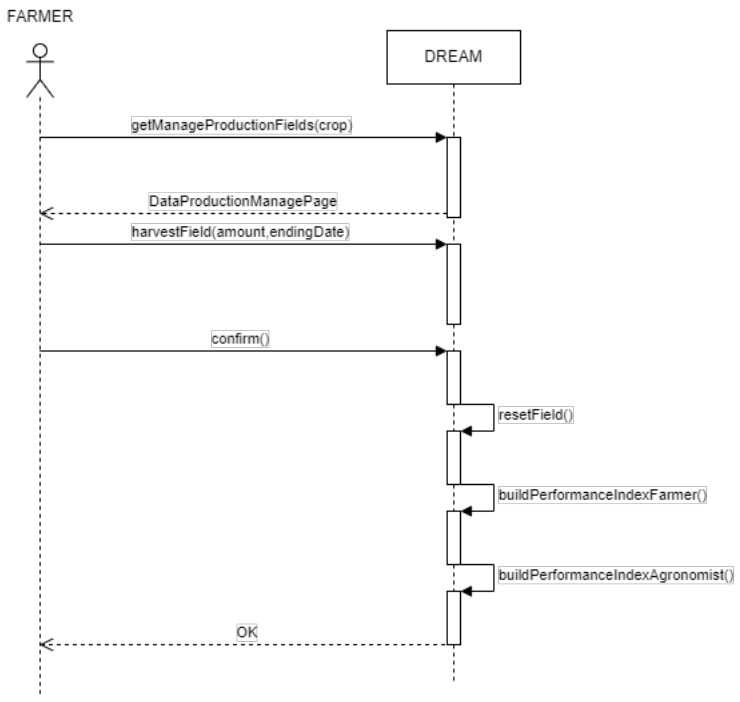

FARMER
DREAM
getManageProductionFields(crop)
DataProductionManagePage
harvestField(amount,endingDate)
confirm()
resetField()
buildPerformanceIndexFarmer()
buildPerformanceIndexAgronomist()
OK

Figure 33: Harvest field sequence diagram

8. **Send help request**

Name	**Send help request**
Actors	Farmer A, Farmer B, Agronomist
Entry conditions	Farmer A is logged in the application
Flow of events	(a) Farmer A selects "Send help request" on his homepage (b) The system displays email-like mandatory fields (c) Farmer A inputs a description about the help he needs (d) Farmer A specifies other mandatory fields (e) Farmer A specifies in particular the username of the Farmer A/Agronomist he wants to contact privately (f) Farmer A presses "Send" (g) The system stores the private message in the database and forwards a notification to the addressee
Exit conditions	The inbox of the contacted Farmer B/Agronomist contains now the help request coming from the Farmer A
Exceptions	• Blacklisted Farmer Exception in step f) • Inexistent Contact Exception in step f) –> The addressee does not exists. The system displays an error message.

Table 16: *Send help request* use case description

9. **Blacklisted Farmer**

Name	**Blacklisted Farmer**
Actors	Farmer A, Farmer B, Agronomist
Entry conditions	• Farmer A is logged in the application • Farmer A just pressed "Send" button to send help request to Farmer B/Agronomist

Flow of events	(a) Farmer is blacklisted by the addressee Farmer B/Agronomist (b) The system displays an error message and a warning to Farmer A, inviting him to stop spamming
Exit conditions	
Exceptions	

Table 17: *Blacklisted Farmer* use case description

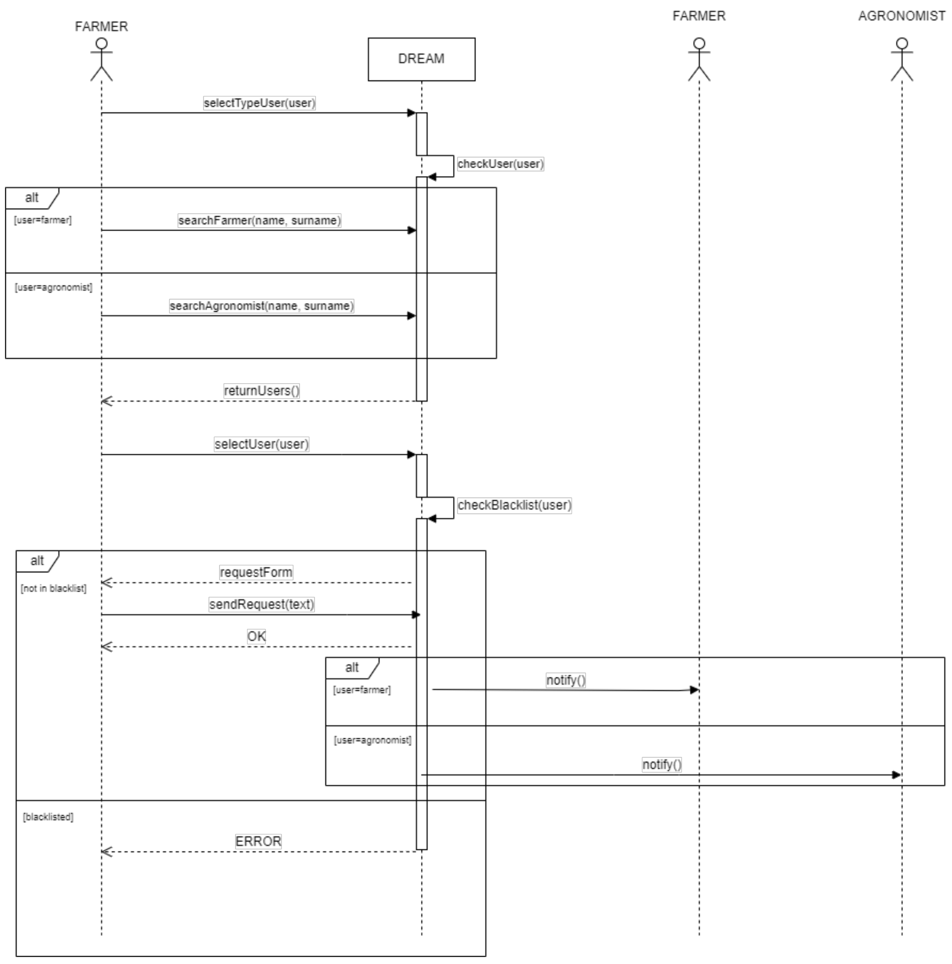


Figure 34: Send Request sequence diagram

10. **Reply to help requests**

Name	**Reply to help requests**
Actors	Farmer A, Agronomist, Farmer B
Entry conditions	Farmer A/Agronomist is logged in the application
Flow of events	(a) Farmer A/Agronomist selects "Check help requests" on his homepage (b) The system lists all help requests for that Farmer A/Agronomist (c) Farmer A/Agronomist selects one among them (d) The system displays the content (e) Farmer A/Agronomist selects "Reply" (f) The system shows email-like mandatory fields (g) Farmer A/Agronomist inputs data and the reply to the help request (h) Farmer A presses "Send" (i) The system stores in the database the reply
Exit conditions	The inbox of the contacted Farmer B contains now the reply to his help request
Exceptions	

Table 18: *Reply to help requests* use case description

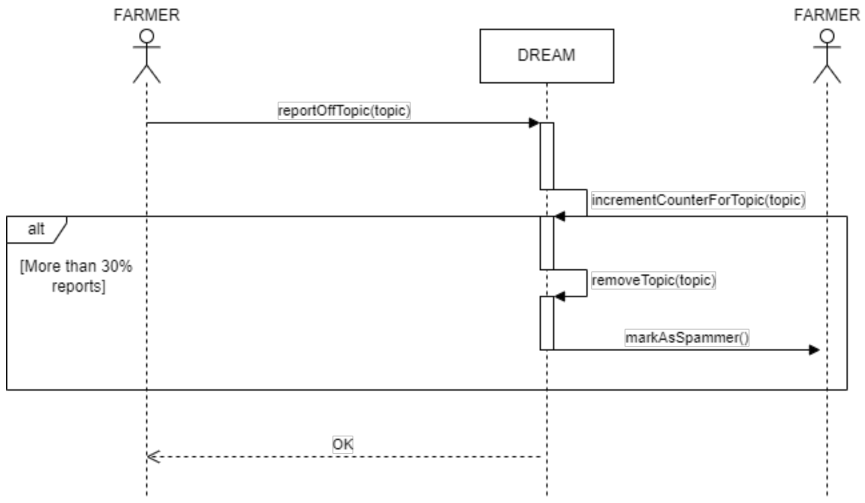


Figure 35: Reply to help requests sequence diagram

11. **Blacklist Contact**

Name	**Blacklist Contact**
Actors	Farmer, Agronomist
Entry conditions	• Farmer/Agronomist is logged in the application • Farmer/Agronomist visualizes his blacklist
Flow of events	(a) Farmer/Agronomist inputs the name/surname of a contact in the search (b) The system displays the list of retrieved contacts (c) Farmer/Agronomist selects a contact (d) Farmer/Agronomist presses the button "Add" (e) The system stores the choice on the database and marks the selected contact as blacklisted for the Farmer/Agronomist
Exit conditions	The selected contact is blacklisted
Exceptions	

Table 19: *Blacklist Contact* use case description

12. **Remove from Blacklist**

Name	**Remove from Blacklist**
Actors	Farmer, Agronomist
Entry conditions	• Farmer/Agronomist is logged in the application • Farmer/Agronomist visualizes his blacklist
Flow of events	(a) Farmer/Agronomist expanded a contact in the Blacklist page (b) Farmer selected "Remove Contact" (c) The system stores the choice on the database and unmarks the blacklisted user
Exit conditions	The list is shown to the authority
Exceptions	The selected contact is no more in the blacklist

Table 20: *Check unread reports* use case description

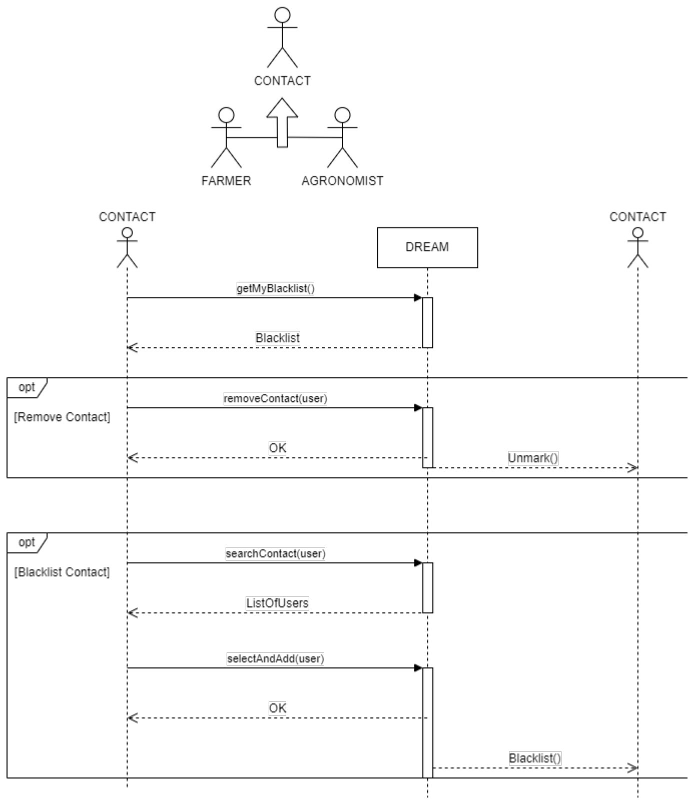


Figure 36: Manage Blacklist sequence diagram

13. **Create Topic in Forum**

Name	**Create Topic in Forum**
Actors	Farmer
Entry conditions	Farmer is logged in the application
Flow of events	(a) Farmer selects "Forum" on his homepage (b) The system displays mandatory fields about the type of discussion forum to be created (c) Farmer specifies all the information and inserts a first post (d) Farmer presses "Create" (e) The system checks if the last topic created by the farmer is not up to the current day (f) The system creates the discussion forum and stores it into the database
Exit conditions	The new Topic is visible now
Exceptions	(a) Daily Topics Limit Reached Exception in e) –> Only one topic per day is allowed, so the daily topics limit has been reached by the current farmer; an error message is displayed and the use case stops

Table 21: *Create Topic in Forum* use case description

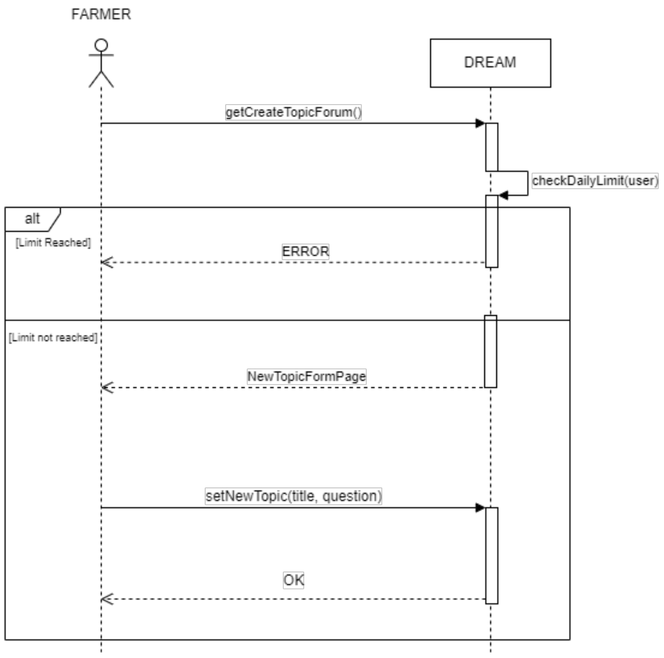


Figure 37: Create Topic in Forum sequence diagram

14. **Visualize Topic in Forum**

Name	**Visualize Topic in Forum**
Actors	Farmer
Entry conditions	Farmer is logged in the application
Flow of events	(a) Farmer selects "Forum" on his homepage (b) The system displays the lists of topics sorted in chronological order (c) Farmer selects a topic (d) The system displays the comments of that topic
Exit conditions	The topic selected with all comments is visible now
Exceptions	

Table 22: *Visualize Topic in Forum* use case description

15. **Reply to topic in Forum**

Name	**Reply to topic in Forum**

Actors	Farmer
Entry conditions	(a) Farmer is logged in the application (b) Farmer visualizes a topic
Flow of events	(a) Farmer selects "Reply" (b) The system displays a field for the reply (c) Farmer inputs the reply and presses "Send" (d) The system stores the reply in the database and appends it to the conversation
Exit conditions	Farmer's reply is shown in the topic
Exceptions	

Table 23: *Reply to topic in Forum* use case description

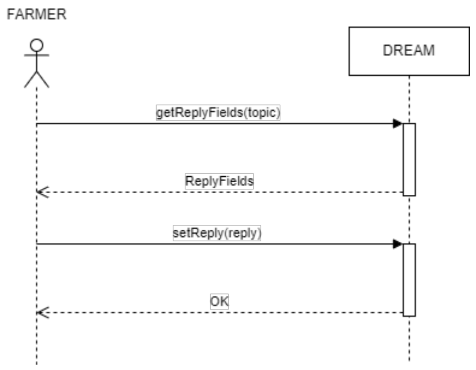


Figure 38: Create Topic in Forum sequence diagram

16. **Report Off topic**

Name	**Report Off topic**
Actors	Farmer A, Farmer B
Entry conditions	(a) Farmer A is logged in the application (b) Farmer A visualizes a discussion forum (c) Another Farmer (Farmer B) goes off topic

Flow of events	(a) Farmer A selects "Report Off Topic" for a specific post of Farmer B (b) The system stores the report in the database (c) The system counts how many reports for that specific post/Farmer B have been sent (d) Alternative Flow B) i. The system counts more than 30 percent of active farmers reports for the post ii. The system removes the off topic post and marks Farmer B as "Spammer" for that forum
Exit conditions	Farmer A has succefully reported Farmer B's off topic post and Farmer B's off topic post is correctly managed by the system
Exceptions	

Table 24: *Report Off topic* use case description

17. **See daily plan**

Name	**See daily plan**
Actors	Agronomist
Entry conditions	Agronomist is logged in the application
Flow of events	(a) Farmer selects "See daily plan" on his homepage (b) The system displays the daily plan to the Agronomist.
Exit conditions	The Agronomist is now able to see the daily plan
Exceptions	

Table 25: *See daily plan* use case description

18. **Fix daily plan**

Name	**Fix daily plan**
Actors	Time
Entry conditions	

Flow of events	(a) Time starts the use case when it becomes 0.00 for the current day (b) For all agronomists the system marks the daily plan which has a temporal window of a month as confirmed and makes it immutable (c) For all agronomists the system creates a new daily plan for the next day
Exit conditions	For all agronomists the daily plan is immutable now for each day a new daily plan is created.
Exceptions	

Table 26: *Fix daily plan* use case description

19. **Update daily plan**

Name	**Update daily plan**
Actors	Agronomist
Entry conditions	(a) Agronomist is logged in the application (b) Agronomist sees his daily plan in the application (c) The daily plan is not fixed
Flow of events	(a) Alternative Flow A) i. Agronomist selects "Add visit" ii. The system displays editable fields (i.e farmer, hour) about the new visit iii. Agronomist updates the fields and selects "Finish" iv. The system stores the visit in the database (b) Alternative Flow B) i. Agronomist selects "Delete visit" ii. Agronomist selects the visits to be deleted iii. Agronomist selects "Finish" iv. The system deletes the visits in the database
Exit conditions	The daily plan is updated in the database
Exceptions	

Table 27: *Update daily plan* use case description

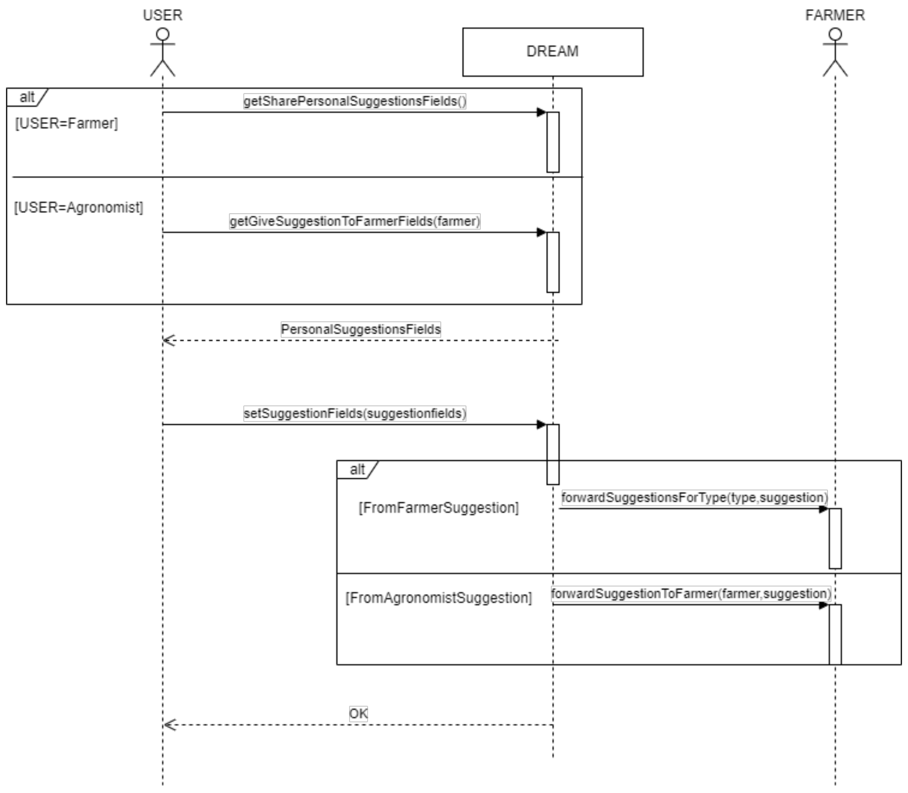

USER
DREAM
FARMER
alt
[USER=Farmer]
getSharePersonalSuggestionsFields()
[USER=Agronomist]
getGiveSuggestionToFarmerFields(farmer)
PersonalSuggestionsFields
setSuggestionFields(suggestionfields)
alt
[FromFarmerSuggestion]
forwardSuggestionsForType(type,suggestion)
[FromAgronomistSuggestion]
forwardSuggestionToFarmer(farmer,suggestion)
OK

Figure 39: Report Off topic sequence diagram

20. **End daily plan**

Name	**End daily plan**
Actors	Agronomist, Time
Entry conditions	(a) Agronomist is logged in the application (b) Agronomist sees his daily plan in the application (c) The daily plan is not fixed
Flow of events	21. Agronomist selects "Confirm daily plan" 22. The system marks the daily plan as confirmed and makes it immutable
Exit conditions	The daily plan is immutable now
Exceptions	

Table 28: *Update daily plan* use case description

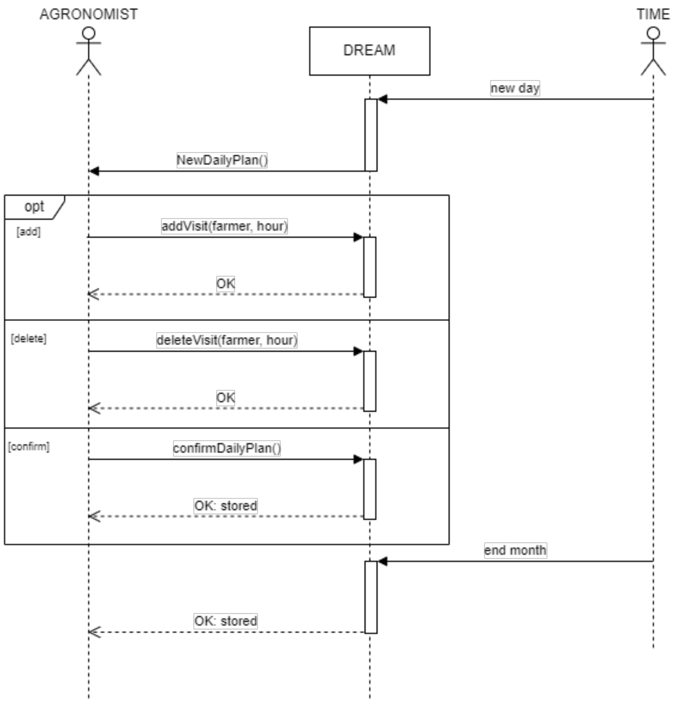


Figure 40: Manage Daily Plan sequence diagram

23. **Notify Agronomists**

Name	**Notify Agronomists**
Actors	Time, Agronomist
Entry conditions	
Flow of events	(a) Time starts the use case when it ends the last day of the current month (b) The system retrieves all the farmers which has not been visited at least twice and their location (c) The system retrieves all the agronomists related to the retrieved locations (d) The system notifies the aforesaid agronomists about the previously retrieved farmers, as they need to be visited
Exit conditions	All the agronomists have been notified
Exceptions	

Table 29: *Notify Agronomists* use case description

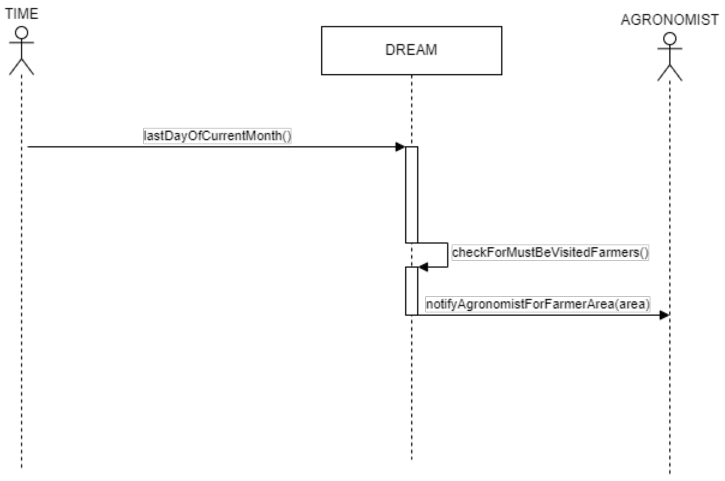

TIME
DREAM
AGRONOMIST
lastDayOfCurrentMonth()
checkForMustBeVisitedFarmers()
notifyAgronomistForFarmerArea(area)

Figure 41: Notify Agronomists sequence diagram

24. **Share Suggestion**

Name	**Share Suggestion**
Actors	Agronomist, Farmer A, Farmer B

Entry conditions	(a) The appropriate user (which is one among Agronomist or Farmer A) is authenticated (b) If Agronomist is authenticated then he has expanded Farmer B's view
Flow of events	25. Farmer A selects "Share Personal Suggestion" 26. Alterantive Flow A) (a) Agronomist selects "Give Suggestion" 27. The system displays useful forms for the suggestion (e.g target type of production...) and a textarea 28. Agronomist/Farmer A fills the textarea with useful suggestions about a particular production field or more general tips 29. Agronomist/Farmer A selects "Publish Suggestion" 30. The system stores in the database the suggestion 31. For each Farmer B that owns a production field targeted by the suggestion based on specific parameters (type of production..) the system forwards to him the personalized suggestion 32. Alternative flow A) (a) The system forwards to Farmer B (whose view was expanded by the Agronomist) the suggestion of the Agronomist
Exit conditions	(a) Farmer A's suggestion is now forwarded and visible to all Farmer Bs that own a production field targeted by the suggestion (b) Agronomist's suggestion is now forwarded and visible to the expanded Farmer B.
Exceptions	

Table 30: *Share Suggestion* use case description

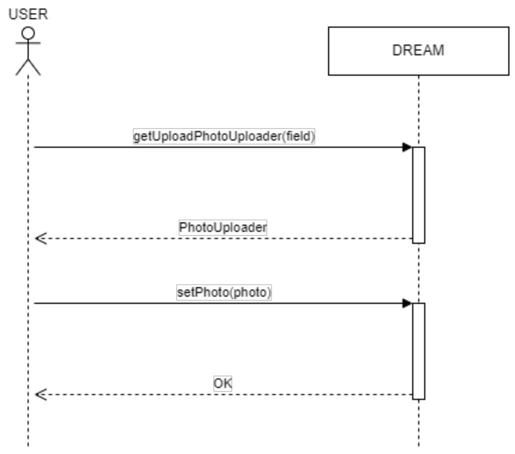


Figure 42: Share Suggestion sequence diagram

33. **Visualize Area Statistics**

Name	**Visualize Area Statistics**
Actors	Agronomist
Entry conditions	(a) Agronomist is logged in the application (b) The database contains the area of interest of the Agronomist
Flow of events	34. Agronomist selects "Visualize Area Statistics" 35. The system displays all the area statistics to the Agronomist
Exit conditions	Area Statistics now visible
Exceptions	

Table 31: *Visualize Area Statistics* use case description

36. **Visualize best/worst performing farmers**

Name	**Visualize best/worst performing farmers**
Actors	Agronomist,TPM

Entry conditions	(a) The appropriate user (which is one among Agronomist or TPM) is authenticated (b) If Agronomist is authenticated then he has selected his area of interest
Flow of events	37. TPM selects "Visualize best performing farmers" 38. Alternative flow A) (a) Agronomist selects "Visualize best/worst performing farmers" (b) The system restricts the displayable farmers to those working in the area of interested of the Agronomist 39. The system displays the list of displayable farmers and their relative performances
Exit conditions	Performance of the farmers are now visible
Exceptions	

Table 32: *Visualize Area Statistics* use case description

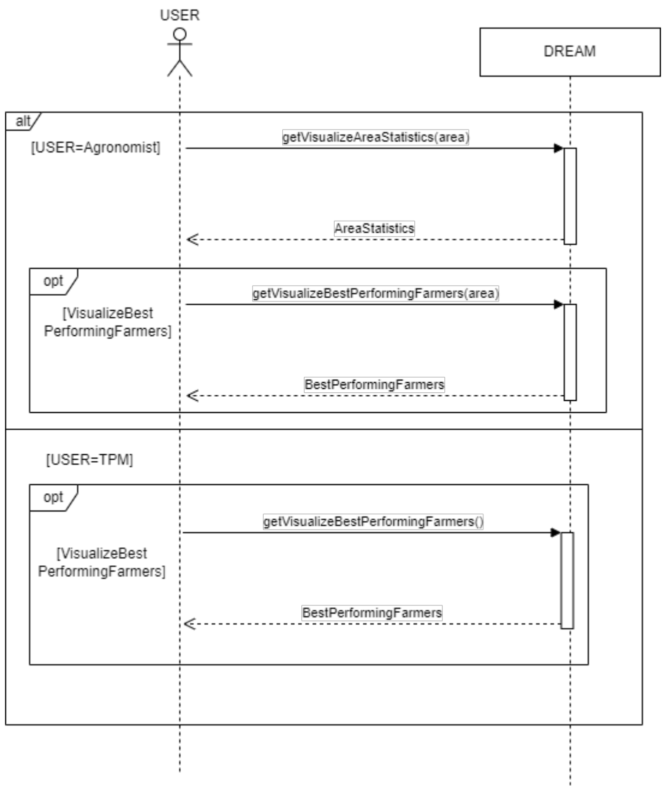

USER
DREAM
alt
[USER=Agronomist]
getVisualizeAreaStatistics(area)
AreaStatistics
opt
[VisualizeBest PerformingFarmers]
getVisualizeBestPerformingFarmers(area)
BestPerformingFarmers
[USER=TPM]
opt
[VisualizeBest PerformingFarmers]
getVisualizeBestPerformingFarmers()
BestPerformingFarmers

Figure 43: Area of General Statistics sequence diagram

40. **Visualize Agronomists**

Name	**Visualize Agronomists**
Actors	Agronomist,TPM
Entry conditions	TPM is authenticated
Flow of events	41. TPM selects "Visualize Agronomists" 42. The system displays a list of agronomists and their working area
Exit conditions	Agronomists are now visible
Exceptions	

Table 33: *Visualize Agronomists* use case description

43. **Visualize Relevant Data**

Name	**Visualize Relevant Data**
Actors	Farmer
Entry conditions	(a) Farmer is logged in the application (b) Farmer defined in the system his location and his production (c) Farmer selected a field in "Manage Production Fields" page
Flow of events	44. Farmer selects "Visualize relevant data 45. The system displays data relevant to the type of production based on the selected field 46. The system displays weather forecasts 47. The system displays personalized suggestions about the selected production
Exit conditions	Data relevant to production fields are now visible
Exceptions	

Table 34: *Visualize Relevant Data* use case description

48. **Visualize Production Fields**

Name	**Visualize Production Fields**

Actors	Farmer
Entry conditions	Farmer must be authenticated
Flow of events	49. Farmer selects "Visualize Production Fields" 50. The system displays all the production fields of the Farmer
Exit conditions	The list of all production fields are now visible
Exceptions	

Table 35: *Visualize Production Fields* use case description

51. **Expand Agronomist**

Name	**Expand Agronomist**
Actors	TPM
Entry conditions	(a) TPM is logged in the application (b) TPM sees the "Visualize Agronomists" page
Flow of events	52. TPM selects an agronomist among listed ones 53. The system shows statistics and helped farmers for the selected agronomist
Exit conditions	The data and the statistics relevant to an agronomist is now visible
Exceptions	

Table 36: *Expand Agronomist* use case description

54. **Expand Farmer**

Name	**Expand Farmer**
Actors	Agronomist
Entry conditions	(a) Agronomist is logged in the application (b) Agronomist sees the "Visualize Area's Statistics" page

Flow of events	55. The Agronomist select a farmer among listed ones 56. The system shows statistics about the selected farmer 57. The system shows all the fields own by the selected farmer
Exit conditions	All the statistics of the selected farmer are now visible
Exceptions	

Table 37: *Expand Agronomist* use case description

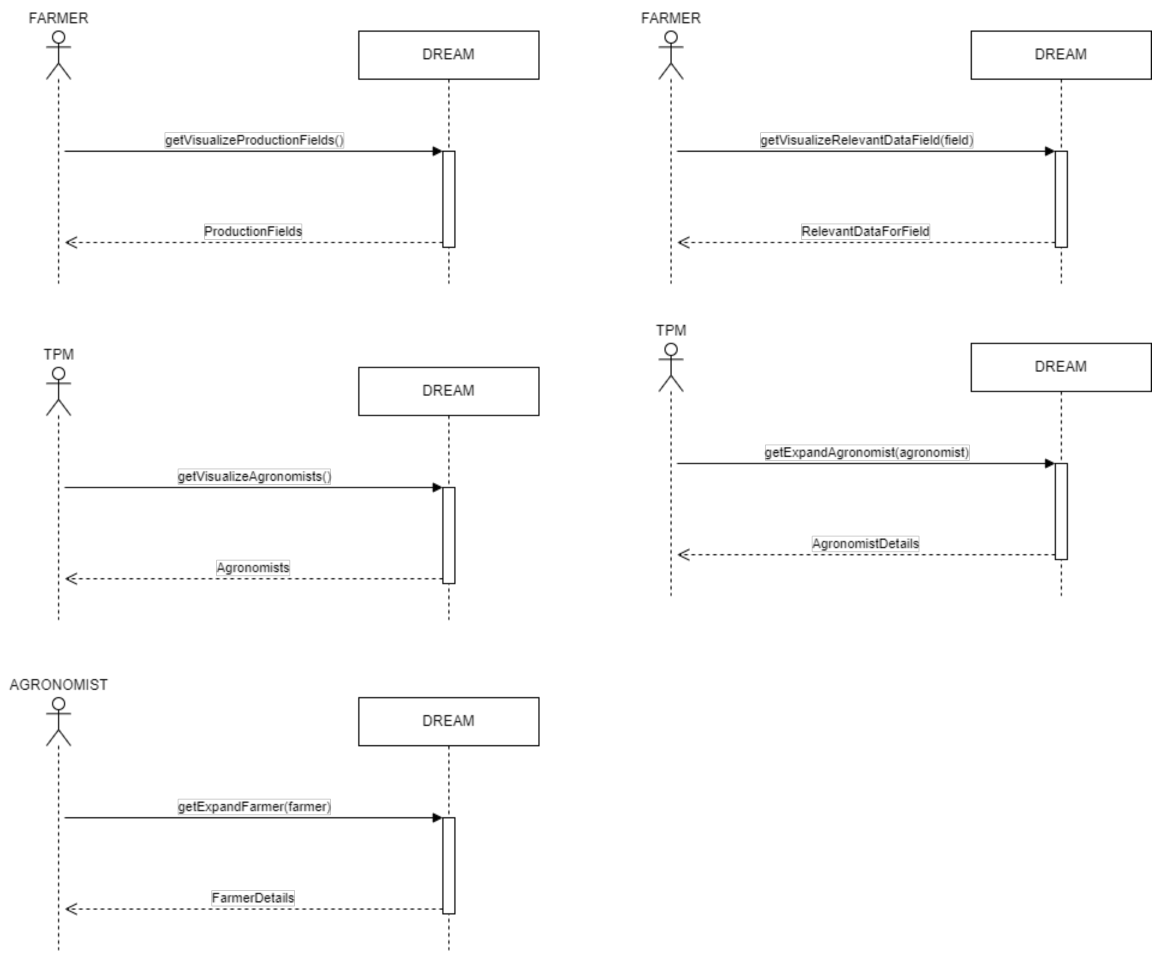


Figure 44: Other Visualizations sequence diagram

3.4 Performance Requirements

3.5 Product Requirments - Usability

One of the main usability factors of DREAM is the ease of learning. As the application will be (potentially) used by old-school farmers and agronomists, their experience facing the features of the system should be easy and of simple use; it means: understandable functions and terminology, clear interpretation of graphs, etc.

3.6 Product Requirments - Efficiency

As the application will support the interaction of three type of users who in turn can use simultaneously the application in a great number, DREAM must be able to manage the pool of incoming connections and reply to them properly. Since the system does not aim to solve critical (i.e. life or death) functions, it is more a matter of space rather than response time. So, the system must be able to store a large quantity of data even for each user, as a Farmer is associated with graph(-s), production fields, discussion topics and inbox.

3.7 Design Constraints

3.7.1 Standards Compliance

Formats: rainfall parameter in meteorological data is expressed in millimeters (mm). Agronomist and Telengana's Policy Maker licenses must be in pdf format and they must comply with standars for legal licenses. Photos must be in png, jpg or bmp format.

Legal Aspects: since DREAM supports personal data sharing, private and public messaging, its functions must be subject to the Harmful Digital Communication Act. Furthermore, privacy of data has to be guaranteed, so the system must follow the rules of the General Data Protection Regulations (GDPR).

3.7.2 Hardware Limitations

As specified in the Hardware Interface sections, the main hardware limitations are:

1. Connection to internet
2. Properly working sensors for monitoring soil state and water irrigation (only if specified by the Farmer)
3. Properly working cameras in sensors for taking photos of the field or a device able to take pictures (only if specified by the Farmer)

3.7.3 Any other Constraint

Discussion Forums Constraints: a Farmer is limited to only 1 topic a day. A topic that receives off-topic reports from the 30 percent or more of active farmers in the discussion forum will be removed.

Inbox Constraints: inbox of Farmer and Agronomist can contain 100 requests max received or sent.

Production Fields Constraints: a Farmer is limited to 30 Production Fields.

3.8 Software System Attributes

3.8.1 Reliability

A crucial aspect of DREAM is the reliable store of important data. A Farmer should not met the risk of losing data about his production field, as it will damage him (who trusts DREAM) and Agronomists who choose to help him (who see unconsistent data and then elaborate inefficient strategies); Telengana's Policy Makers too can be affected by the problem, as the performance indexes built on missing/erroneous data of a Farmer can lead to misreadings/misunderstandings of the above mentioned actors. To avoid this risks, fault tolerance should be the main reliable requiremet of DREAM, and can be ensured by redundant data (for example by implementing RAID 4 disks with parity bit to retrieve lost data) and reducing to minimum the MTTDL.

3.8.2 Availability

As mentioned in the Efficiency section, DREAM does not need critical response times or so, however it has to ensure that no one of the simultaneously connected users/sessions will suffer of starvation, hence every istance currently connected to the system should be equally served with a proper response time.

3.8.3 Security

DREAM functionalities are about storing sensitive informations about the user itself and (in the case of the Farmer) his production. As privacy is an important aspect of the system, the database must be able to implement access control methods to avoid unauthorized users to extract private informations. Since DREAM runs on browser, countermeasures againsts XSS and SQL injections attacks must be taken; this means implement prepared statementes whenever a query is called, preprocess all the input data to avoid Stored XSS, Reflected XSS and DOM XSS; cross-site-reforgery-tokens must be implemented whenever POST request are triggered (i.e. on update/creation of production fields, or on sending private messages). Sessions and data sent by the user must be encrypted with SHA-256 or better. Last but not least, the site has to own a valid and up do date SSL Certificate.

3.8.4 Maintainability

The system must be developed in a way that future maintenance operations can be performed in an easy way. This means: clear and understandable bounds among components and tiers, well commented code, reuse of code.

3.8.5 Portability

DREAM runs on browsers, so the most important web engines like Internet Explorer, Chrome, Firefox, Safari, Opera, and others must be able to render pages in a common or, if not, understandable fashion. If the website is accessed by Android browsers, the user must be able to see them in a resized and proper way

3.8.6 Scalability

As mentioned before, DREAM must be able to offer a connection pool and a storing space to a large quantity of users for three different type of actors.

4 Formal Analysis Using Alloy

The following section considers the essential properties and constraints identified for the specification of the problem and provides a formal model in which it is shown how they will be satisfied. The alloy modeling language is used to model the problem, and some possible worlds are also provided in order to clarify the most critical aspects. Before to read the alloy model, it is important to keep in mind that TPM actor is not considered in the worlds shown because they have only the functionality of visualizing and monitor farmers and agronomists.

4.1 Alloy Model

```
// A Farmer can be associated to multiple Production Fields
    -----
// A Production Field has one and only one crop -----
// Only farmers participate to forum; at least one farmer
    per forum ------
// Only one user with a determined username exists ---------
// One Account for one user ----------
// Agronomist and TPM must be associated to one and only one
     license, and viceversa -------
// All Password with at least one Account -------
// Agronomist to one Area, an Area involves multiple
    location; One Farmer one Location, and viceversa -----
// Location are relative to only one Area; two locations
    have different areas --------
// All areas are covered by at least one agronomist
    --------
// All production fields have one and only one owner/farmer
    --------
// Technical Fact: Dates D1 and D2 are equal iff they share
    the same parameters



//------Authentication Part-------//

sig Email{}

sig Password{}

sig Account{
  mail: one Email,
  pass: one Password
}

abstract sig User{
  registration: one Account
}

//------Farmer Production And Area Part------//

sig Location{}

sig Area{
  locs: some Location
}

abstract sig Suggestion{}

sig FarmerSuggestion extends Suggestion{}

sig DirectSuggestion extends Suggestion {
  direct: one Farmer
}

sig Crop{
  suggestions: set Suggestion
}

sig Number{}
sig Day{}
sig Month{}
sig Year{}
sig Date{
  day: one Day,
  month: one Month,
  year: one Year
}
sig Fertilizer{}
sig Iot{}
sig Feedback{}

sig ProductionField{
  crop: one Crop,
  amount: one Number,
  date: one Date,
  fert: lone Fertilizer,
  iot: set Iot,
  updates: set Feedback
}

sig HarvestedField extends ProductionField{
  harvested: one Number,
  finalDate: one Date
}

sig Farmer extends User{
  location: one Location,
  fields: set ProductionField,
  messageRequest: set Request,
  messageResponseF: set Request,
  author: set Topic,
  comment: set Topic,
  suggest: set FarmerSuggestion
}

```

```
sig Agronomist extends User{
  areaofInterest: one Area,
  legalAgronomistLicense: one AgronomistLicense,
  dailyPlans: some DailyPlan,
  messageResponseA: set Request,
  agrosuggest: set DirectSuggestion
}

sig TPM extends User{
  legalTPMLicense: one TPMLicense
}

//------Daily Plan------//

sig DailyPlan{
  dateDailyPlan: one Date,
  visitedFarmers: set Farmer
}

//------Licenses Part-------//

abstract sig License{}

sig AgronomistLicense extends License{}

sig TPMLicense extends License{}

//----Request----/

sig Request{}

//------Forum------//

sig Topic {
  forum: one Forum
}

one sig Forum{}

//------Facts------//

fact DifferentDateInHarvestField{
  all h:HarvestedField | h.finalDate != h.date
}

fact OneDirectSuggestionFromOneAgro {     //All directs
    suggestions are written by only one agronomist
  all d:DirectSuggestion | one a:Agronomist | d in a.
    agrosuggest
}

fact OneSuggestionOneFarmer{          //All farmer
    suggestions are written by only one Farmer
  all s:FarmerSuggestion | one f:Farmer | s in f.suggest
```

```
}

fact UpdatesOneProductionField{          //All updates are
    related to one production field
  all f:Feedback| one p:ProductionField | f in p.updates
}

fact AllSuggestionsOneCrop{          //All suggestions are
    related to one crop
  all s:Suggestion | one c:Crop | s in c.suggestions
}

fact AllFertilizerAssociated {
  all f:Fertilizer, p:ProductionField | f in p.fert
}

fact DifferentDatesDifferentTimestamp {            //There
    exists oly different dates
  all da1, da2:Date | da1.day=da2.day and da1.month=da2.
    month and da1.year=da2.year implies da1=da2
}

fact AccountOneMail {
  all e:Email | #(mail.e)=1   // There can't be two or more
    accounts with the same email,  and all emails are
    associated to
              // an account
}

fact AccountOneUser {
  all a:Account | #(registration.a)=1   // All User have one
     and only one account and there can't be two users with
    the same account

}

fact AllPassWithAtLeastOneAccount {
  all p: Password | some a: Account | p in a. pass // All
    Password must be associated with at least one Account
}


fact TPMOneLicense {
  all l:TPMLicense | #(legalTPMLicense.l)=1   // One and
    only one license for TPM.
                          // Every license has a different
    owner.
}

fact AgronomistOneLicense {
  all l:AgronomistLicense | #(legalAgronomistLicense.l)=1
    // One and only one license for Agronomist.
                              // Every license has a
    different owner.
```

```
}

fact LocationOneArea {
  all l:Location | #(locs.l)=1   // Two different locations
    are relative to different Areas
}

fact LocationOneFarmer {
  all l:Location | #(location.l)=1   // Two different
    locations are relative to different Farmers
}

fact AllAreasCoveredByAtLeastOneAgronomist {
  all a:Area | #(areaofInterest.a)>=1   // All areas are
    covered by at least one agronomist
}

fact FieldsForDifferentFarmers{
  all p:ProductionField | #(fields.p)=1   // All production
    fields have one and only one owner/farmer
}

fact IotForProductionFields{
  all i:Iot | one p:ProductionField | i in p.iot
}

fact CropInProductionFields{
  all c:Crop | some p:ProductionField | c in p.crop
}

fact DailyPlansdifferentdates{
  all d1,d2:DailyPlan | (d1.dateDailyPlan = d2.dateDailyPlan
    ) implies d1 = d2
}

fact DailyPlanToOneAgronomist{            //All Daily Plan
    to Exactly One Agronomist
  all d:DailyPlan | one a:Agronomist | d in a.dailyPlans
}

//----At Least Twice a Year-----///

fun DateAndYearInWhichFarmerHasBeenVisited[f: Farmer]: set
    year {
   ((visitedFarmers.f).dateDailyPlan) <: year
}

pred VisitedTwiceAYear[f:Farmer]  {
  all y:Year | #(DateAndYearInWhichFarmerHasBeenVisited[f].y
    )>=2
}

//----Forum----//

```

```
fact OneTopicOneAuthor{
  all t:Topic | one f:Farmer | t in f.author
}

fact NotOnlyAuthorComment{
  all t:Topic, f:Farmer | (t in f.author) and (t in f.
    comment) implies #(comment.t) >= 1
}

//----Request----//

fact OneRequestResponsebyFarmerorAgronomist{       //The
    response can't received by both a farmer and an
    agronomist
  no r:Request |  one a:Agronomist, f:Farmer | (r in f.
    messageResponseF) and (r in a.messageResponseA)
}

fact OneRequestOneMessageRequest{
  all r:Request | one f:Farmer | r in f.messageRequest
}

fact NoBothRequestandResponse{      //A farmer can't
    response to the request he did
  no f:Farmer, r:Request | r in f.messageRequest and r in f.
    messageResponseF
}

fact OneResponeF{
  all r:Request | #(messageResponseF.r) < 2
}

fact OneResponseA{
  all r:Request | #(messageResponseA.r) < 2
}


pred World1{
  #Farmer =2 and
  #Agronomist = 1 and
  #HarvestedField = 0 and
  #Topic = 0 and
  #Suggestion = 0 and
  #Request = 0 and
  #DailyPlan = 3
}


pred World2{
  #Farmer =2 and
  #Agronomist = 2 and
  #ProductionField = 2 and
  #HarvestedField = 1
  #updates = 3
  #DirectSuggestion = 1 and
  #Topic = 0 and
  #Request = 0 and
  #DailyPlan = 2 and
  #Number = 3
}


pred World3{
  #Farmer =2 and
  #Agronomist = 1 and
  #ProductionField = 0 and
  #HarvestedField = 0 and
  #Topic = 2 and
  #Request = 4 and
  #messageResponseA = 2
}



pred World4{
  #Farmer =2 and
  #Agronomist = 1 and
  #ProductionField = 3 and
  #DirectSuggestion = 2 and
  #FarmerSuggestion = 2 and
  #Crop = 3  and
  #Topic = 0 and
  #Request = 0
}


run show for 4
```

4.2 First World

In this first world (Figure 45), the focus is on the registration, the types of users, how an account is managed. In the model illustrated each account has the same password, it is not restriction of the system but it is a possible scenario. Here the crucial aspect is: the farmers have only one location and every location belongs to a certain area, the agronomist has different daily plans for different dates where visits certain farmers, the agronomist is in charge of an area.

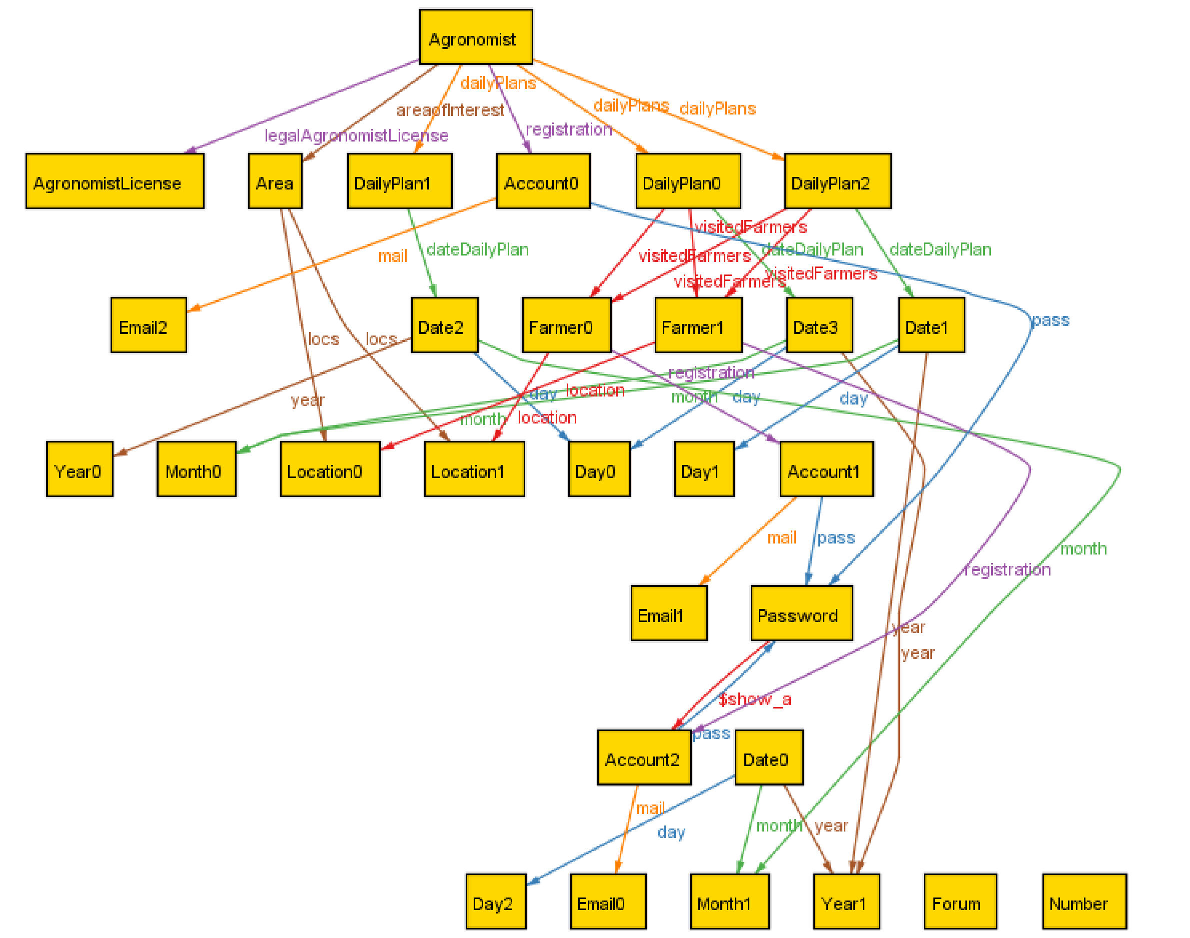


Figure 45: First World generated

4.3 Second World

This second world (Figure 46) is similar to the previous one with additional aspects: the possibility for farmers to add production fields with their set of parameters and if it is harvested there exists the final date e the amount planted. For each production field we have updates, identified like feedbacks of the farmers

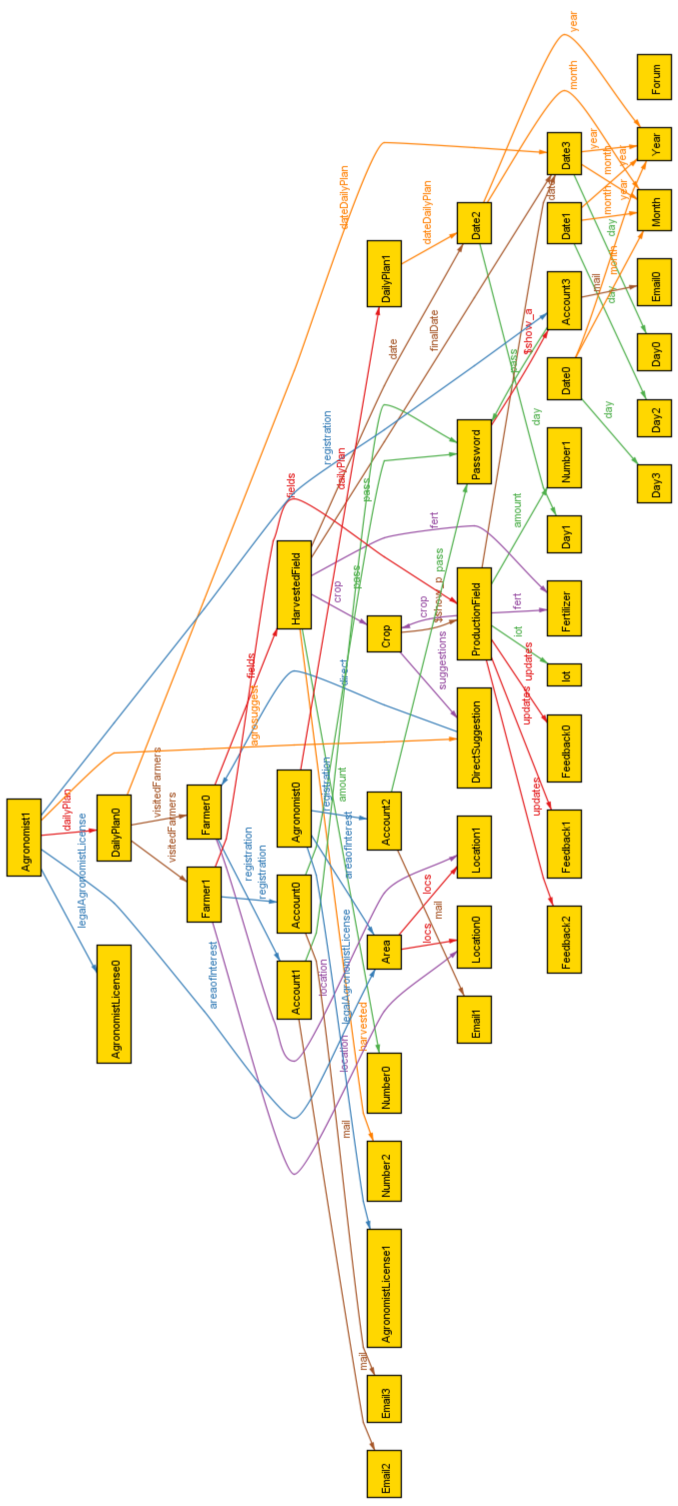


Figure 46: Second World Generated

4.4 Third World

The third example world (Figure 47) focuses on the request function and on the forum interaction between farmers.

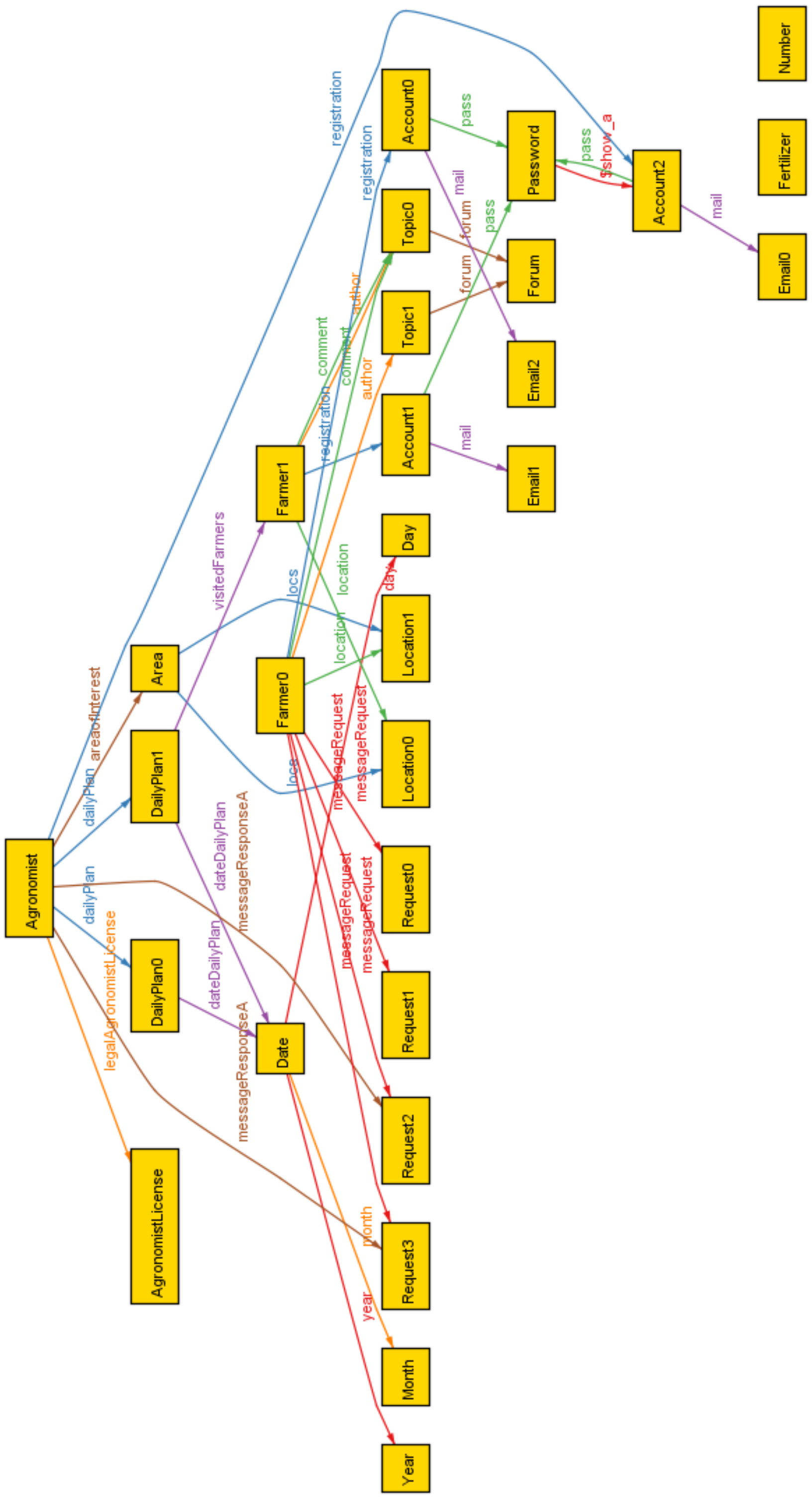


Figure 47: Third World Generated

4.5 Fourth World

The Fourth example world (Figure 48) is also similar to the first world but it emphasizes the funcionality of the suggesstions. As described previously in the document we have two different types of suggestions: the farmer suggestion so written by a farmer and the direct suggestion written by an agronomist.

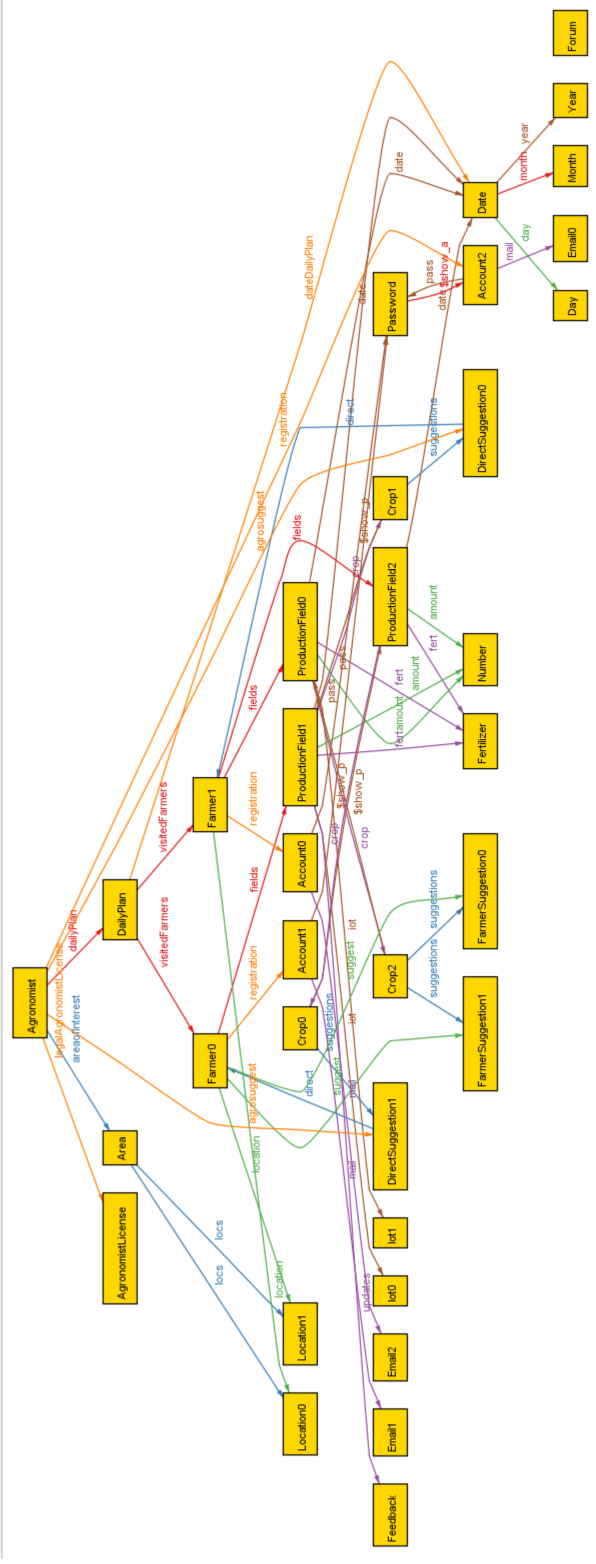


Figure 48: Fourth World Generated

5 Effort Spent

5.1 Teamwork

Task	Teamwork's Hours
Introduction	2
Product Perspective	2
Product Functions	4
Domain Assumptions	2
External Interface Requirements	1
Functional Requirements	10
Non-functional Requirements	0.5
Formal Analysis Using Alloy	3

Table 38: Teamwork effort

5.2 Individual Work

Task	Mattia Sabella's Hours
Introduction	0.5
Product Perspective	4
Product Functions	8
Domain Assumptions	2
External Interface Requirements	0.5
Functional Requirements	10
Non-functional Requirements	0.5
Formal Analysis Using Alloy	10

Table 39: Mattia's effort

Task	Francesca De Donato's Hours
Introduction	1.5
Product Perspective	1
Product Functions	2
Domain Assumptions	3
External Interface Requirements	6
Functional Requirements	4
Non-functional Requirements	2
Formal Analysis Using Alloy	2

Table 40: Francesca's effort

Task	**Salvatore Scozzari's Hours**
Introduction	1
Product Perspective	2
Product Functions	1.5
Domain Assumptions	6
External Interface Requirements	3
Functional Requirements	8
Non-functional Requirements	3
Formal Analysis Using Alloy	10

Table 41: Salvatore's effort

Appendices

A Revision History

- **v.1.0** December 1, 2021
  - Defined Goals, Domain, Requirments, World and Shared Pheonomena.
  - Defined State Diagrams, Product functions, class diagrams
- **v.2.0** December 15, 2021
  - Defined Use Cases, Sequence Diagrams
  - Defined External Interface Requirments
  - Defined Alloy models
- **v.1.3** December 22, 2021
  - Small Fixes

B Software and Tools used

- LaTeX as document preparation system
- Git & GitHub as version control system. The repository of the project is here
- Draw.io for the pictures (interfaces and World and Machine)
- Balsamiq for the mockups
- Alloy as model analyzer