POLITECNICO
MILANO 1863

Computer Science and Engineering

Software Engineering 2

Academic year 2021-2022

Requirements Analysis and Specification Document

Data-dRiven PrEdictive FArMing in Telangana

Authors:

Arslan Ali	971503
Elisa Servidio	996387
Federica Suriano	953085

December 23, 2021

Version 1.0

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The Requirement Analysis and Specification Document (RASD) has the purpose of describing, to a wide range of potential readers, the system to be developed for the problem under consideration.
It contains the description of the scenarios, the corresponding use cases and the models describing requirements and specifications.
It focuses also on interactions between the system and the users, including the implied constraints and functionalities that are going to be implemented.

The drafting of this document aims to follow the assignment which was proposed by the Software Engineering course of the Politecnico di Milano and to satisfy the required points of analysis. The project comes from an initiative presented by UNDP India, a United Nations division and within the Healthsites initiative.

1.1 Purpose

Telangana is the 11th largest and the 12th most populated state in India with a geographical area of $112,077 km^2$ and 35,193,978 residents (data from 2011, Wikipedia). The economy of Telangana is mainly driven by agriculture, a sector which plays a pivotal role in all India's economy: over 58% of rural households depend on it as the principal means of livelihood, 80% of whom are smallholder farmers with less than 2 hectares of farmland. More than a fifth of the small-holder farm households are below poverty.

Worldwide there are many threats to the agriculture sector.
World population is estimated to reach 9.7 billion by 2050 (recent UN estimate), therefore food demand is expected to increase anywhere between 59% to 98% by 2050 (Harvard Business Review). Climate change is predicted to result in a 4%-26% loss in net farm income towards the end of the century. The COVID-19 pandemic has greatly exposed the vulnerabilities

of marginalized communities, small holder farmers and the importance of building resilient food systems.

This calls for a revamp of the entire food supply chain to help bolster countries against shocks and challenges. For this reason, Telangana's government aims to design, develop and demonstrate anticipatory governance models for food systems using digital public goods and community-centric approaches to strengthen data-driven policy making in the state. To achieve this goal, Telangana wants to partner with IT providers with the aim of acquiring and combining data concerning weather forecasts, agriculture production (types and produced amount per product, amount of water used by each farmer), humidity of soil and information provided by the governmental agronomists. Acquiring and combining such data, the software system DREAM supports the work of three types of actors: policy makers, farmers, and agronomists.

DREAM allows Telangana's policy makers to identify farmers who are performing well and those who are performing particularly bad. The first ones, especially the more resilient to meteorological adverse events, will receive special incentives and will be asked to provide useful best practices to the others. Moreover, the system will help policy makers to understand whether the steering initiatives carried out by agronomists with the help of good farmers produce significant results. Thanks to the application the policy makers will be able to make decisions in the real world by analyzing and visualising data regarding farmers.

On the other hand, farmers are allowed to visualize data relevant to them based on their location and type of production, such as weather forecasts and personalized suggestions (i.e. specific crops to plant or specific fertilizers to use). Farmers can insert in the system data regarding their production and any problem they face. They can also request for help by agronomists and other farmers with whom they can also create discussion forums.

Eventually, agronomists are in charge of a certain mandal, which is a local government area and administrative division of Telangana. They can receive information about help requests and answer them and they can visualize data concerning weather forecasts and farmers in the area. Furthermore, agronomists can visualize, update and confirm a daily plan to visit farms, assuming that all farms must be visited at least twice a year, but those that are under-performing should be visited more often, depending on the type of problem they are facing.

1.1.1 Goals of the Application

G.1 Allows policy makers to visualize and analyze data of farmers

G.2 Allows policy makers to identify those farmers who are performing well and those who are performing bad

G.3 Allows policy makers to verify the improvement of farmers who have been already helped by agronomist or good farmers

G.4 Allows farmers to visualize weather conditions

G.5 Allows farmers to visualize their next visits

G.6 Allows farmers to receive notifications

G.7 Allows farmers to insert in the system data about their production and any problem they face

G.8 Allows farmers to receive help from agronomist and other well performing farmers through help requests

G.9 Allow farmers to interact on the discussion forum

G.10 Allows agronomists to receive notifications regarding new help requests created

G.11 Allows agronomists to visualize data concerning farmers in the mandal

G.12 Allows agronomists to visualize weather conditions in the mandal

G.13 Allows agronomists to visualize their own daily plan to visit farms in the mandal

G.14 Allows agronomists to manage their own daily plan

1.2 Scope

The goal of the DREAM software product is to develop and adopt anticipatory governance models for food systems while strengthening data-driven state policy.
It takes care of the acquisition and management of all data collected in order to support the work of farmers, agronomists and policy makers.
The system aims to collect data not only from sensors located throughout the territory, but also from farmers. The analysis of the acquired data aims to improve the production of farmers.
Bad performing farmers are identified by policy makers and helped by their related agronomist and by well performing farmers.

Everything is supervised by agronomists who take care of their own geographical areas of competence.

To better understand all the phenomena involved, we distinguish them into two types according to the "World and Machine paradigm" [M. Jackson and P. Zane]. The World is the environment surrounding the system, while the Machine is the system itself.

1.2.1 World, Machine and Shared Phenomena

Table 1.1: World, Machine and Shared phenomena

Phenomenon	**Who controls it?**	**Is it shared?**
Telangana government assigns to each agronomist his/her ID code	W	N
Telangana government assigns to each policy maker his/her ID code	W	N
Telangana government assigns to the agronomist the mandal which he/she is responsible of	W	N
An unregistered user signs up to the application	W	Y
A user logs into the application	W	Y
The system checks validity of data inserted by the (unregistered) user when he/she (registers) logs into the application	M	Y
The system retrieves longitude and latitude from the address of a farmer's farm	M	N
The system retrieves weather forecasts, humidity of soil and amount of water used corresponding to a certain position	M	Y

A farmer inserts in the system data about his/her harvest	W	Y
The system processes farmer's personalized suggestions based on his/her address and his/her production	M	N
The system sends notification to the farmer	M	Y
The system processes the performance of a farmer	M	N
A policy maker selects the options which he/she wants to apply to the visualization of farmers' performance on the map	W	Y
The system associates each farmer to the corresponding mandal	M	N
A policy maker visualizes the performance of farmers on a map	W	Y
A policy maker selects the options which he/she wants to apply to the visualization of farmers' performance trend on the time chart	W	Y
The system processes the performance trend of farmers over time according to the options selected by the policy maker	M	N
A policy maker visualizes the performance trend of farmers over time on a time chart	W	Y
A farmer receives an incentive	W	N
A farmer makes a help request	W	Y
The system retrieves the help request recipients (agronomist or even well performing farmers)	M	N

The system sends notification to the agronomist	M	Y
A well performing farmer replies to a help request	W	Y
An agronomist replies to the help requests sent to him/her by farmers	W	Y
A farmer solves a help request	W	Y
The system updates the status of an help request	M	N
A farmer opens a thread about a topic on the dedicated forum	W	Y
A farmer replies on the discussion forum creating a post	W	Y
The system associates each agronomist to the corresponding mandal	M	N
An agronomist visualizes weather conditions on the app	W	Y
An agronomist selects the options which he/she wants to apply to the visualization of farmers' performance score on the map	W	Y
The system processes the performance score of farmers according to the options selected by the agronomist	M	N
An agronomist visualizes performance score and personal information of farmers on a map	W	Y
An agronomist visualizes on a table performance of his/her mandal's farmers	W	Y

The system creates the agronomist's daily plan, guaranteeing at least two visits per year for each farmer	M	N
An agronomist visualizes his/her daily plan	W	Y
An agronomist visits a farm belonging to his/her mandal	W	N
An agronomist carries out his/her daily plan	W	N
A farmer visualizes relevant information which are: weather forecasts, soil moisture and visits	W	Y
A farmer and his/her associated agronomist reschedule a visit on call	W	N
An agronomist updates his/her daily plan	W	Y
An agronomist confirms his/her daily plan execution	W	Y
An agronomist specifies deviations from his/her daily plan at the end of the day	W	Y
The system updates the status of daily plan	M	N

1.3 Definitions, Acronyms, Abbreviations

In the following section is clarified the meaning of some definitions, acronyms and abbreviations which will be use in the RASD, in order to help the general understanding of the document.

1.3.1 Definitions

Table 1.2: Definitions

The system	The whole system to be developed
User	A farmer/agronomist/policy maker who uses the application
Unregistered user	A farmer/agronomist/policy maker who is not yet registered into the application
Application service	Functionality offered by the system for certain users
Policy maker	The user of the application who decides about new policies for Telangana
Farmer	The user of the application who owns or manages a farm
Address	The address inserted by the farmer which corresponds to his/her farm's location. It is composed by city, zip code, street and number
Performance	Indicator of the progress of a farmer's activity up to a certain date. Its value is calculated as numerical score
Score	Performance rating computed by a function that depends on: type and quantity of harvested product, weather conditions, quantity of water consumed, soil moisture
Well performing farmer	A farmer who has a score higher than a certain threshold
Bad performing farmer	A farmer who has a score below a certain threshold
Agronomist	The user of the application dealing with the management of a certain mandal

Mandal	A local government area and administrative division. Telangana is subdivided into districts which are themselves subdivided into mandals
Daily plan	Application service that allows the agronomist to manage his/her daily work schedule. Specifically, it allows him/her to track and organize visits to farmers
Discussion forum	Application service which a farmer can use to exchange ideas and opinions about a topic
Post	Contributions of the participants to the discussion placed one below the other in sequence
Thread	It is composed of the topic followed by the posts left by the various participants in the discussion
Help request	Application service which a farmer can use to request for help to the agronomist or/and other well performing farmers

1.3.2 Acronyms

Table 1.3: Acronyms

DREAM	Data-dRiven PrEdictive FArMing in Telangana
RASD	Requirement Analysis and Specification Document
UML	Unified Modelling Language
API	Application Programming Interface

1.3.3 Abbreviations

Table 1.4: Abbreviations

G.i	i-th goal
R.i	i-th requirement
D.i	i-th domain assumption
UC.i	i-th use case

1.4 Revision History

Table 1.5: Revision History

Version	Date	Authors	Summary
1.0	23/12/2021	Arslan Ali Elisa Servidio Federica Suriano	First release

1.5 Reference Documents

- Specification document: "R&DD Assignment A.Y. 2021/2022"
- Software Engineering 2 course slides A.Y. 2021/2022 - Politecnico di Milano
- EEE Std 830-1998: IEEE Recommended Practice for Software Requirements Specifications
- ETH Zürich (2009), Requirements Specification: `se.inf.ethz.ch/courses/2011a_spring/soft_arch/exercises/02/Requirements_Specification.pdf`

- Michael A. Jackson (1995), The World and the Machine: `http://mcs.open.ac.uk/mj665/icse17kn.pdf`.
- UML offcial specification `https://www.omg.org/spec/UML/`
- Alloy offcial documentation `https://alloytools.org/documentation.html`

1.6 Document Structure

The RASD is structured in the following five chapters:

- **Chapter 1 - *Introduction*:** It contains a general introduction to the problem of interest and a more detailed list of the goals of this project. The scope of the application is described thanks to the analysis of the phenomena involved. A part relating to the definitions used in the document is also included.
- **Chapter 2 - *Overall Description*:** It contains an overall description of the project including some possible scenarios of interest, the system class diagram and state diagrams. The most important product functions necessary for the correct functioning of the application and all the assumptions on the domain are also described.
- **Chapter 3 - *Specific Requirements*:** In this part the software product requirements are described in detail. The description of the interfaces is included and the functional requirements are defined thanks to the use of UML diagrams. This section also includes performance requirements, design constraints and software system attributes.
- **Chapter 4 - *Formal Analysis using Alloy*:** It contains formal modeling of the software product using the Alloy tool and dedicated comments to better clarify each part of the modeling.
- **Chapter 5 - *Effort Spent*:** It contains all the information regarding the hours of work required to create the document and the tasks assigned to each person in the team.

At the end of the document the bibliography is also included.

Chapter 2

Overall Description

This chapter contains a general description of the system from a high level point of view.
It describes the general factors that affect the product and its requirements. It does not state specific requirements. Instead, it provides them a background, which is useful to define them in detail in *Chapter 3*.
Starting from scenarios and domain models, it proceeds with an analysis of product functions and users' most relevant needs in order to define assumptions, dependencies and constraints of the entire application system.

2.1 Product Perspective

In subsection *2.1.1* are listed the most relevant scenarios which are a narrative description of what users do and experience as they try to make use of the application.
They provide a general depiction of how major components of the system and users interact.
Moreover, in subsections *2.1.2* and *2.1.3* the domain of the system is defined through different models using UML.

2.1.1 Scenarios

1. Scenario: **Sunil discovers DREAM**

   Sunil is a farmer from Telangana. Unfortunately, due to the adverse and unpredictable effects of climate change, his production of black carrots is gradually decreasing. Sunil learns from a colleague of his that there is an application called DREAM that could help him. Sunil registers and logs in, hoping to profit from it and improve his production.

2. Scenario: **Mahima is thrilled with joy for DREAM**

   Mahima is the Additional Director of Agriculture who assist the state Head quarter Commissioner&Director of Agriculture in Telangana. She belongs to the Department of Agriculture, which provides agricultural services to farmers and aims to transfer the latest technical knowledge to the farming community. She was one of those who pushed for the creation of DREAM. Mahima is thrilled with joy for its market launch and she is looking forward to use the application, therefore she gets her personal ID code to be able to registers with the role of policy maker and logs into DREAM.

3. Scenario: **A new job for Shanti**

   Shanti is a young agronomist who operated in Palakeedu, a mandal in Suryapet district. She sees the announcement published on the Telangana government website in which it is reported that the DREAM application has been launched on the market and that for each mandal is required an agronomist. She is unemployed and interested in filling this role. Therefore, she sends her CV, gets hired and receives her personal ID code which allows her to register into DREAM. Eventually, she inserts Palakeedu as mandal of her competence.

4. Scenario: **Rajesh looking for advice**

   Rajesh is a landowner from Adilabad district in Telangana, he owns several agricultural properties in the city. After his recent business trip to Russia, where he attended the beet fair, he is convinced to plant beets on his land too. As this is a little-known plant in the area, he doesn't know who to ask for advice. Rajesh's son Ram, also in the agricultural sector for several years, advises his father to use DREAM since Rajesh already had an account registered with the application. Rajesh consults on the application the discussions already opened by other Telangana farmers to see if any of them had ever sought advice on beet cultivation. Unfortunately he cannot find any, so thanks to the suggestion of his son Ram he decides to open a thread on the dedicated forum, so that he can ask for advice about the plant from other farmers throughout Telangana.

5. Scenario: **Champak tries natural farming**

   Champak is a farmer who has planted wheat. The last crop was poor because he didn't have enough money to buy suitable pesticides. He still can't afford them, however he cannot suspend his agricultural activity since his family depends on it as the principal means of livelihood. For this reason Champak creates a help request to both the agronomist and well performing farmers. The first to answer is Jayapal, a well performing farmer from his same mandal. Jayapal suggests him to plant potatoes instead of wheat because they are more resistant. However, Champak has already planted wheat therefore changing crops is out of the question. Champak waits for other answers until Rajat, the agronomist, replies suggesting him to cover the seeds with microorganisms obtained from particular formulations of cow dung. Champak is willing to follow the advice and is satisfied with the answer. He marks the help request as solved.

6. Scenario: **Deepa fights soil salinity**

   Deepa is a farmer of Telangana who is already using the application DREAM. While approaching to plant cotton seeds, she has a problem regarding the salinity of the soil which is particularly high and would not allow cotton to grow. One month later, the cotton she planted is ready for harvest. The amount of cotton obtained from her hectare of land results in 3 bales, which correspond approximately to 107 kg each. Deepa accesses the app and inserts in the system type and quantity of product harvested, also adding that it was necessary to buy a biostimulant whose formulation neutralizes excess salts within the soil.

7. Scenario: **Haresh's first encounter with coffee**

   Haresh is a farmer of Telangana who is already using the application DREAM. It is October and within a few days he will have to harvest the sorghum he has planted. To better organize the work of the next few days, Haresh accesses the application and checks the weather forecasts. After the harvest he will also have to decide what to plant, which is why he also checks the soil moisture. Furthermore, due to the drought expected for November, the app suggests Haresh to plant coffee, a product that does not require a particularly humid soil. Haresh thinks that it could be a good idea, however he has never cultivated coffee so it could be necessary to confront with the agronomist about some technical details. Haresh pleasantly reads that his next visit with

the agronomist is scheduled for November 5th.

8. Scenario: **Anita and her strategic plan**

   Anita is the administrator of her mandal Sarangapur, in Jagtial district, as an inspector of production and development of the primary sector. Sarangapur is characterized by great periods of drought, so water is a particularly precious resource for citizens. Recent analyzes have shown that over 90% of the water is used by farmers. Anita discovers DREAM and she wants to monitor the water consumption within each mandal in Telangana over the current year. Her goal is to identify the one that consumes the least water to study its strategies, so she enters the parameters of interest (quantity of water and Telangana mandals) on the application. At this point the app returns the mandals and the corresponding consumed water in this last year and Anita is pleased to discover that the primacy is Pegadapalli, a mandal from her own district.

9. Scenario: **Manoj, the Indian tycoon**

   Manoj is an Indian tycoon who would like to invest his capital in the agricultural sector. He accesses the DREAM application as policy maker and checks the map showing the performance score of each mandal. Manoj identifies the best performing ones and decides to invest in Geesugonda mandal which is located in Warangal district.

10. Scenario: **Durvish, the inspector**

    Durvish is the administrative head of Doma mandal, in Vikarabad district. The DREAM application was launched on the market the year before and now he wants to understand whether the steering initiatives carried out by agronomists with the help of good farmers have produced significant results. Therefore, Durvish accesses DREAM with his account and he enters the parameters of interest on the application to select only farmers of his mandal with at least an help request solved. At this point the app returns the selected farmers. Durvish wants to visualize their general trend of the performance score as indicator of the efficiency of the application. He selects score as attribute and mean as operation, in return a time chart is shown. He pleasantly discovers that, except for an initial fluctuating trend, the general performance score of farmers with at least a solved help request of his mandal is constantly growing, a sign of effective usefulness of DREAM.

11. Scenario: **The busy life of Aruna**

    Aruna is the DREAM agronomist responsible for Jaipur mandal in Mancherial district. It is early morning and the working day is about to begin. Aruna consults the daily plan to check which are the scheduled visits for the day. Further on, she completes all the appointments scheduled in the morning and she drives her car to reach Jaya, a farmer. Unfortounately, Aruna has a small accident with her car, she is unharmed but she will miss the appointment. She informs Jaya and calls a mechanic. She finally returns home in the evening and confirms the daily plan, also specifying the deviation corresponding to the missed visit due to the accident. She also has to reschedule in the first free slot in her agenda the appointment with Jaya. To do so, she updates the daily plan of the next day.

12. Scenario: **Lohit helps the most needy**

    Lohit is the DREAM agronomist responsible for Raipole mandal in Siddipet district. The performance score in his mandal for a good percentage of farmers has been around a value that has not been particularly high for more than six months. For this reason Lohit wants to visualize data regarding best performing farmers to formulate strategies to be applied to the most needy. He accesses DREAM and visualizes farmers whose performance score is between the highest. Analyzing the corresponding data he finds out that they cultivate all the same type of product: rice, which is cultivable under widely varying conditions but prefers hot and humid climate. Therefore, Lohit consults weather forecasts and humidity of soil trend, discovering that with a very high probability there will be the perfect conditions to plant rice in the next weeks. He decides that he will advice Abhik, a bad performing farmer, about this opportunity when he will visit him in two days.

2.1.2 Class Diagram

In this section is reported the class diagram of the system, whose main concepts are presented from a high-level point of view.

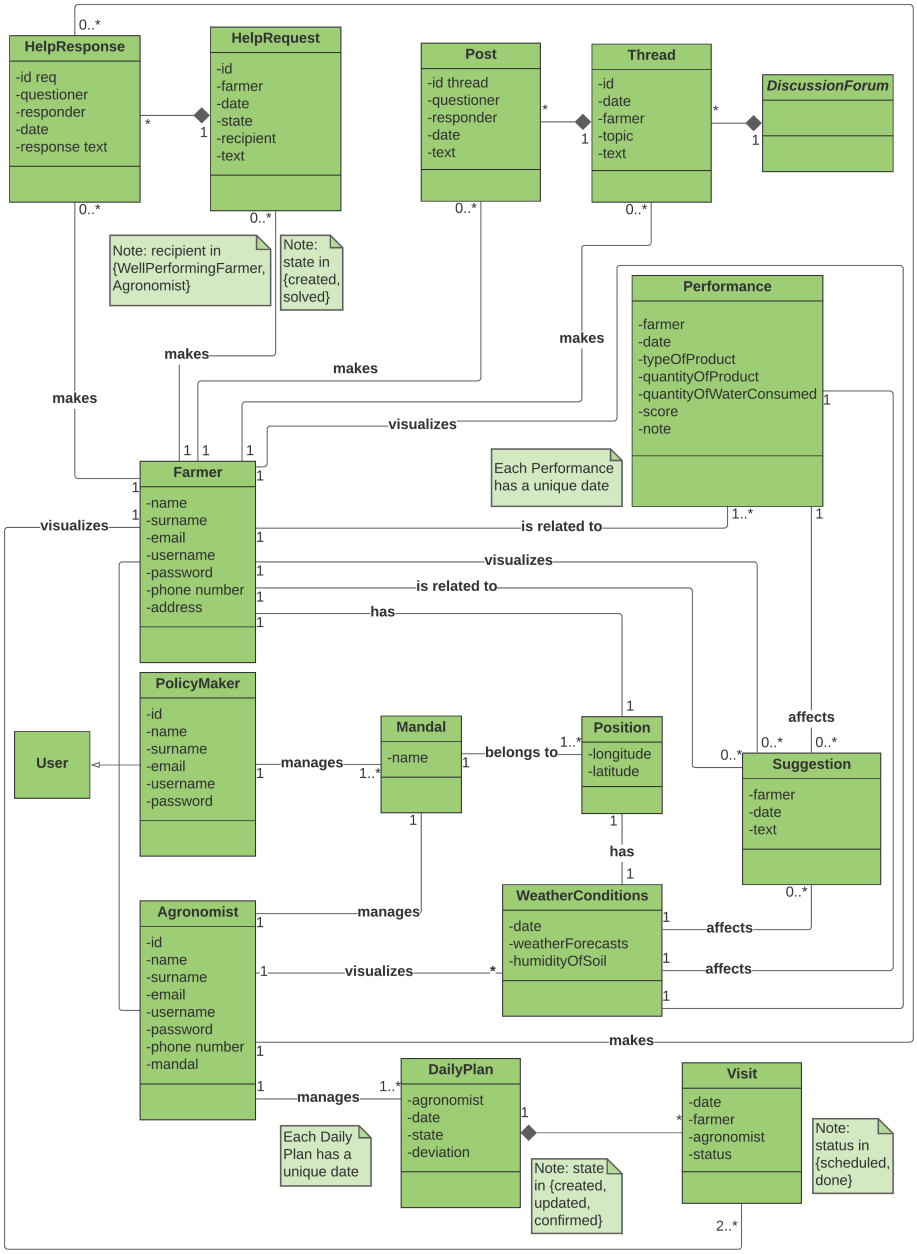

Figure 2.1: Class Diagram

2.1.3 State Charts

Help Request State Diagram

This state diagram shows the two states of a help request.
The help request is *Created* and then *Solved* by the farmer, depending on whether he/she is satisfied with the reply received or not.

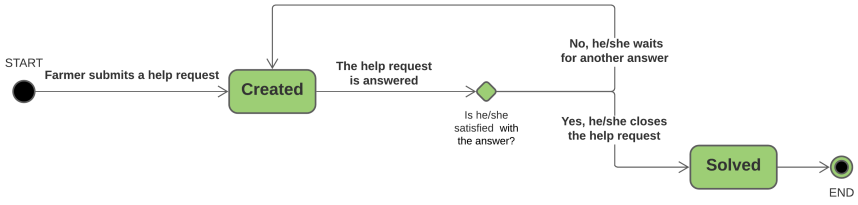

Figure 2.2: State Diagram - Help Request

Daily Plan State Diagram

This state diagram shows the three states of a daily plan.
The daily plan is *Created* by the system and then *Updated* or carried out and *Confirmed* by the agronomist, depending on whether he/she needs to change the plan or not.

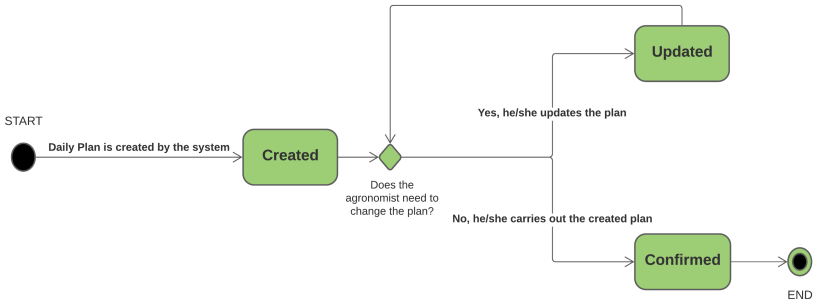

Figure 2.3: State Diagram - Daily Plan

Visit State Diagram

This state diagram shows the two states of a visit.
The visit is *Scheduled* and then *Done* after the agronomist carries out the visit and confirms the corresponding daily plan.

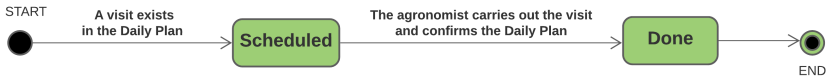

Figure 2.4: State Diagram - Visit

2.2 Product functions

2.2.1 Registration

In order to use the application all the actors (policy makers, agronomists and farmers) must be registered. The registration of an account can be performed in two ways: through the web application on a computer or through the mobile application on a smartphone.

In the case of the web application, the farmer opens it and clicks on the "Register" button on the homepage. At this point he/she is redirected to the registration page where the farmer is asked to select which category of users he/she belongs to: "Policy Maker", "Farmer" or "Agronomist". After selecting the right category he/she can enter in the predetermined fields the required data which in the case of the farmer are: name, surname, email, password, confirmation password, telephone number and his/her farm's address. The format of the characters entered is checked so that it complies with the predetermined rules of good formatting. Before submitting registration registration, it's also required to accept terms and conditions. When the farmer clicks on the "Register" button, he/she is redirected to a new page that suggests to check his/her mailbox, where he/she will find an email generated and sent automatically by the system and with which the farmer can confirm his/her email by clicking on a link. After that the farmer is redirected to the login page in the Web app. The farmer then receives an email confirming his/her registration to DREAM.

Registration through the mobile application takes place in the same way as above. In fact, the farmer opens the application previously downloaded on his/her device. On the home page he/she clicks on the "Register" button. At this point he/she is redirected to the registration page where he/she can enter his/her data. The registration process continues as previously explained.

Registration takes place in the same way for all three actors, but there are some differences in the required data depending on the account role of the person who is registering. The data that agronomists and policy makers can enter are the same as those required by a farmer, with the exception

of the farm's address. The latter two roles (agronomists and policy makers) must enter an ID code assigned to them by the Telangana government to gain access to the system with their specific role after it is validated. Moreover, only agronomists also have to inserts the mandal they are responsible of.

2.2.2 Help Request

DREAM allows farmers to make help request to the agronomist or even to well performing farmers if specified. The farmer accesses to the page "Help Request" and clicks on "Create a Help Request" button and accesses the corresponding page in which, in addition to the agronomist provided by default, he/she can select as recipient even well performing farmers of his/her same mandal. Eventually the farmer completes the text form corresponding to the body of the help request. Once finished, he/she confirms clicking on the correlated button "Confirm" and the help request is created. The help requests created by each farmer can be visualized by themselves accessing the page "Help Request" and clicking in the button "My Help Request". If someone replies the farmer visualizes in the same page the answer with the associated question *"Are you satisfied with the answer?"* and the corresponding two possible answers *"Yes"* or *"No"*. Whether he/she is satisfied or not, he/she can:

1. close the help request clicking on the button "Yes"
2. not close it clicking on the button "No", waiting to receive other answers

On the other hand, the users who can reply to a help request are the agronomist and well performing farmers of the same mandal. When a help request is created and they are mentioned among its recipients, they can visualize it in the "Help Request" page , where it is listed among the existing others. For each one there is a button "Answer" which they can click on to reply to the request.
Once the help request is solved by the farmer who made it, recipients cannot visualize it anymore.

2.2.3 Discussion Forum

DREAM allows farmers to open a thread on the discussion forum. The farmer accesses the page "Discussion forum" and clicks on the button "Open a thread". He/She is redirected to a form which he/she fills in with topic and text and then he/she confirms.
The farmer can also look for a certain topic: he/she accesses the page "Discussion forum" and clicks on the search button, the system provides his/her a form to fill in with the topic to be found and eventually it provides the farmer the list of existing associated threads.

If he/she wants to reply to one of them he/she can click on it and then, after pressing the button "Answer", he/she is redirected to a form in which he/she has to fill in the text's field. Eventually he/she confirms.

2.2.4 Daily Plan

DREAM allows agronomists to update the daily plan. He/She accesses the page "Daily Plan" and clicks on the button "Update". He/She is redirected to a form which he/she fills in with date (day and hour) and farmer and then he/she confirms.
The agronomist can also confirm the daily plan at the end of the day: he/she accesses the page "Daily Plan" and clicks on the button "confirm". He/She is redirected to a form with a field called "deviations" to be filled in only if modifications have been applied to the daily plan carried out during the day. Eventually he/she confirms.

2.2.5 Notifications

The system provides the farmer notifications regarding:

- suggestions according to his/her relevant information regarding his/her production and his/her address
- new visits scheduled in the daily plan regarding him/her
- replies to his/her own help requests not yet solved
- replies to his/her own threads opened on the discussion forum

The agronomist and only well performing farmers are also provided with notifications regarding new help requests received.

The actors involved can view the notification area directly from their homepage by clicking on the notification icon. At this point the actor is able to view all the notifications received in chronological order.

2.2.6 Farmer inserts data regarding his/her production

The farmer can enter relevant data about his/her production. Through his/her homepage he/she has the possibility to insert new data on his/her harvest. The data that the farmer has to enter in the form are: type of product and quantity of product collected. In addition, the farmer is allowed to add notes as well.
The system will add the current date of the insertion, the identification of the farmer (the username that uniquely identifies him/her) and the quantity of water consumed, information obtained thanks to some sensors. After entering these data, the system will update the score for the farmer taking

into account the new parameters entered. The data entered will be saved in the database.

2.3 User Characteristics

The actors of the applications are the following:

- *Unregistered user*: a person who has not yet registered and is only allowed to sign up becoming either a Policy maker or an Agronomist or a Farmer depending on the type of role selected.
- *User*: a person who is already registered into the application and is allowed to sign in according to his/her role (Policy maker, Agronomist, Farmer).
- *Policy maker*: a user of DREAM. He/She is an institutional figure in the public or private sector who has the power to elaborate and determine guidelines and strategies on the most relevant issues for society and politics.
- *Agronomist*: a user of DREAM. He/She is a multidisciplinary professional figure. He/She works in favor of farms and small farmers by analyzing and developing projects based on direct observation and on the study of the best environmental solutions in order to improve productivity and enhance agricultural products.
- *Farmer*: a user of DREAM. He/She is the figure who deals with managing, carrying out and producing crops of different kinds and types, constantly taking care of their good health. Furthermore, the farmer ensures that the places where the cultivation takes place fully comply with the hygienic-sanitary rules of the cultivation. They also provide for the maintenance of the structures necessary for the activities and the reclamation of the environments in which the cultivation takes place.

2.4 Assumptions, dependencies and constraints

This subsection of the RASD lists each of the factors that affect the requirements stated in *Chapter 3*.

An assumption is something that is believed to be true and which is outside of the project team's control. Unlike constraints, which put restrictions on DREAM, assumptions open possibilities for it and make its successful completion possible. Eventually, dependencies refer to technology components, applications, and servers on which DREAM relies in order to be functional.

2.4.1 Domain Assumptions

D.1 Each user who wants to use DREAM is needed to have a device connected to Internet

D.2 Data retrieved by external services (quantity of water consumed by each farmer, soil moisture and weather forecasts) are supposed to be accurate

D.3 A farmer does not forget to insert data regarding his production (type and quantity of product harvested)

D.4 A farmer does not insert fraudulent information as input to improve his performance score (e.g. higher quantity of product harvested)

D.5 A farmer inserts in the system the correct address corresponding to his farm's location

D.6 Each farmer is related to exactly one farm's address

D.7 A farmer doesn't forget to solve a help request

D.8 The external service used by the system to retrieve latitude and longitude of a farm using the address provided by a farmer is supposed to be accurate

D.9 A unique ID code is given to each policy maker to be able to register

D.10 A unique ID code is given to each agronomist to be able to register

D.11 ID codes assigned to policy makers and agronomists from Telangana government are saved in a database

D.12 An agronomist inserts in the system the correct mandal he/she is responsible of

D.13 An agronomist updates the daily plan when scheduling a new visit

D.14 An agronomist doesn't forget to confirm the daily plan by the end of the day specifying deviations correctly if needed

2.4.2 Dependencies

- The system will exploit Internet connectivity of users' device
- The system will retrieve data collected in Telangana (water consumed by each farmer, soil moisture and weather forecasts) by external services

- The system will use an external service to compute latitude and longitude corresponding of a farm's address provided by a farmer
- The system will use an external service to check validity of ID code of policy makers and agronomist

2.4.3 Constraints

Regulatory policies

Access to confidential and sensitive data must be restricted to protect it from being lost or compromised in order to avoid adversely impacting users. At the same time, users must be able to access data as required for them to work effectively.
All sensitive data and user information are acquired by the company under the accepted terms and conditions. Telephone numbers and email addresses won't be used for commercial uses.

Hardware limitations

The system requires a compatible device (smartphone or personal computer) with Internet connection to work properly.

Interfaces to other applications

The system relies on various external services to retrieve, compute and validate data needed for DREAM to work properly.

Parallel operations

Users of DREAM must be able to access database at the same time, therefore parallel operations must be granted in order to avoid collision or any other integrity issue.

Chapter 3

Specific Requirements

3.1 External Interface Requirements

In this section, most important user interfaces are shown.
Moreover, is given a description of the system in terms of hardware, software and communication interfaces.

3.1.1 User Interfaces

DREAM contains two main interfaces: mobile application and webapp interface. In both interfaces, all three actors can perform the same operations. However, it should be noted that certain features are more comfortable to use only from a certain interface. The following mockups will give a more detailed description in terms of interfaces.

WebApp Registration page

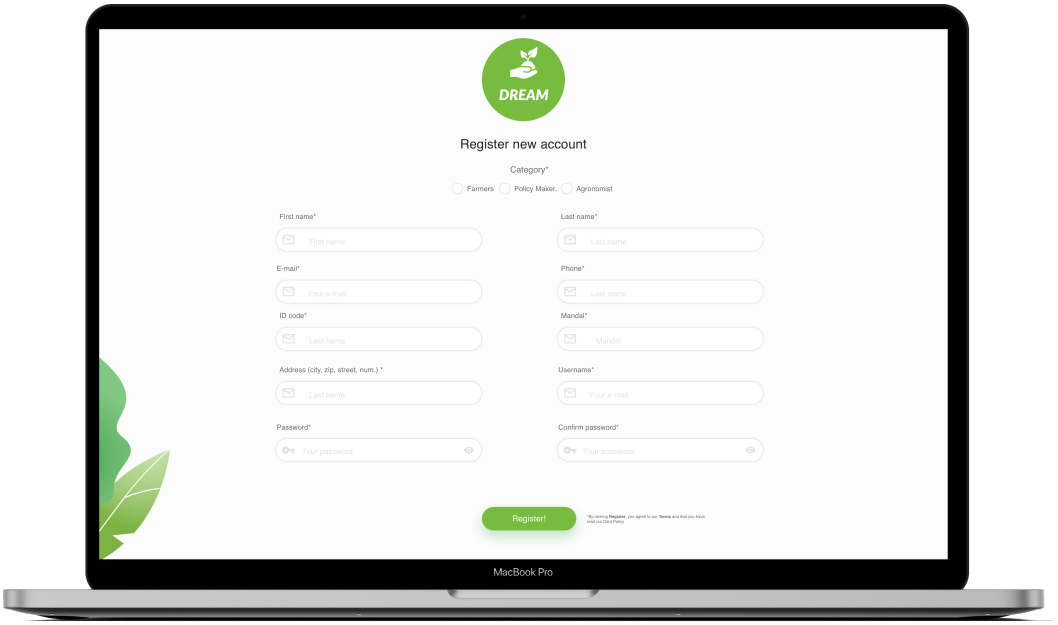

Figure 3.1: WebApp Registration page

The three actors can register to DREAM through this registration page, where there are some required fields, and some other fields according to radio button selected corresponding to their role. Registration is completed by accepting term and conditions, and by confirming the following mail.

WebApp Login page

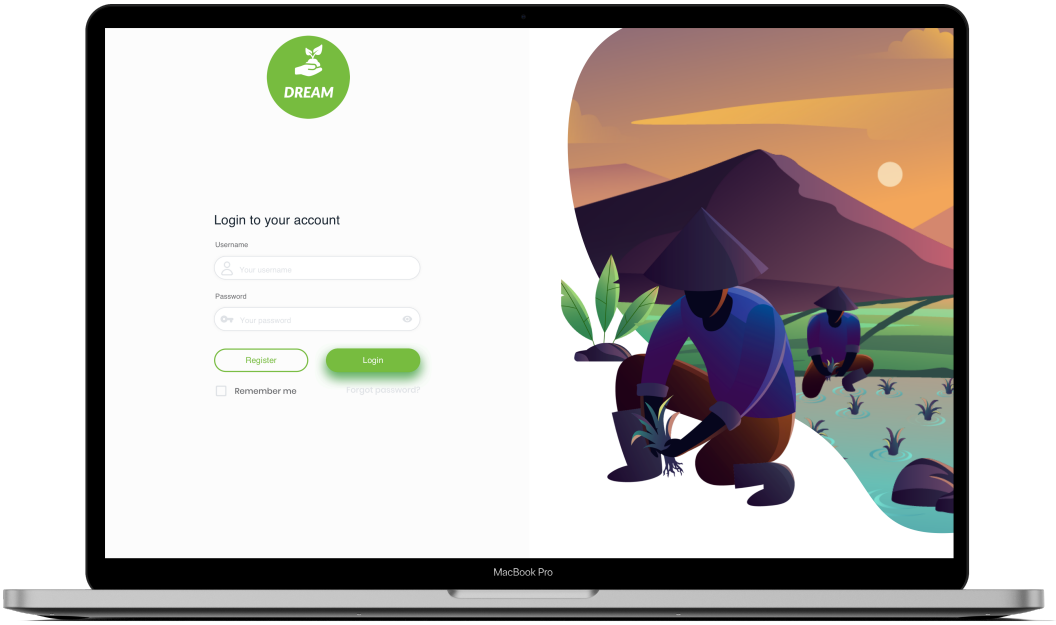

Figure 3.2: WebApp Login page

The three actors of DREAM can access to the application through this login page. If they do not yet have an account, they can simply access the registration page via an appropriate button.

WebApp Policy Maker Homepage

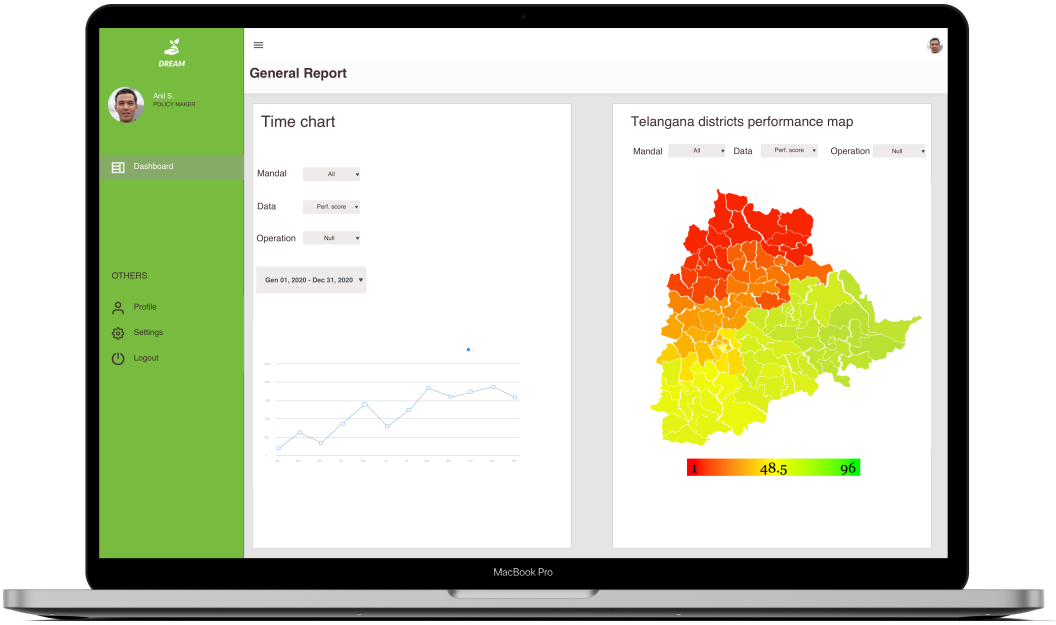

Figure 3.3: WebApp Policy Maker Homepage

This is policy maker homepage (in *Figure 3.3* called "Dashboard") where the he/she can generate a time chart according to some inputs, and can also visualize Telangana's farmers performance map using some filters.

WebApp Agronomist Homepage

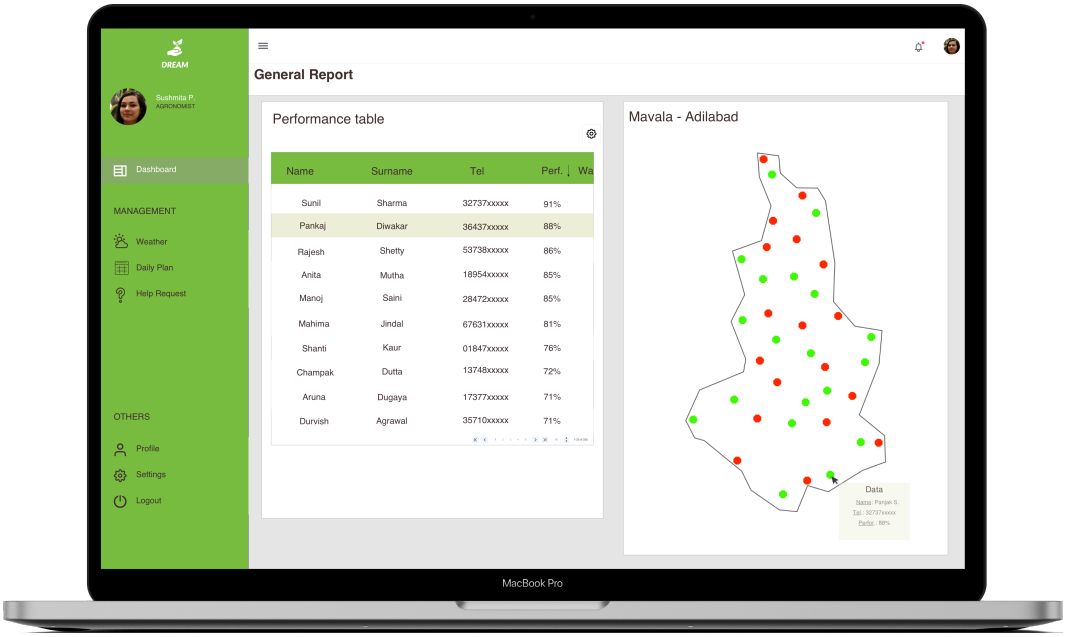

Figure 3.4: WebApp Agronomist Homepage

This is agronomist homepage (in *Figure 3.4* called "Dashboard") where he/she can view his/her mandal's farmer performance on a map, and he/she can also view some info about a farmer by pointing mouse cursor on a point, and the performance table, that resume, in descending order (by performance score), info's about farmers.

WebApp Agronomist Weather Conditions page

Figure 3.5: WebApp Agronomist Weather Conditions page

This is agronomist weather page, where he/she can visualize info about weather conditions and soil moisture of his/her mandal. He/she can also view information about weather forecasts in the next seven days by setting it into the menu.

WebApp Farmer Homepage

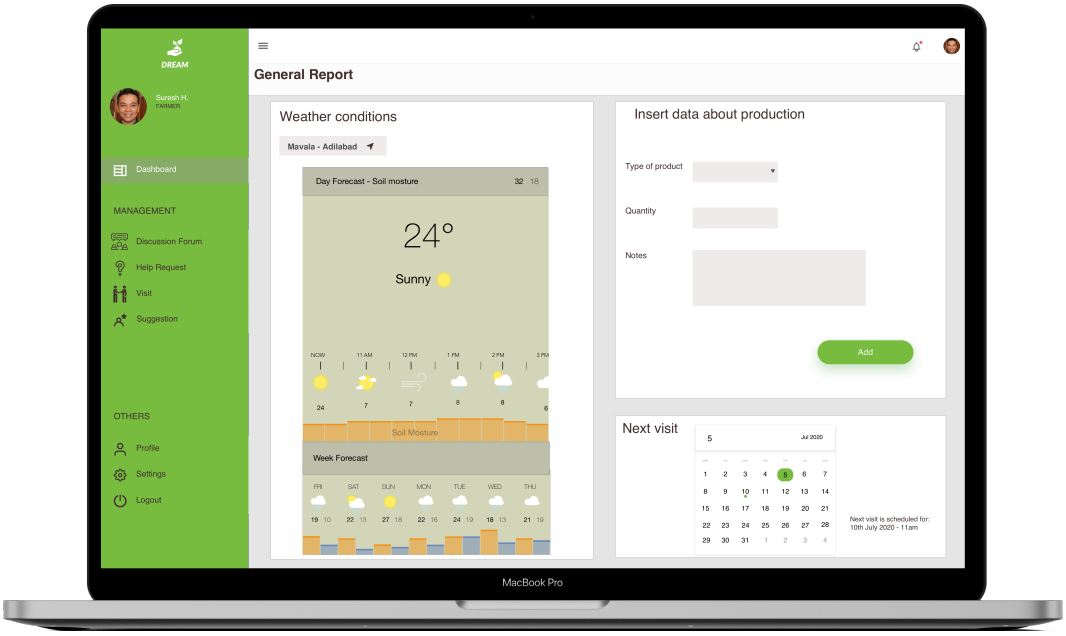

Figure 3.6: WebApp Farmer Homepage

This is farmer homepage (in *Figure 3.6* called "Dashboard") where he/she can visualize weather forecast and soil moisture based on her/his location, he/she can insert data about his/her production and can visualize on the calendar the next visit scheduled by his/her mandal's responsible agronomist.

WebApp Farmer Discussion Forum page

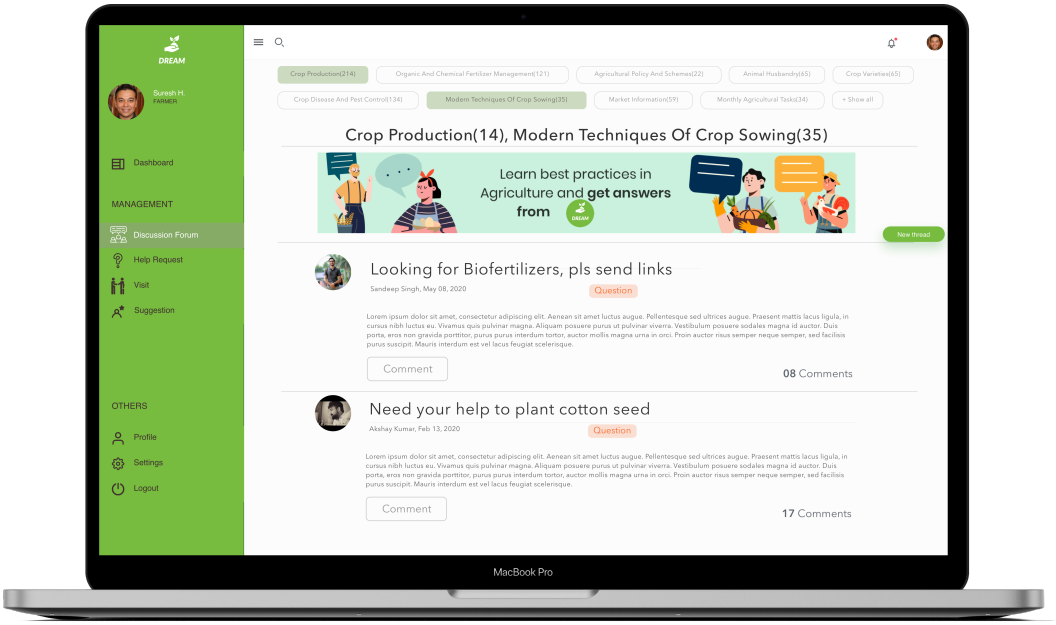

Figure 3.7: WebApp Farmer Discussion Forum page

This is farmer forum section, where he/she can interact with other farmers by opening new threads or replying to existing one. He/she can also look up for a thread using search function.

MobileApp Login and Registration page

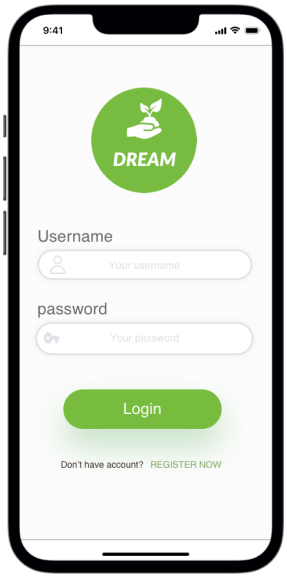

Figure 3.8: Mobile Login page

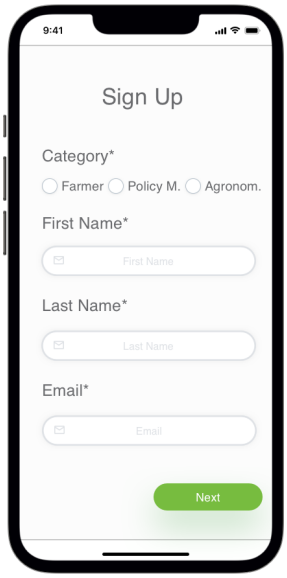

Figure 3.9: Mobile Registration page

MobileApp Farmer Menu

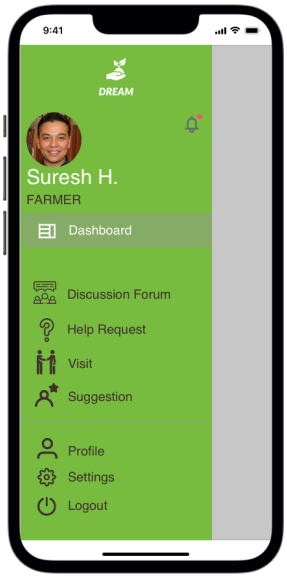

Figure 3.10: MobileApp Farmer Menu

Interface that shows the menu of actions that a farmer can perform in the application.

MobileApp Farmer Help Request page

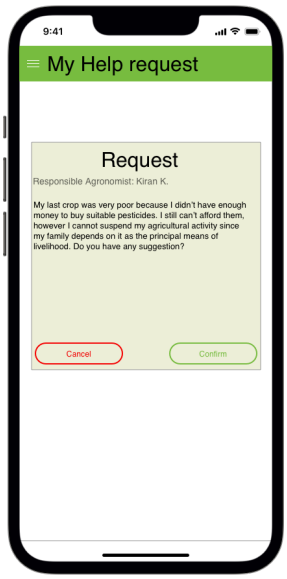

Figure 3.11: MobileApp Farmer Help Request page

Interface showing a farmer's help request with only agronomist as recipient.

MobileApp Agronomist Daily Plan page

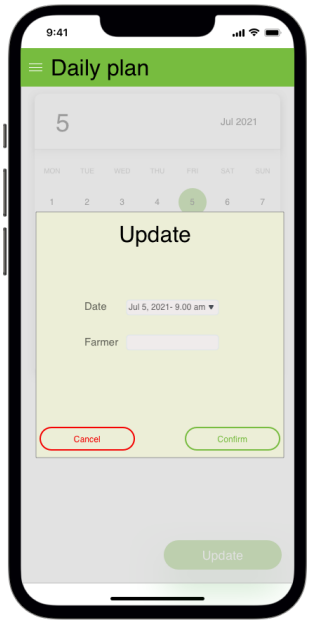

Figure 3.12: Mobile Agronomist Daily Plan

Interface that shows an update of the daily plan made by an agronomist.

MobileApp Farmer Notifications

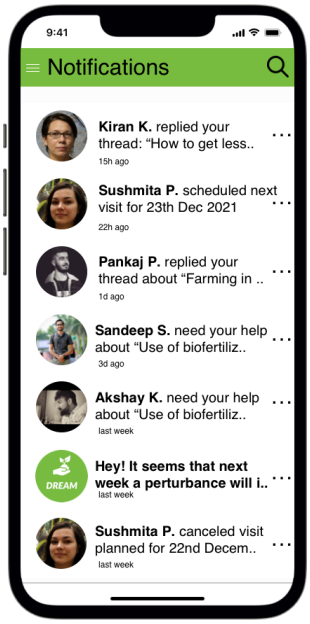

Figure 3.13: Mobile Farmer Notifications

Interface that shows various kind of notifications received by a farmer: suggestions, visits scheduled by the associated agronomist, reply to a help request, reply on the discussion forum.

3.1.2 Hardware Interfaces

All actors requires a compatible device (smartphone or personal computer) with Internet connection to allow DREAM to work properly.
In particular, for policy makers is highly recommended to access the application using a computer to maximize their experience while farmers should prefer to use a smartphone. Eventually agronomist must use both computer and smartphone using the application according to the functionalities they need to exploit.

3.1.3 Software Interfaces

The system relies on various external services accessible through API to compute and retrieve near-real time data:

- *GoogleMapsAPI* to retrieve the position (latitude and longitude) of farms' address inserted by farmers
- *Copernicus Climate Data Store Service* to retrieve data regarding soil moisture
- *Telangana Government Service* to retrieve weather forecasts
- *Telangana Water Irrigation System Service* to retrieve the amount of water consumed by each farmer

It also accesses ID codes' DB managed by Telangana government to check validity of ID code inserted by policy makers and agronomist while registering into DREAM.

3.1.4 Communication Interfaces

All devices used by actors are connected to the system through Internet connection.

3.2 Functional Requirements

Functional requirements capture the intended behavior of the system and are derived through Use Cases.
A use case defines a goal-oriented set of interactions between external actors and the system under consideration.

3.2.1 Use Case Diagrams

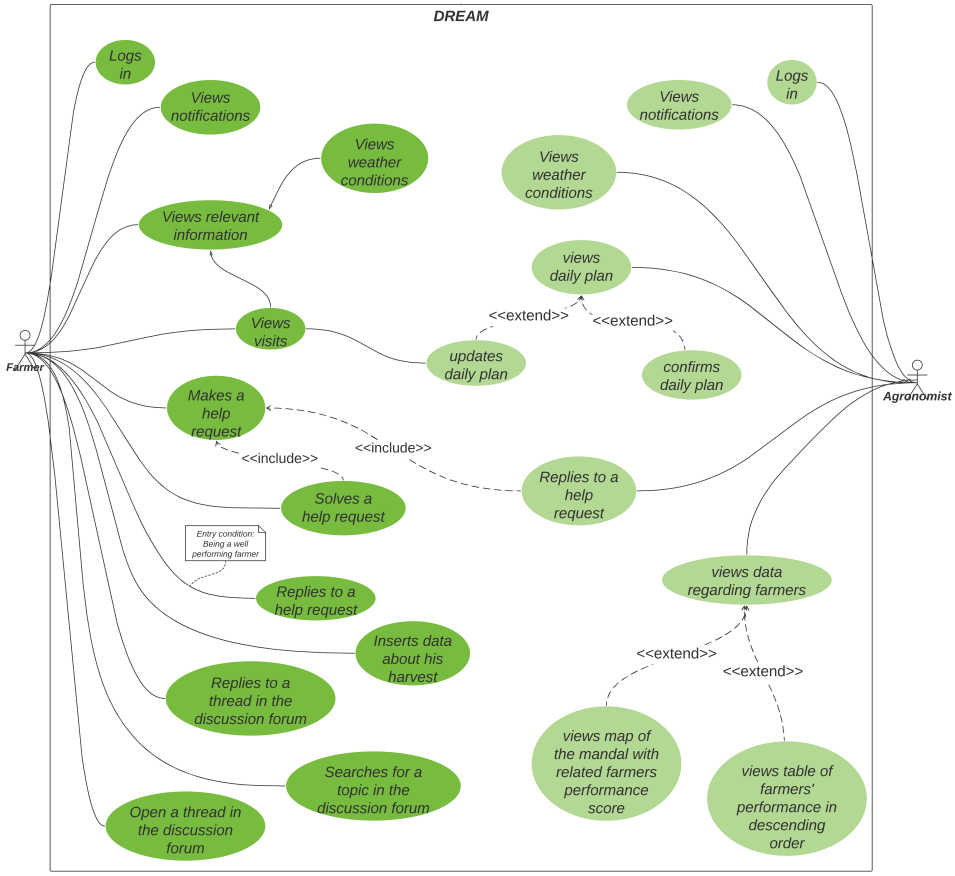

Figure 3.14: Use Case Diagram - Farmer and Agronomist

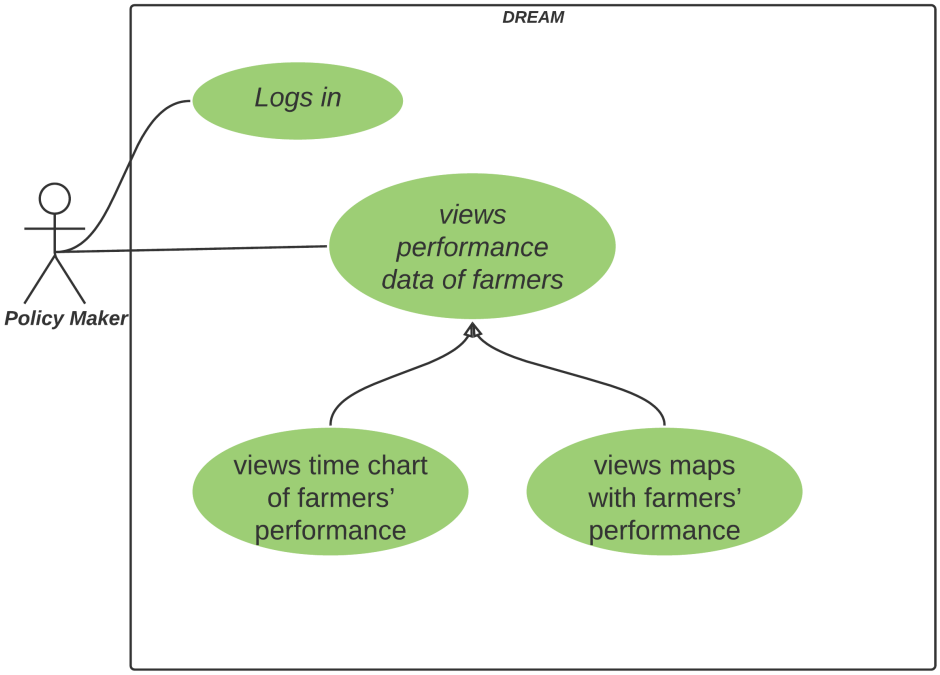

Figure 3.15: Use Case Diagram - Policy maker

3.2.2 Use Case Analysis

Name	UC.1 *Unregistered user registers into the application*
Actors	Unregistered user
Entry conditions	Unregistered user wants to join DREAM
Flow of events	1. The unregistered user accesses the *Registration* page on the application. 2. The unregistered user selects which category of users he/she belongs to: "Policy Maker", "Farmer" or "Agronomist". 3. The system provides the user with the appropriate fields to fill. 4. The unregistered user fills in all the mandatory fields. 5. The unregistered user accepts terms and conditions. 6. The unregistered user clicks on the "Register" button. 7. The system saves data of the new user. 8. The system sends an email to the new user. 9. The user is redirected on a waiting page that suggests him/her to check his/her mailbox. 10. The user confirms his/her email by clicking on the link in the email within ten minutes. 11. The user is redirected to the *Login* page.
Exit conditions	The unregistered user successfully ends the registration process and he/she becomes new user.
Exceptions	• The unregistered user inserts invalid information in the fields. • The unregistered user does not fill in all the required fields. • The unregistered user inserts a username that is not available. • The user does not confirm his/her email by clicking on the link within ten minutes.

Name	UC.2 *User logs in the application*
Actors	Farmer / Agronomist / Policy maker
Entry conditions	The user has already registered.
Flow of events	1. The user opens the *Login* page. 2. The user inserts his/her username. 3. The user inserts his/her password. 4. The user click on the "Login" button. 5. The system checks that the parameters entered correspond to those of a registered account. 6. The user is redirected to his/her *Home* page.
Exit conditions	The user successfully logs in and he/she is identified with the role (Farmer / Agronomist / Policy maker) with which he/she was previously registered.
Exceptions	• The user did not fill in both required fields. • The user has filled in one or both fields incorrectly.

Name	UC.3 *Farmer inserts data about product harvested into the application*
Actors	Farmer
Entry conditions	Farmer has already logged in.
Flow of events	1. The farmer clicks on the "+" button on his/her *Home* page. 2. The system shows a new form to fill out. 3. The farmer inserts the type of product of his/her harvest. 4. The farmer inserts the quantity of product of his/her harvest. 5. The farmer inserts notes about his/her harvest. 6. The farmer clicks on the "Add" button. 7. The system adds the parameter "username" of the farmer to the request. 8. The system adds the parameter "current date" to the request. 9. The system adds the parameter "quantity of water consumed" to the request. 10. The system updates farmer's performance. 11. The system refresh the *Home* page. 12. The system returns a message that the information has been added.
Exit conditions	• The farmer has successfully inserted his/her harvest information. • The farmer received the confirmation message for adding data on his/her crop.
Exceptions	The inserted data are not well formatted.

Name	UC.4 *Agronomist visualizes data regarding farmers*
Actors	Agronomist
Entry conditions	Agronomist has already logged in and wants to analyse data regarding farmers
Flow of events	1. The system shows the map of the mandal associated to the agronomist with all related farmers' performance scores. 2. The system shows the table of farmers' performance in ascending order based on their performance score. 3. The agronomist clicks on a certain performance score icon on the map. 4. The system provides him/her a pop up in which are listed name, phone number and performance score of that farmer.
Exit conditions	Data are shown and can be analysed by the agronomist.

Name	UC.5 *Policy maker visualizes data regarding farmers*
Actors	Policy maker
Entry conditions	Policy maker has already logged in and wants to analyse data regarding farmers
Flow of events	1. The system shows Telangana map in which are displayed all related farmers' most recent performance scores using different colours to differentiate bad and well performing farmers. 2. In the map section in the Home page the policy maker selects: (a) mandal he/she wants to visualize (by default mandal field is set *all*). (b) data he/she wants to visualize (by default data field is set *performance score*). (c) operation he/she wants to compute data with (by default operation field is set *null*). 3. The system shows the map according to the previously selected options. 4. The policy maker clicks on a certain data icon on the map. 5. The system provides him/her a pop up in which are listed name, performance score, type and quantity of product harvested and quantity of water consumed of that farmer.
Exit conditions	Data are shown and can be analysed by the policy maker.

Name	UC.6 *Policy maker analyzes data regarding farmers*
Actors	Policy maker
Entry conditions	Policy maker has already logged in and wants to analyse data regarding farmers.
Flow of events	1. In the time chart section in the *Home* page the policy maker selects: (a) mandal he/she wants to analyze (by default mandal field is set *all*); (b) data he/she wants to analyze (by default data field is set *performance score*); (c) operation he/she wants to compute data with (by default operation field is set *null*); (d) time interval, which is set *last year* by default; (e) whether to check only farmer with at least an help request solved or not (by default this field is not checked). 2. The system shows the time chart with selected data.
Exit conditions	Time chart is shown and can be analysed by the policy maker.

Name	UC.7 *Farmer makes a help request*
Actors	Farmer
Entry conditions	Farmer has already logged in.
Flow of events	1. The farmer accesses the *Help request* page by clicking on the appropriate icon on the *Home* page. 2. The farmer clicks on the "Create Help Request" button. 3. By default the system adds the responsible agronomist to the recipients of the help request. 4. The farmer decides whether to selects well performing farmers as recipients or not. 5. The farmer writes the help request in the appropriate field. 6. The farmer clicks on the "Confirm" button. 7. The system adds the parameter "id" to the request. 8. The system adds the parameter "username" of the farmer to the request. 9. The system adds the parameter "current date" to the request. 10. The system updates the status of the help request as created. 11. The system updates the *My Help Request* page. 12. The system returns a message that the help request has been created.
Exit conditions	The farmer has successfully send a help request.
Exceptions	The inserted data are not well formatted.

Name	UC.8 *Agronomist replies to a help request*
Actors	Agronomist
Entry conditions	Agronomist has already logged in and has already received a notification about a new help request received not yet solved.
Flow of events	1. The agronomist accesses the *Help Request* page by clicking on the appropriate icon on the *Home* page. 2. The system shows a list of help requests created and not yet solved. 3. The agronomist reads the details of the new help request received by clicking on it. 4. The agronomist clicks on the button "Reply" associated to the help request received. 5. The system shows the agronomist a form to be filled in with mandatory field *text*. 6. The agronomist fills in the form. 7. The agronomist confirms by clicking on the corresponding button. 8. The system adds the agronomist as responder to the response. 9. The system adds the farmer who made the help request as questioner to the response. 10. The system adds the current date to the response. 11. The system associates the response of the agronomist to the help request though the "id req". 12. The system updates the *Help Request* page. 13. The system returns a message that the agronomist has successfully replied to the help request.
Exit conditions	The agronomist has successfully replied to the help request.
Exceptions	The inserted data are not well formatted.

Name	UC.9 *Farmer replies to a help request*
Actors	Farmer
Entry conditions	Farmer has already logged in and his/her performance score computed by the system identifies him/her as a *Well performing farmer*. He/She has already received a notification about a new help request received not yet solved.
Flow of events	1. The farmer accesses the *Help Request* page by clicking on the appropriate icon on the *Home* page. 2. The system shows a list of help requests created and not yet solved sent to him/her being a well performing farmer. 3. The farmer reads the details of a help request received by clicking on it. 4. The farmer clicks on the button "Reply" associated to the help request received. 5. The system shows the farmer a form to be filled in with mandatory field *text*. 6. The farmer fills in the form. 7. The farmer confirms by clicking on the corresponding button. 8. The system adds the farmer as responder to the response. 9. The system adds the farmer who made the help request as questioner to the response. 10. The system adds the current date to the response. 11. The system associates the response to the corresponding help request though the id req. 12. The system updates the *Help Request* page. 13. The system returns a message that the farmer has successfully replied to the help request.
Exit conditions	The farmer has successfully replied to the help request.
Exceptions	The inserted data are not well formatted.

Name	UC.10 *Farmer solves a help request*
Actors	Farmer
Entry conditions	Farmer has already logged in and he/she has previously created a help request that is not yet marked as solved.
Flow of events	1. The farmer accesses the *Help Request* page by clicking on the appropriate icon on the *Home* page. 2. The farmer visualize his/her help requests by clicking on "My Help Request" button. 3. The system shows the list of all farmers' the help requests created but not solved with related answers. 4. The system shows two answer buttons "Yes" or "No" to the question "Are you satisfied with the answer?" when the farmer receives a response to a help request. 5. The farmer closes the request for help by declaring that he/she is satisfied with the answer received by clicking on the "yes" button. 6. The system updates the status of the help request to solved. 7. The system updates the *My Help Request* page. 8. The system returns a message that the help request has been solved.
Exit conditions	The farmer has successfully solved a help request.
Exceptions	• All the farmers' help requests are already solved. • No one has yet replied to the farmer's help request.

Name	UC.11 *Farmer finds a topic on the discussion forum*
Actors	Farmer
Entry conditions	Farmer has already logged in
Flow of events	1. The farmer accesses the *Discussion Forum* page by clicking on the appropriate icon on the *Home* page. 2. The farmer clicks on the corresponding button. 3. The system shows a form to be filled in with the mandatory field *topic*. 4. The farmer inserts the topic to be found. 5. The farmer submit the form.
Exit conditions	The system provides the farmer a list with all associated threads already existing on the discussion forum.
Exceptions	• There are no existing threads related to a certain topic on the discussion forum. In this case, the system suggests to attempt a new research or to open a new thread. • The inserted data are not well formatted.

Name	UC.12 *Farmer opens a thread on the discussion forum*
Actors	Farmer
Entry conditions	Farmer has already logged in.
Flow of events	1. The farmer accesses the *Discussion Forum* page by clicking on the appropriate icon on the *Home* page. 2. The farmer clicks on the "Open a thread" button. 3. The system shows a form to be filled in with the the mandatory fields *topic* and *text*. 4. The farmer inserts mandatory data. 5. The farmer confirms by clicking on the corresponding button. 6. The system adds the id to the thread. 7. The system adds the username of the farmer to the thread. 8. The system adds the current date to the thread. 9. The system updates the *Discussion Forum* page. 10. The system returns a message that the thread has been created on the discussion forum.
Exit conditions	The farmer has successfully opened a new thread.
Exceptions	The farmer does not fill out the form with the mandatory data.

Name	UC.13 *Farmer replies on the discussion forum*
Actors	Farmer
Entry conditions	Farmer has already logged in.
Flow of events	1. The farmer accesses the *Discussion Forum* page by clicking on the appropriate icon on the *Home* page. 2. The system shows a list of existing threads only visualizing the associated topic. 3. The farmer clicks on a certain thread. 4. The farmer visualizes its details and all the correlated answers already received. 5. The farmer clicks on the button "Reply". 6. The system shows a form to be filled in with the mandatory field *text*. 7. The farmer inserts mandatory data. 8. The farmer confirms by clicking on the corresponding button. 9. The system adds the username of the farmer to the post as responder. 10. The system adds the username of the farmer how opened the thread to the post as questioner. 11. The system adds the current date to the post. 12. The system associates the post of the farmer to the thread on the discussion forum adding the thread id. 13. The system updates the *Thread* page. 14. The system returns a message that the farmer has successfully answered to a thread on the discussion forum.
Exit conditions	The farmer has successfully answered to a thread on the discussion forum.
Exceptions	The farmer does not fill out the form with the mandatory data.

Name	UC.14 *Agronomist visualizes daily plan*
Actors	Agronomist
Entry conditions	Agronomist has already logged in.
Flow of events	1. The agronomist accesses the *Daily Plan* page by clicking on the appropriate icon on the *Home* page.
Exit conditions	The system shows the daily plan of the current day.

Name	UC.15 *Agronomist updates daily plan*
Actors	Agronomist
Entry conditions	Agronomist has already logged in
Flow of events	1. The agronomist accesses the *Daily Plan* page by clicking on the appropriate icon on the *Home* page. 2. The agronomist clicks on the "Update" button. 3. The system shows a form to be filled in with mandatory fields *date* and *farmer*. 4. The agronomist inserts mandatory data in corresponding fields. 5. The agronomist confirms by clicking on the corresponding button. 6. The system updates the status of the daily plan into updated. 7. The system updates the *Daily Plan* page. 8. The system returns a message that the agronomist has successfully updated his/her daily plan.
Exit conditions	The agronomist has successfully updated the daily plan.
Exceptions	• The agronomist does not fill out the form with the mandatory data. • The date entered by the agronomist in the corresponding field refers to a day in the past. • The agronomist entered by the agronomist in the corresponding field doesn't refer to any of those under his/her responsibility.

Name	UC.16 *Agronomist confirms daily plan*
Actors	Agronomist
Entry conditions	Agronomist has already logged in and has already carried out the current daily plan.
Flow of events	1. The agronomist accesses the *Daily Plan* page by clicking on the appropriate icon on the *Home* page. 2. The agronomist clicks on the "Confirm the execution" button. 3. The system shows a form to be filled in with optional field *deviations*. 4. The agronomist fills in the form. 5. The agronomist confirms by clicking on the corresponding button. 6. The system updates the status of the daily plan into done. 7. The system updates the *Daily Plan* page. 8. The system returns a message that the agronomist has successfully confirmed his/her daily plan.
Exit conditions	The agronomist has successfully confirmed the daily plan.

Name	UC.17 *Agronomist visualizes weather forecasts and soil moisture of his/her associated mandal*
Actors	Agronomist
Entry conditions	Agronomist has already logged in.
Flow of events	1. The agronomist accesses the *Weather conditions* page by clicking on the associated widget on the *Home* page. 2. The system shows weather conditions and soil moisture of the current date on the mandal map. 3. The agronomist selects the day (up to seven day after the current date) for which he/she wants to visualize weather forecasts.
Exit conditions	The system shows weather forecasts on the mandal map.
Exceptions	The agronomist selects a not valid day.

Name	UC.18 *Farmers visualizes relevant information*
Actors	Farmer
Entry conditions	Farmer has already logged in.
Flow of events	1. The system shows on the *Home* page weather conditions and soil moisture regarding farmer's position and the next visit scheduled by the associated agronomist. 2. The farmer accesses the *Visit* page by clicking on the associated icon on the *Home* page. 3. The system shows the list of scheduled visits (even the ones that have already occurred in the past) in chronological order.
Exit conditions	The farmer has successfully visualized relevant information regarding him/her.

Name	UC.19 *Agronomist receives notifications*
Actors	Agronomist
Entry conditions	Agronomist has already logged in and a new help request has been created and not yet solved.
Flow of events	1. The agronomist receives a notification from the system informing him/her that a help request has been created. 2. The agronomist clicks on the notification icon. 3. The system redirects the agronomist to the *Notification* page and displays a list of help requests received in chronological order and the details associated to the notifications received. 4. The agronomist clicks on the notification to know the details of the new help request received.
Exit conditions	Agronomist is aware of the presence of a new help request.

Name	UC.20 *Farmer receives notifications*
Actors	Farmer
Entry conditions	Farmer has already logged in and at least one of the following events has occurred: • The system has computed suggestions according to his/her relevant information regarding his/her production and his/her farm's position. • New visits have been scheduled in the daily plan regarding him/her. • Someone has replied to his/her own help requests not yet solved. • Someone has replied to his/her own threads opened on the Discussion Forum.
Flow of events	1. The farmer receives a notification from the system. 2. The farmer clicks on the notification icon. 3. The system redirects the farmer to the Notification page and displays a list of notifications received in chronological order and the details associated to the notifications received. 4. The farmer clicks on the notification to know the details of it. 5. The system returns the details associated to the notification received.
Exit conditions	Farmer is aware of the presence of a new notification.

3.2.3 Sequence Diagrams

The following UML sequence diagrams depict the objects involved in the main use cases previously described and the sequence of interactions between them and the system needed to carry out their functionalities.

In order to ensure greater understanding of what could be a complete real sequence of interactions among user and system, not necessarily correlated use cases have been taken under consideration while building the corresponding sequence diagrams.
In particular, the discussion forum sequence diagram implies a farmer searching for a topic (UC.11) and if he/she cannot find any thread already existing, he/she opens a new thread (UC.12). Moreover, as shown in the daily plan sequence diagram, an agronomist visualizes (UC.14) and then he/she up-

dates (UC.15) the daily plan.
The mentioned use cases belonging to the corresponding sequence diagram are not necessarily related as previously pointed out, however connecting them allows a complete depiction of related product functions (discussion forum and daily plan).

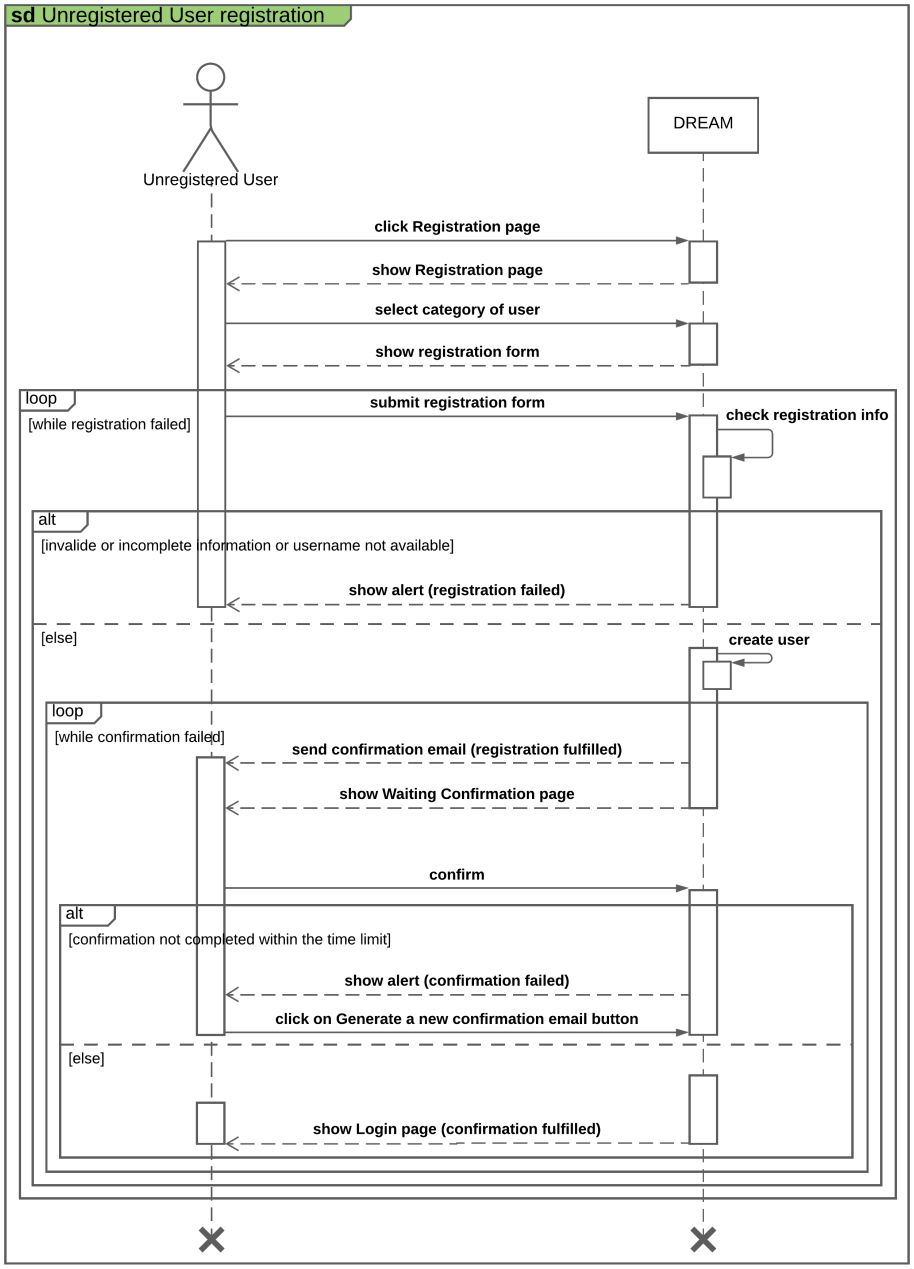

Figure 3.16: Sequence Diagram - Unregistered User Registration

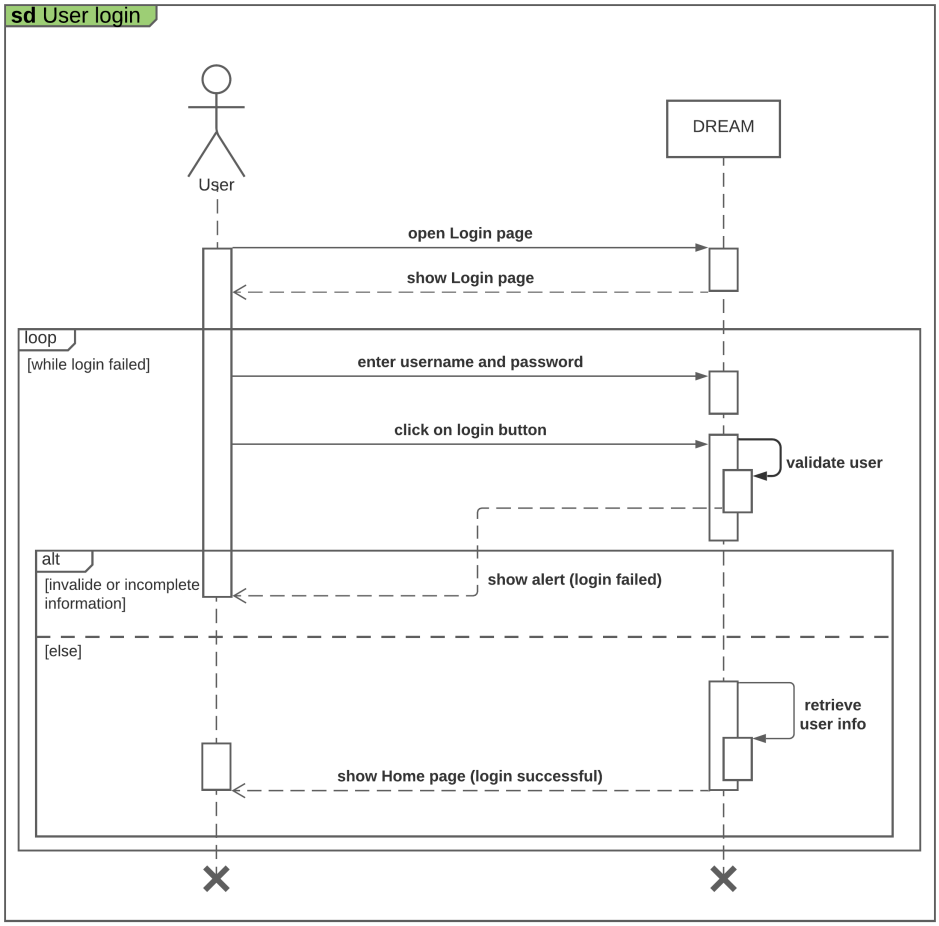

Figure 3.17: Sequence Diagram - User Login

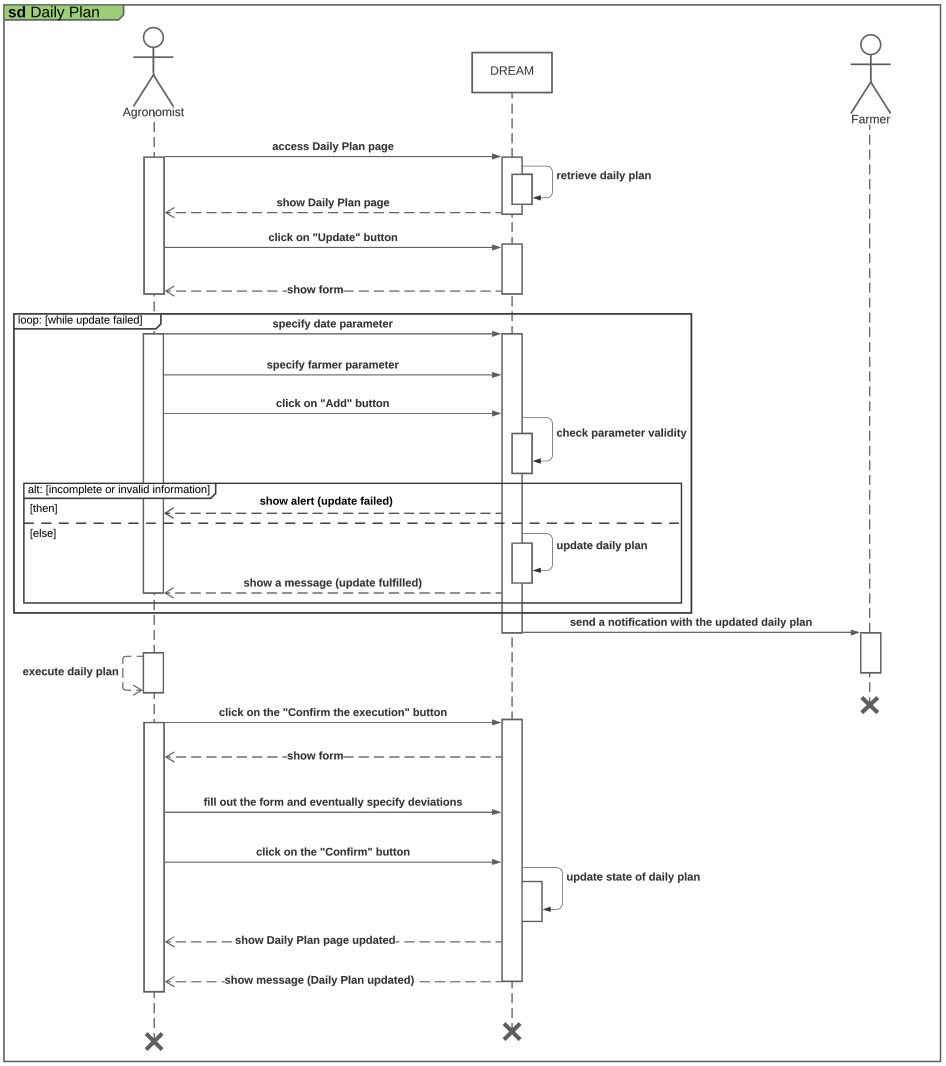

Figure 3.18: Sequence Diagram - Daily Plan

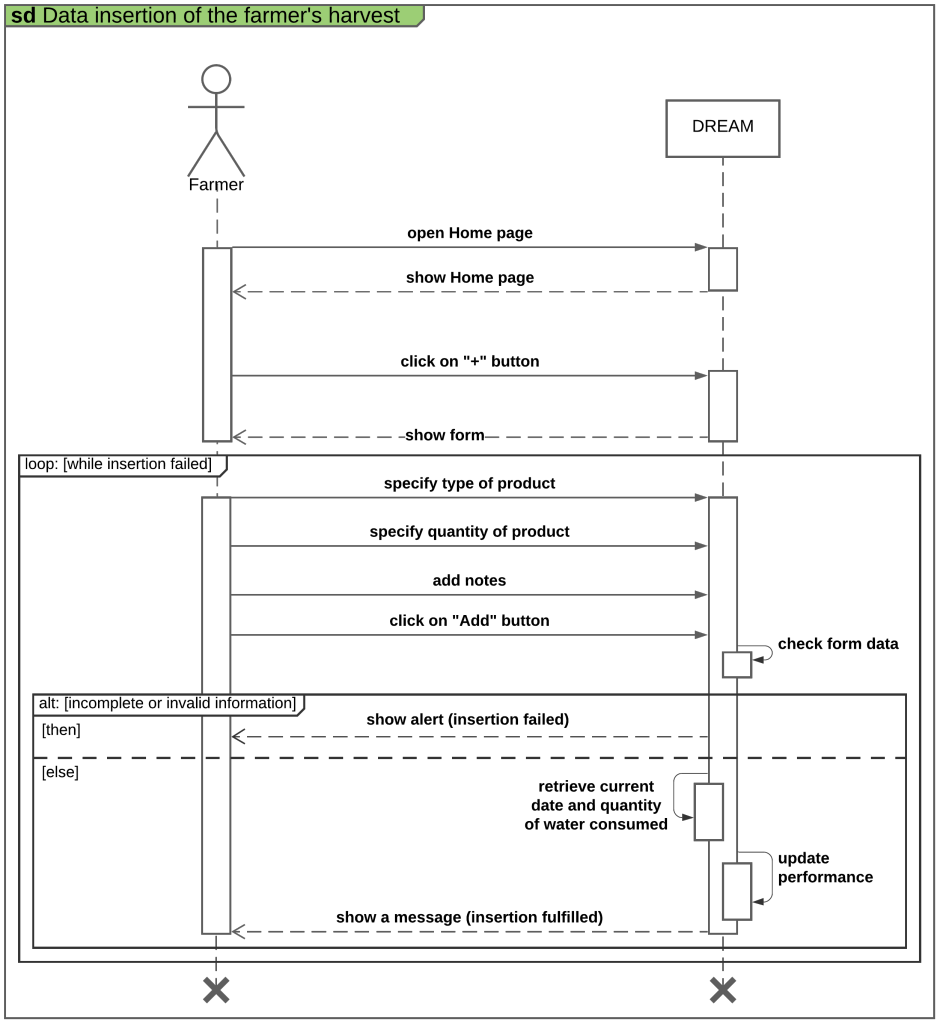

Figure 3.19: Sequence Diagram - Data insertion of the farmer's harvest

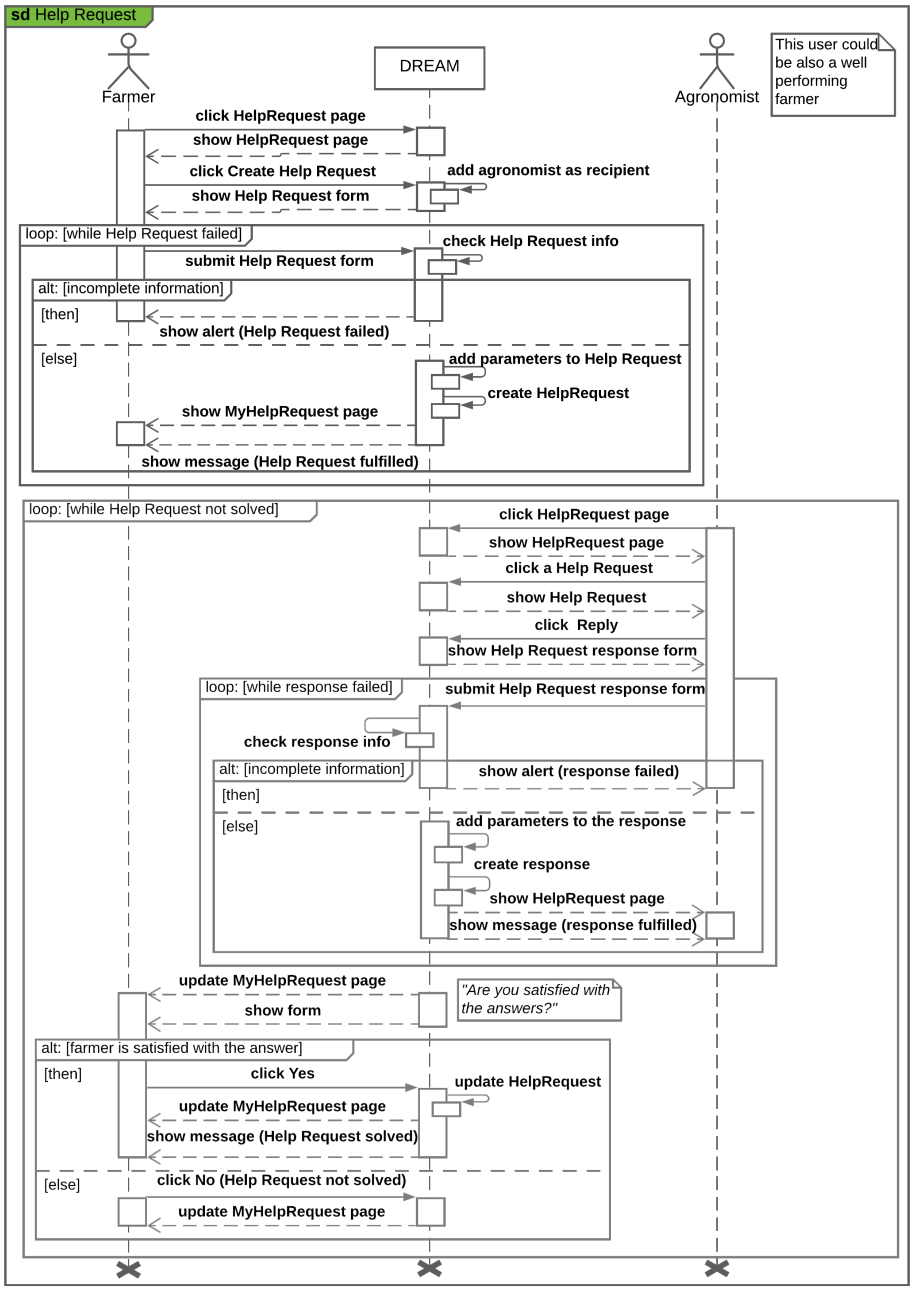

Figure 3.20: Sequence Diagram - Help Request

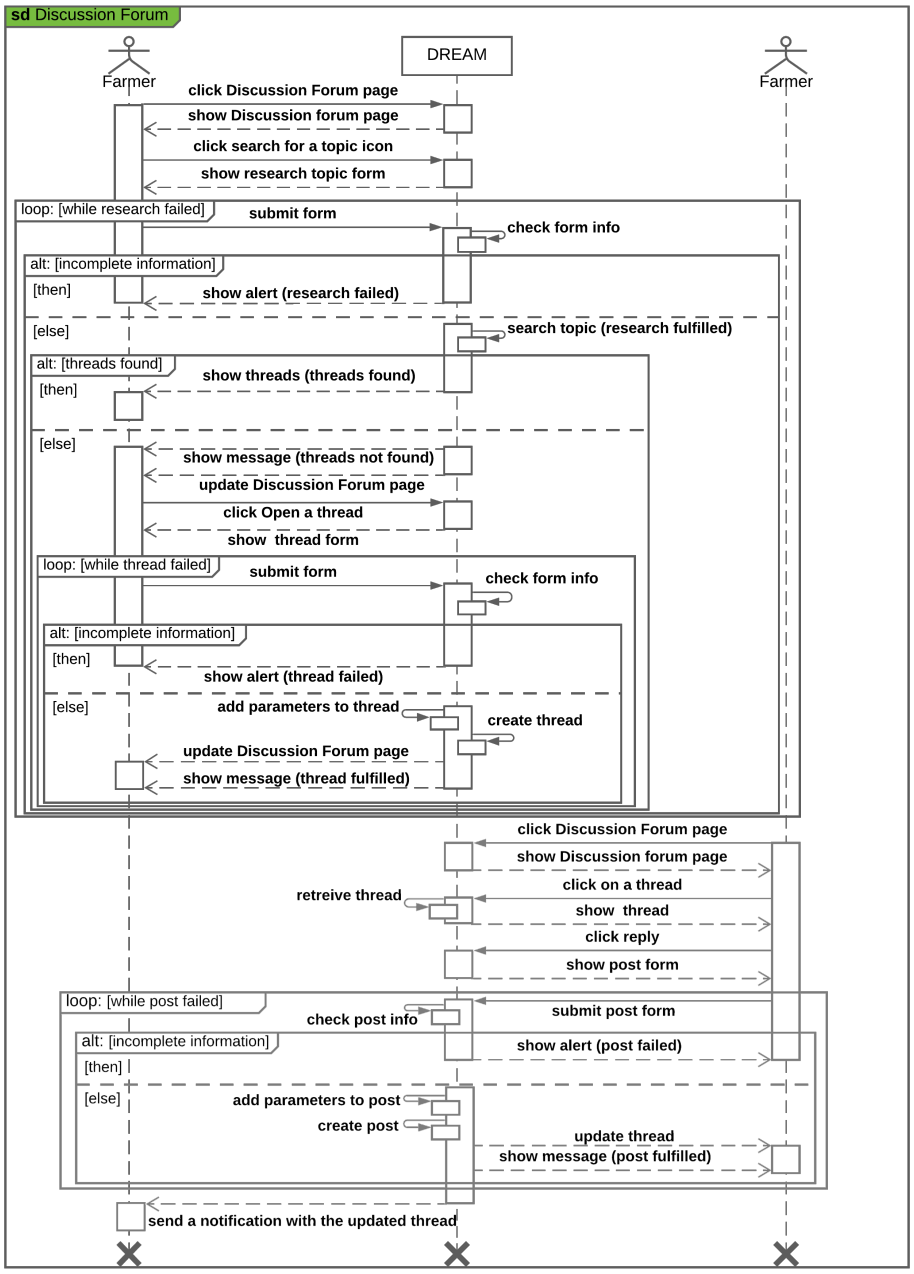

Figure 3.21: Sequence Diagram - Discussion Forum

3.2.4 Requirements

In this section are listed all the requirements needed to achieve the goals previously described in *Section 1.1.1*.

R.1 **Unregistered user must be able to register to DREAM**

- *R.1.1* Unregistered user must be able to select his/her role
- *R.1.2* Unregistered user must be able to fill out the registration form

R.2 **User must be able to login to DREAM by filling out the login form**

R.3 **The system must be able to associate each farmer to the corresponding mandal according to his/her position**

R.4 **Farmer must be able to fill out the form for data insertion**

- *R.4.1* Farmer must be able to insert the type of product of his/her harvest
- *R.4.2* Farmer must be able to insert the quantity of product of his/her harvest
- *R.4.3* Farmer must be able to insert note about his/her harvest

R.5 **The system must be able to add parameters to data inserted by farmers regarding their production**

- *R.5.1* The system must be able to add the current date to the inserted data
- *R.5.2* The system must be able to add the quantity of water consumed to the inserted data

R.6 **The system must be able to return whether a data insertion has been done correctly by the user or not**

- *R.6.1* The system must be able to return a message in case of successful data insertion
- *R.6.2* The system must be able to return an alert in case of failed data insertion

R.7 **The system must be able to classify farmers into bad and well performing farmers according to their performance score value**

- *R.7.1* The system must be able to compute performance score of each farmer

R.8 **The system must be able to show the map with performance score regarding farmers**

R.8.1 The system must be able to identify the farmer's address on the map

R.8.2 The system must to able to upload most recent computed performance score on the map

R.8.3 The system must be able to visualize on the map the color associated with each farmer based on his/her performance score

R.8.4 The system must be able to show the farmers information linked to a certain icon in the map

R.9 **The policy maker must be able to select options which he/she wants to update the map with**

R.9.1 The policy maker must be able to indicate the mandal he/she wants to visualize

R.9.2 The policy maker must be able to indicate the data regarding farmers he/she wants to visualize

R.9.3 The policy maker must be able to indicate the operation he/she wants to compute data with

R.10 **The system must be able to show the updated map according to the option chosen by policy maker.**

R.11 **The policy maker must be able to select options for the time chart**

R.11.1 The policy maker must be able to indicate the mandal he/she wants to analyze

R.11.2 The policy maker must be able to indicate the data regarding farmers he/she wants to analyze

R.11.3 The policy maker must be able to indicate the operation he/she wants to compute data with

R.11.4 The policy maker must be able to indicate the time interval

R.11.5 The policy maker must be able to visualize data about farmers with help request solved

R.12 **The system must be able to show the time chart according to the option chosen by policy maker.**

R.13 **The system must be able to compute data according to the chosen operation**

R.14 **The system must be able to show a table with farmers' data in descending order based on their performance score**

R.15 **The system must be able to show the map of mandal's agronomist with weather conditions and soil moisture of the current date**

R.16 **The agronomist must be able to visualize weather forecasts up to seven days after the current date**

R.16.1 The agronomist must be able to select the day for which he/she wants to visualize weather forecasts

R.16.2 The system must be able to show the updated mandal map with weather forecasts regarding the day selected by the agronomist

R.17 **The system must be able to show the daily plan of the current day**

R.17.1 The system must be able to compute the daily plan assigning two visits per year to each farmer

R.17.1.1 The system must be able to set the status of the daily plan into created in the database

R.18 **The agronomist must be able to update the daily plan**

R.18.1 The agronomist must be able to fill out a form to update the daily plan

R.18.1.1 The agronomist must be able to insert in the form the date of the day he/she wants to update in the daily plan

R.18.1.2 The agronomist must be able to insert in the form the farmer related to the visit which the agronomist is updating the daily plan with

R.18.2 The system must be able to add parameters to the updating visit

R.18.2.1 The system must be able to add the full name of the agronomist to the updating visit

R.18.3 The system must be able to change the status of the daily plan into updated in the database

R.19 **The agronomist must be able to confirm the daily plan**

R.19.1 The agronomist must be able to fill out a form to confirm the daily plan

R.19.1.1 The agronomist must be able to insert in the form deviations from the plan

R.19.2 The system must be able to change the status of the daily plan into done in the database

R.20 **The system must be able to show weather conditions and soil moisture regarding farmer's position**

R.21 **The farmer must be able to visualize all visits scheduled for him/her**

R.21.1 The system must be able to show all scheduled visits in chronological order (even ones that have already occurred in the past)

R.21.2 The system must be able to show the next visit scheduled by the associated agronomist

R.22 **The system must be able to send notifications to the farmer**

R.22.1 The system must be able to send notification to the farmer regarding personalized suggestions

R.22.1.1 The system must be able to compute suggestions according to the information inserted by the farmer and his/her position

R.22.2 The system must be able to send notification to the farmer regarding new visits scheduled in the daily plan regarding him/her

R.22.3 The system must be able to send notification to the farmer regarding new answers to his/her created help requests not yet solved

R.22.4 The system must be able to send notification to the farmer regarding new posts associated to his/her threads on the discussion forum

R.23 **The farmer must be able to visualize notifications' details**

R.23.1 The farmer must be able to select the notification whose details he/she wants to visualize

R.23.2 The system must be able to show the details regarding the notification selected by the farmer

R.24 **The system must be able to show a list of the notifications received in chronological order**

R.25 **The farmer must be able to request for help**

R.25.1 The farmer must be able to fill out a form to create a help request

R.25.1.1 The farmer must be able whether to select well performing farmer as recipients of the help request or not

R.25.1.2 The farmer must be able to write the body of the help request in the appropriate field

R.25.2 The system must be able to add parameters to the help request

R.25.2.1 The system must be able to add agronomist as recipient of the help request by default

R.25.2.2 The system must be able to add the username of the farmer to the help request

R.25.2.3 The system must be able to add the current date to the help request

R.25.2.4 The system must be able to add the status created to the help request

R.25.2.5 The system must be able to add the parameter "id" to the request

R.26 **The system must be able to show to well performing farmers a list of the help requests created and not yet solved with them as recipient.**

R.27 **The well performing farmer must be able to select a help request to read its details**

R.28 **The system must be able to show the details of the help request selected by the well performing farmers**

R.29 **The well performing farmer must be able to answer to a help request**

R.29.1 The well performing farmer must be able to fill out a form to reply to the help request

R.29.2 The well performing farmer must be able to insert in the form the text of the response

R.29.3 The system must be able to add parameters to the response

R.29.3.1 The system must be able to add to the response the username of the farmer who created the help request as questioner

R.29.3.2 The system must be able to add the username of the well performing farmer to the response as responder

R.29.3.3 The system must be able to add the current date to the response

R.29.3.4 The system must be able to associate the response to the corresponding help request by adding the id req

R.30 **The system must be able to notify the agronomist when a new help request is created and not yet solved**

R.31 **The agronomist must be able to visualize notification's details**

R.31.1 The agronomist must be able to select the notification whose details he/she wants to visualize

R.31.2 The system must be able to show the details regarding the notification selected by the agronomist

R.32 **The system must be able to show to the agronomist a list of the help requests created and not yet solved with him/her as recipient.**

R.33 **The agronomist must be able to visualize help requests' details**

- *R.33.1* The agronomist must be able to select a help request to read its details
- *R.33.2* The system must be able to show the details of the help request selected by the agronomist

R.34 **The agronomist must be able to answer to a help request**

- *R.34.1* The agronomist must be able to fill out a form to reply to the help request
  - *R.34.1.1* The agronomist must be able to insert in the form the text of the response
- *R.34.2* The system must be able to add parameters to the response
  - *R.34.2.1* The system must be able to add to the response the full name of the farmer who created the help request as questioner
  - *R.34.2.2* The system must be able to add the full name of the agronomist to the response as responder
  - *R.34.2.3* The system must be able to add the current date to the response
  - *R.34.2.4* The system must be able to associate the response to the corresponding help request by adding the id req

R.35 **The farmer must be able to solve a help request**

- *R.35.1* The system must be able to show the list of all the help requests of the farmer created, but not solved
- *R.35.2* The system must be able to show a message regarding the level of satisfaction of the farmer when a help request is replied
- *R.35.3* The farmer must be able to select an answer
- *R.35.4* The system must be able to change the status of the help request into solved in the database
- *R.35.5* The system must be able to return a message in case of help request solved

R.36 **The system must be able to show a list of existing threads only visualizing the associated topic**

R.37 **The farmer must be able to visualize a certain thread with related posts**

R.37.1 The farmer must be able to select an already existing thread

R.37.2 The system must be able to show the selected thread with all associated posts

R.38 **The farmer must be able to search for a topic on the discussion forum**

R.38.1 The farmer must be able to fill out a form to search for a topic in the forum

R.38.1.1 The farmer must be able to insert in the form the topic to be searched

R.38.2 The system must be able to show a list of already existing threads associated to the searched topic

R.38.3 The system must be able to show a message in case of no existing threads associated to the searched topic, to suggest to attempt a new research or to open a new thread

R.39 **The farmer must be able to open a thread on the discussion forum**

R.39.1 The farmer must be able to fill out a form to open a thread on the discussion forum

R.39.1.1 The farmer must be able to insert in the form the topic of the thread

R.39.1.2 The farmer must be able to insert in the form the text of the thread

R.39.2 The system must be able to add parameters to the thread

R.39.2.1 The system must be able to add the id to the thread

R.39.2.2 The system must be able to add the username of the farmer to the thread

R.39.2.3 The system must be able to add the current date to the thread

R.40 **The farmer must be able to answer on the discussion forum**

R.40.1 The farmer must be able to fill out a form to reply to a thread

R.40.1.1 The farmer must be able to insert in the form the text of the post

R.40.2 The system must be able to add parameters to the post

R.40.2.1 The system must be able to add the username of the farmer to the post as responder

R.40.2.2 The system must be able to add to the post the username of the farmer how opened the thread as questioner

R.40.2.3 The system must be able to associate the post of the farmer to the thread on the discussion forum adding the thread id.

R.40.2.4 The system must be able to add the current date to the post

3.2.5 Traceability Matrix

In this section is reported a detailed mapping between requirements defined in *Section 3.2.4* and goals of the application listed in *Section 1.1.1*, which are achieved under the domain assumptions defined in *Section 2.4.1*.

Table 3.1: Traceability Matrix

G.1 Allows policy makers to visualize and analyze data of farmers									
Domain Assumptions					**Requirements**				
D1	D2	D3	D4	D5	R1	R2	R3	R4	R5
D6	D8	D9	D11		R6	R7	R8	R9	R10
					R11	R12	R13		
G.2 Allows policy makers to identify those farmers who are performing well and those who are performing bad									
Domain Assumptions					**Requirements**				
D1	D2	D3	D4	D5	R1	R2	R3	R4	R5
D6	D8	D9	D11		R6	R7	R8		
G.3 Allows policy makers to verify the improvement of farmers who have been already helped by agronomist or good farmer									
Domain Assumptions					**Requirements**				
D1	D2	D3	D4	D5	R1	R2	R3	R4	R5
D6	D8	D9	D11		R6	R7	R11	R12	R13
G.4 Allows farmers to visualize weather conditions									
Domain Assumptions					**Requirements**				
D1	D2	D5	D6	D8	R1	R2	R3	R6	R20
G.5 Allows farmers to visualize their next visits									
Domain Assumptions					**Requirements**				

D1	D5	D6	D8	D10	R1	R2	R3	R6	R17.1
D11	D12	D13	D14	D15	R18	R19	R21		

G.6 Allows farmers to receive notifications

Domain Assumptions					**Requirements**				
D1	D2	D3	D4	D5	R1	R2	R3	R4	R5
D6	D7	D8	D10	D11	R6	R7	R18	R22	R23
D12	D13				R24	R25	R29	R34	R39
					R40				

G.7 Allows farmers to insert in the system data about their production and any problem they face

Domain Assumptions					**Requirements**				
D1	D2	D3	D4	D5	R1	R2	R4	R5	R6
D6	D8								

G.8 Allows farmers to receive help from agronomist and other well performing farmers through help requests

Domain Assumptions					**Requirements**				
D1	D5	D6	D7	D8	R1	R2	R3	R6	R7
D10	D11	D12			R22.3	R23	R25	R26	R27
					R28	R29	R30	R31	R32
					R33	R34	R35		

G.9 Allow farmers to interact on the discussion forum

Domain Assumptions					**Requirements**				
D1					R1	R2	R6	R22.4	R23
					R36	R37	R38	R39	R40

G.10 Allows agronomists to receive notifications regarding new help requests created									
Domain Assumptions					**Requirements**				
D1	D5	D6	D7	D8	R1	R2	R3	R6	R24
D10	D11	D12			R25	R30	R31	R32	R33
					R35				
G.11 Allows agronomists to visualize data concerning farmers in the mandal									
Domain Assumptions					**Requirements**				
D1	D2	D3	D4	D5	R1	R2	R3	R4	R5
D6	D8	D10	D11	D12	R6	R7	R8	R14	
G.12 Allows agronomists to visualize weather conditions in the mandal									
Domain Assumptions					**Requirements**				
D1	D2	D10	D11	D12	R1	R2	R6	R15	R16
G.13 Allows agronomists to visualize their own daily plan to visit farms in the mandal									
Domain Assumptions					**Requirements**				
D1	D5	D6	D8	D10	R1	R2	R3	R6	R17
D11	D12								
G.14 Allows agronomists to manage their own daily plan									
Domain Assumptions					**Requirements**				
D1	D5	D6	D8	D10	R1	R2	R3	R6	R18
D11	D12	D13	D14	D15	R19				

3.3 Performance Requirements

In order to evaluate the performance of DREAM system it is necessary to specify that DREAM must maintain a response time of 1 second at most to allow all users to seamlessly interact with the application when needed. Data obtained from external resources such as connections to external databases must also have adequate response times, less than 2 seconds.
The system should be capable of supporting 250'000 users (workload). In fact, considering all 288 mandal of Telangana, 400 farmers per mandal, 269 working day per year and 3 visits maximum per day for each agronomist, it results 115200 farmers and 288 agronomist. The number of policy makers using the application cannot be exactly defined, therefore the most reasonable amount of users has been chosen.
Partitioning techniques and replicated databases are used to make the system scalable.

3.4 Design Constraints

This section indicates the design constraints that the application must respect.

3.4.1 Standards compliance

All sensitive data entered by the user are saved in the app's internal database. The analysis of such data is only possible within the application by registered users with their respective roles, based on the relative permissions.

3.4.2 Hardware limitations

Users who intends to register on DREAM and then use it must have at least one of the following devices:

- Smartphone on which there is the possibility to install the DREAM app;
- Computer on which there is the possibility to access DREAM from the browser.

Moreover, the sensors installed on the territory are necessary to retrieve the data relating to the soil.

For the correct functioning of the system, the devices of the users must have a stable internet connection.

3.4.3 Any other constraint

There are currently no other constraints. They will be added in later versions of this document if needed.

3.5 Software System Attributes

The software system attributes used to evaluate the performance of a system are listed below. DREAM is designed to respond to all the following attributes in order to ensure safe, easy and continuous use over time.

3.5.1 Reliability

It is the ability of a system or component to function under the conditions established for a given period of time, that is, continuity of the correct service. Therefore, the system must be fault tolerant and robust.

3.5.2 Availability

It refers to the system's ability to be available for use, especially after a failure has occurred. The fault must be recognized and then the system must respond in some way.
The system should be available for 99% of the time, which implies a maximum of 3.65 day of downtime per year and 7.31 hours of downtime per month.

3.5.3 Security

Attacks against a system can compromise the confidentiality, integrity or availability of a system or its data. System security includes the development and implementation of security countermeasures.
The system is provided with firewalls, encrypted communication protocols, hashed passwords. In addition, only authorized agronomists and policy maker can access DREAM by entering a unique ID code given to them by Telangana government.

3.5.4 Maintainability

It is the ease with which a product can be maintained to isolate and correct defects, prevent unexpected failures, meet new needs.
In order to guarantee maintainability and future updates and integration,the system should be divided in modules and be clearly documented.

3.5.5 Portability

Portability refers to the ease with which software that was created to run on one platform can be modified to run on a different platform.
Web app is platform independent, therefore it is only needed a modern web browser to access DREAM.
Mobile application instead is available for two different operating systems: iOS and Android.

Chapter 4

Formal Analysis using Alloy

4.1 Alloy Model

In this chapter the critical and most complex parts of the system are analyzed through Alloy model.
In particular the analysis focuses on the following components of the system:

- Discussion forum
- Daily plan
- Help request (with related responses)

In the following model, these components are described specifying properties and constraints to validate the consistency of the corresponding generated worlds.
Some signatures and facts have been introduced or simplified in order to build the model, improving readability without losing accuracy.

4.1.1 Signatures

This section shows the piece of code in Alloy comprising all signatures needed to build the model.

```
module util/boolean

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

//A Mandal in Telangana
sig Mandal {
agronomist: one Agronomist,
farmer: some Farmer }

//A Farmer belonging to a certain Mandal
sig Farmer {
mandal: one Mandal,
```

```
helpRequest: set HelpRequest,
visit: some Visit,
thread: set Thread,
post: set Post }

// There exists exactly one Discussion Forum
one sig Forum {
thread: set Thread }

// A Thread in the Discussion Forum
sig Thread {
id: one Int,
questionner: one Farmer,
post : set Post } { id > 0 }

// A Post in a Thread
sig Post  {
id: one Int,
responder: one Farmer } { id > 0 }

//An Agronomist responsible of a certain Mandal
sig Agronomist {
mandal: one Mandal,
helpResponse: set HelpResponse,
dailyPlan: some DailyPlan }

//A DailyPlan managed by a certain Agronomist
sig DailyPlan {
agronomist:one Agronomist,
day: one Day,
state: one DailyPlanState,
visit: some Visit,
deviation: lone Deviation }

sig Deviation { }

abstract sig DailyPlanState { }
//To avoid logical error due to the homonymy with HelpRequestState,
    ↪ DailyPlanState CREATED has been modified into GENERATED
one sig GENERATED extends DailyPlanState { }
one sig UPDATED extends DailyPlanState {
selectedHour: one Hour,
updateHour: lone Hour,
updateDay: lone Day }
one sig CONFIRMED extends DailyPlanState { }

sig Day { }
sig Hour { }

//A Visit belonging to a certain DailyPlan
sig Visit {
day: one Day,
hour: one Hour,
farmer: one Farmer,
dailyPlan: one DailyPlan,
status: one Status}

abstract sig Status { }

one sig SCHEDULED extends Status { }
one sig DONE extends Status { }
```

```
//A Farmer who can reply to a HelpRequest if indicated as recipient
sig WellPerformingFarmer {
helpResponse: set HelpResponse }

//A HelpRequest made by a Farmer
sig HelpRequest {
id: one Int,
questionner: one Farmer,
state: one HelpRequestState,
recipient: one Recipient,
wellPerformingFarmerAsRecipient: one Bool,
helpResponse:some HelpResponse } { id > 0 }

abstract sig HelpRequestState { }

one sig CREATED, SOLVED extends HelpRequestState { }

//A Recipient of a HelpRequest
sig Recipient {
agronomist: one Agronomist,
wellPerformingFarmer: set WellPerformingFarmer }

//A HelpResponse to a HelpRequest
sig HelpResponse {
id: one Int,
responder: one Responder } { id > 0 }

//A Responder of a HelpResponse
sig Responder {
agronomist: lone Agronomist,
wellPerformingFarmer: lone WellPerformingFarmer}
```

4.1.2 Facts

In this section is shown the piece of code in Alloy that contains facts.

```
//RELATION BETWEEN AGRONOMIST AND MANDAL

//Mandal is associated to a unique Agronomist
fact { all disj m1, m2: Mandal | m1.agronomist ≠ m2.agronomist }

//Agronomist is associated to a unique Mandal
fact { all disj a1, a2: Agronomist | a1.mandal ≠ a2.mandal }

//Mandal and Agronomist are related
fact { all a: Agronomist, m: Mandal |  a.mandal = m implies m.
    ↪ agronomist = a }

//RELATION BETWEEN MANDAL AND FARMER

//Two Farmer (or more) can be associated to the same Mandal
fact { some disj f1, f2: Farmer | f1.mandal = f2.mandal }

//Mandal and Farmer are related
fact { all f: Farmer, m: Mandal |  f.mandal = m implies f in m.farmer
    ↪ }
fact { all f: Farmer, m: Mandal |  f in m.farmer implies  f.mandal = m
    ↪  }
```

```
//DISCUSSION FORUM

//For each Farmer the set of Thread must be different
fact { all disj f1, f2: Farmer | f1.thread & f2.thread = none }

//For each Farmer the set of Post must be different
fact { all disj f1, f2: Farmer | f1.post & f2.post = none }

//For each Thread the set of Post must be different
fact { all disj t1, t2: Thread | t1.post & t2.post = none }

//The responder must not be the Farmer who made the thread
fact { all t: Thread, p: Post | t.id = p.id implies p.responder ≠ t.
    ↪ questionner }

//A Thread belongs to the Thread set of the Thread questionner
fact { all t: Thread |  t in t.questionner.thread }

//A Post belongs to the Post set of the Post responder
fact { all p: Post |  p in p.responder.post }

//All Thread belong to one Forum
fact { all t: Thread, df: Forum |  t in df.thread }

//Each Thread is associated to a unique id
fact uniqueThread{ all disj t1, t2: Thread | t1.id ≠ t2.id }

//A Post cannot exist without an associated Thread
fact { all p: Post, t: Thread| p.id = t.id implies p in t.post }

//Each Post must be related to the corresponding Thread
fact { no p: Post | all t: Thread | p.id ≠ t.id }

//Two Post (or more) can be associated to the same Thread
fact { some disj p1, p2: Post | p1.id = p2.id }

//Two Thread (or more) can have the same questionner
fact { some disj t1, t2: Thread |  t1.questionner = t2.questionner }

//DAILY PLAN

//For each Agronomist the set of DailyPlan must be different
fact { all disj a1, a2: Agronomist | a1.dailyPlan & a2.dailyPlan =
    ↪ none }

//For each Farmer the set of Visit must be different
fact { all disj f1, f2: Farmer | f1.visit & f2.visit = none }

//For each DailyPlan the set of Visit must be different
fact { all disj d1, d2: DailyPlan | d1.visit & d2.visit = none }

//A DailyPlan cannot exist without the associated Agronomist
fact { all a: Agronomist, d: DailyPlan | d.agronomist = a implies d in
    ↪  a.dailyPlan }

//DailyPlan is associated to a unique Agronomist
fact { all disj d1, d2: DailyPlan | d1.agronomist ≠ d2.agronomist }

//A Visit is associated to a DailyPlan
fact { all v: Visit, d: DailyPlan, a: Agronomist | v.farmer.mandal.
    ↪ agronomist = a and d.agronomist = a and v.day = d.day implies v
    ↪ .dailyPlan = d }
```

```
//A Visit cannot exist without the associated DailyPlan
fact { all v: Visit, d: DailyPlan | v.dailyPlan = d implies v in d.
    ↪ visit }

//A Visit cannot exist without the associated Farmer
fact { all v: Visit, f: Farmer | v.farmer = f  implies v.dailyPlan.
    ↪ agronomist = f.mandal.agronomist }

//Visit and Farmer are related
fact { all f: Farmer, v: Visit | v.farmer = f  implies v in f.visit }

//A Visit refers only to one Farmer
fact { all v: Visit | ( no disj f1 , f2: Farmer | v in f1.visit and v
    ↪ in f2.visit ) }

//Farmer cannot have two Visit in the same day
fact { all f: Farmer | ( no disj v1 , v2: Visit | v1.day = v2.day and
    ↪ v1.dailyPlan.agronomist = v2.dailyPlan.agronomist and v1.farmer
    ↪  = f and v2.farmer = f ) }

//Two DailyPlan (or more) can have the same day or state
fact { some disj d1, d2: DailyPlan | d1.day = d2.day or d1.state = d2.
    ↪ state }

//Two DailyPlan having the same day cannot have the same Agronomist
fact { all disj d1, d2: DailyPlan | d1.day = d2.day implies d1.
    ↪ agronomist ≠ d2.agronomist }

//Two DailyPlan having different day can have the same Agronomist
fact { some disj d1, d2: DailyPlan | d1.day ≠ d2.day implies d1.
    ↪ agronomist = d2.agronomist }

//All Visit associated to the same DailyPlan must have different hour
fact { all disj v1, v2: Visit | v1.day = v2.day implies v1.hour ≠ v2.
    ↪ hour }

//Two Visit (or more) can be associated to the same Farmer or Status
fact { some disj v1, v2: Visit |  v1.farmer = v2.farmer or v1.status =
    ↪  v2.status }

// If a DailyPlan exists it must be CREATED, UPDATED or CONFIRMED
fact { all d: DailyPlan | d.state = GENERATED or d.state = UPDATED or
    ↪ d.state = CONFIRMED }

// If a Visit exists it must be SCHEDULED or DONE
fact { all v: Visit | v.status = SCHEDULED or v.status = DONE }

//If DailyPlan's State is CREATED for a certain day, associated Visit'
    ↪ s Status is SCHEDULED
fact { all v: Visit | v.dailyPlan.state = GENERATED implies v.status =
    ↪  SCHEDULED and v.dailyPlan.deviation = none }

//If DailyPlan's State is UPDATED for a certain day, associated Visit'
    ↪ s Status is SCHEDULED
fact { all v: Visit | v.dailyPlan.state = UPDATED implies v.status =
    ↪ SCHEDULED and v.dailyPlan.deviation = none}

//If DailyPlan's State is UPDATED, associated Visit's date is modified
    ↪  according to the update
fact { all v: Visit | v.dailyPlan.state = UPDATED and v.dailyPlan.
    ↪ state.selectedHour = v.hour implies v.dailyPlan.state.updateDay
```

```
    ↪  ≠ none and v.dailyPlan.state.updateHour ≠ none and v.hour = v.
    ↪ dailyPlan.state.updateHour and v.day = v.dailyPlan.state.
    ↪ updateDay and v.status = SCHEDULED }

//DailyPlan's State is UPDATED adding a new Visit
fact { all d: DailyPlan, v: Visit, f: Farmer | d.state = UPDATED and d
    ↪ .state.selectedHour not in d.visit.hour implies d.state.
    ↪ updateDay = none and d.state.updateHour = none and v in d.visit
    ↪  and v in f.visit and v.hour = v.dailyPlan.state.selectedHour
    ↪ and  v.status = SCHEDULED }

//If DailyPlan's State is CONFIRMED, associated Visit's Status is DONE
fact { all v: Visit | v.dailyPlan.state = CONFIRMED implies v.status =
    ↪  DONE and (v.dailyPlan.deviation ≠ none or v.dailyPlan.
    ↪ deviation = none)}

//HELP REQUEST

//For each Agronomist the set of HelpResponse must be different
fact { all disj a1, a2: Agronomist | a1.helpResponse & a2.helpResponse
    ↪  = none }

//For each WellPerformingFarmer the set of HelpResponse must be
    ↪ different
fact { all disj w1, w2: WellPerformingFarmer | w1.helpResponse & w2.
    ↪ helpResponse = none }

//For each responder the set of HelpResponse must be different
fact { all w: WellPerformingFarmer, a: Agronomist | w.helpResponse & a
    ↪ .helpResponse = none }

//For each Farmer set of HelpRequest must be different
fact { all disj f1, f2: Farmer | f1.helpRequest & f2.helpRequest =
    ↪ none }

//For each HelpRequest the set of helpResponse must be different
fact {  all disj h1, h2: HelpRequest | h1.helpResponse & h2.
    ↪ helpResponse = none }

//Each HelpRequest is associated to a unique id
fact uniqueHelpRequest { all disj h1, h2: HelpRequest | h1.id ≠ h2.id
    ↪ }

//A HelpResponse cannot exists without an associated HelpRequest
fact { all r: HelpResponse, h: HelpRequest| r.id = h.id implies r in h
    ↪ .helpResponse }

//Each HelpResponse must be related to the corresponding HelpRequest
fact { no r: HelpResponse | all h: HelpRequest | r.id ≠ h.id }

//Two HelpResponse (or more) can be associated to the same HelpRequest
fact { some disj r1, r2: HelpResponse | r1.id = r2.id }

//Two HelpRequest (or more) can have the same questionner or state or
    ↪ wellPerformingFarmerAsRecipient
fact { some disj h1, h2: HelpRequest |  h1.questionner = h2.
    ↪ questionner or  h1.state = h2.state or h1.
    ↪ wellPerformingFarmerAsRecipient = h2.
    ↪ wellPerformingFarmerAsRecipient }

//Two HelpRequest (or more) have always the same Recipient if they
    ↪ have the same questionner and if they have the same
```

```
    ↪ wellPerformingFarmerAsRecipient
fact { all disj h1, h2: HelpRequest | h1.questionner = h2.questionner
    ↪ and h1.wellPerformingFarmerAsRecipient =h2.
    ↪ wellPerformingFarmerAsRecipient implies h1.recipient =  h2.
    ↪ recipient }

//Each HelpResponse belongs to the set of HelpResponse of the related
    ↪ responder
fact { all  r: HelpResponse | r.responder.agronomist = none implies r
    ↪ in r.responder.wellPerformingFarmer.helpResponse }
fact { all  r: HelpResponse | r.responder.wellPerformingFarmer = none
    ↪ implies r in r.responder.agronomist.helpResponse }

//All HelpResponses are from Agronomist or WellPerformingFarmer
fact { all h: HelpRequest, r: HelpResponse | h.id = r.id implies r.
    ↪ responder.agronomist.helpResponse + r.responder.
    ↪ wellPerformingFarmer.helpResponse = h.helpResponse }

//HelpRequest has always responsible Agronomist as Recipient (the one
    ↪ belonging from the same Famer's Mandal)
fact { all h: HelpRequest | h.questionner.mandal.agronomist = h.
    ↪ recipient.agronomist }

//WellPerformingFarmer can be Recipient of HelpRequest only when
    ↪ wellPerformingFarmerAsRecipient is True
fact {
all h: HelpRequest, w: WellPerformingFarmer | h.
    ↪ wellPerformingFarmerAsRecipient = True implies h.recipient.
    ↪ wellPerformingFarmer = w }

//Each HelpResponse can have exactly one responder
fact {
all h: HelpRequest, r: HelpResponse | one w: WellPerformingFarmer | r.
    ↪ id = h.id and h.wellPerformingFarmerAsRecipient = True implies
    ↪ (( r.responder.wellPerformingFarmer = w and r.responder.
    ↪ agronomist = none ) or (r.responder.agronomist = h.recipient.
    ↪ agronomist and r.responder.wellPerformingFarmer = none )) }

//WellPerformingFarmer are not Recipient of HelpRequest when
    ↪ wellPerformingFarmerAsRecipient is False
fact {
all h: HelpRequest, r: HelpResponse | r in h.helpResponse and h.
    ↪ wellPerformingFarmerAsRecipient = False implies h.recipient.
    ↪ wellPerformingFarmer = none and r.responder.
    ↪ wellPerformingFarmer = none and r.responder.agronomist = h.
    ↪ recipient.agronomist }

//If WellPerformingFarmer is not Recipient of a Help Request, only the
    ↪  Agronomist will reply though a Help Response
fact {
all h: HelpRequest | h.recipient.wellPerformingFarmer = none implies h
    ↪ .helpResponse.responder.agronomist = h.recipient.agronomist
    ↪ and h.helpResponse.responder.wellPerformingFarmer = none}

// If a HelpRequest exists it must be CREATED or SOLVED
fact { all h: HelpRequest | h.state = CREATED or h.state = SOLVED }

//A CREATED HelpRequest must be visible by the questionner (Farmer) in
    ↪  HelpRequest
fact { all f: Farmer, h: HelpRequest | h.state = CREATED and h.
    ↪ questionner = f implies h in f.helpRequest }
```

```
//A SOLVED HelpRequest cannot be in HelpRequest (it cannot be visible
    ↪ by Farmer who solved it and it must be deleted with associated
    ↪ HelpResponse)
fact { all f: Farmer, h: HelpRequest | h.state = SOLVED and h.
    ↪ questionner = f implies h not in f.helpRequest and h = none }
```

4.1.3 Assertions

This section shows the piece of code in Alloy that contains assertions to be checked in order to validate the model.

```
//DISCUSSION FORUM

//G.9 Allow farmers to interact on the Discussion Forum

//Farmer opens a Thread
assert openAThread {
all t: Thread | one f: Farmer | t in f.thread }
check openAThread for 4

//Farmer replies to a Thread writing a Post
assert writeAPost {
no f: Farmer | one p: Post | p.responder = f and p not in f.post }
check writeAPost for 5

//Every Post is in exactly one Thread
assert oneLocation {
all p: Post | one t: Thread | p in t.post }
check oneLocation for 5


//DAILYPLAN

//G.13 Allows agronomists to visualize their own Daily Plan to visit
    ↪ farms in the mandal

//Agronomist visualizes DailyPlan
assert visualizeDailyPlan {
all d: DailyPlan | ( one a: Agronomist | d in a.dailyPlan) }
check visualizeDailyPlan for 4

//Each DailyPlan is associated to exactly one Agronomist
assert oneAgronomist {
all d: DailyPlan | one a: Agronomist | d in a.dailyPlan }
check oneAgronomist for 5

//Each Visit is in exactly one DailyPlan
assert oneDailyPlan {
all v: Visit | one d: DailyPlan | v in d.visit }
check oneDailyPlan for 5

//Each Visit is associated to exactly one Farmer
assert oneFarmer {
all v: Visit | one f: Farmer | v in f.visit }
check oneFarmer for 5

//G.14 Allows agronomists to manage their own Daily Plan

//Agronomist confirms DailyPlan specifying deviations if needed
```

```
assert confirmDailyPlanDev {
all d: DailyPlan | d.state = CONFIRMED implies ( one a: Agronomist | d
    ↪  in a.dailyPlan and (d.deviation ≠ none or d.deviation = none))
    ↪  }
check confirmDailyPlanDev for 4

//Agronomist updates the DailyPlan
assert updateDailyPlan {
all d: DailyPlan | d.state = UPDATED implies ( one a: Agronomist | d
    ↪ in a.dailyPlan) }
check updateDailyPlan for 4

//Agronomist adds a new Visit in the DailyPlan
assert newVisit {
no d: DailyPlan | one v: Visit | d.state = UPDATED and d.state.
    ↪ selectedHour not in d.visit.hour and v.hour = d.state.
    ↪ selectedHour and v not in d.visit }
check newVisit for 4

//Agronomist updates a Visit in the DailyPlan
assert updateVisit {
no d: DailyPlan | one v: Visit | d.state = UPDATED and d.state.
    ↪ selectedHour in d.visit.hour and v.hour = d.state.updateHour
    ↪ and v.day = d.state.updateDay and v not in d.visit }
check updateVisit for 4


//HELPREQUEST

//G.8 Allows farmers to receive help from agronomist and other farmers
    ↪  through Help Requests

//Farmers makes a HelpRequest
assert makeHelpRequest {
all h: HelpRequest | h. state = CREATED implies ( one f: Farmer | h in
    ↪  f.helpRequest) }
check makeHelpRequest for 4

//Farmer makes HelpRequest with Agronomist as Recipient by default
assert agronomistByDefault {
no a : Agronomist | one h : HelpRequest | a = h.questionner.mandal.
    ↪ agronomist and a not in h.recipient.agronomist }
check agronomistByDefault for 4

//Farmer makes HelpRequest with WellPerformingFarmer as Recipient too
assert addRecipient {
no w : WellPerformingFarmer | one h : HelpRequest | h.
    ↪ wellPerformingFarmerAsRecipient = True and w not in h.recipient
    ↪ .wellPerformingFarmer }
check addRecipient for 4

//Agronomist replies to a HelpRequest
assert AgronomistHelpResponse {
no a: Agronomist | one r: HelpResponse | r.responder.agronomist = a
    ↪ and r not in a.helpResponse }
check AgronomistHelpResponse for 5

//WellPerformingFarmer replies to a HelpRequest
assert FarmerHelpResponse {
no w: WellPerformingFarmer | one r: HelpResponse | r.responder.
    ↪ wellPerformingFarmer = w and r not in w.helpResponse }
check FarmerHelpResponse for 5
```

```
//Each HelpResponse is in exactly one HelpRequest
assert oneLocationHelpResponse {
all r: HelpResponse | one h: HelpRequest | r in h.helpResponse }
check oneLocationHelpResponse for 5

//Farmer solves a HelpRequest
assert solveHelpRequest {
all h: HelpRequest, r: HelpResponse | h. state = SOLVED and h.id = r.
    ↪ id implies ( no f : Farmer, a: Agronomist, w:
    ↪ WellPerformingFarmer | h in f.helpRequest or r in a.
    ↪ helpResponse or r in w.helpResponse) }
check solveHelpRequest for 4
```

4.1.4 Analysis Results

All assertions checked in Alloy code may be valid. No counterexample was found as shown below.

```
18 commands were executed. The results are:
  #1: No counterexample found. openAThread may be valid.
  #2: No counterexample found. writeAPost may be valid.
  #3: No counterexample found. oneLocation may be valid.
  #4: No counterexample found. visualizeDailyPlan may be valid.
  #5: No counterexample found. oneAgronomist may be valid.
  #6: No counterexample found. oneDailyPlan may be valid.
  #7: No counterexample found. oneFarmer may be valid.
  #8: No counterexample found. confirmDailyPlanDev may be valid.
  #9: No counterexample found. updateDailyPlan may be valid.
  #10: No counterexample found. newVisit may be valid.
  #11: No counterexample found. updateVisit may be valid.
  #12: No counterexample found. makeHelpRequest may be valid.
  #13: No counterexample found. agronomistByDefault may be valid.
  #14: No counterexample found. addRecipient may be valid.
  #15: No counterexample found. AgronomistHelpResponse may be valid.
  #16: No counterexample found. FarmerHelpResponse may be valid.
  #17: No counterexample found. oneLocationHelpResponse may be valid.
  #18: No counterexample found. solveHelpRequest may be valid.
```

Figure 4.1: Alloy Analysis Results

4.1.5 Generated Worlds

In this section are shown the three different Generated Worlds, one for each component analyzed through Alloy.

```
//DISCUSSION FORUM

//Simulation that shows a set of Thread and related Post
pred world0 {
```

```
#Thread > 3
#Post > 3
}
run world0 for 7

//DAILY PLAN

//Simulation that shows set of Visit belonging to different DailyPlan
    ↪ with different State
pred world1 {
#DailyPlan > 0
#Visit > 0
one d: DailyPlan | d.state = GENERATED
one d: DailyPlan | d.state = UPDATED
one d: DailyPlan | d.state = CONFIRMED and d.deviation ≠ none
one d: DailyPlan | d.state = CONFIRMED and d.deviation = none
}
run world1 for 5

//HELP REQUEST

//Simulation that shows a set of Help Request with both Agronomist and
    ↪  WellPerformingFarmers as Recipient
pred world2 {
#HelpRequest > 0
#HelpResponse > 0
#Farmer > 0
#WellPerformingFarmer > 0
#Agronomist > 0
one h: HelpRequest | h.wellPerformingFarmerAsRecipient = False
one h: HelpRequest | h.wellPerformingFarmerAsRecipient = True
}
run world2 for 5
```

```
Executing "Run world0 for 7"
  Solver=sat4j Bitwidth=4 MaxSeq=7 SkolemDepth=1 Symmetry=20 Mode=batch
  61595 vars. 2324 primary vars. 137542 clauses. 248ms.
  Instance found. Predicate is consistent. 505ms.

Executing "Run world1 for 5"
  Solver=sat4j Bitwidth=4 MaxSeq=5 SkolemDepth=1 Symmetry=20 Mode=batch
  92834 vars. 3664 primary vars. 207018 clauses. 161ms.
  Instance found. Predicate is consistent. 440ms.

Executing "Run world2 for 5"
  Solver=sat4j Bitwidth=4 MaxSeq=5 SkolemDepth=1 Symmetry=20 Mode=batch
  124096 vars. 5004 primary vars. 276564 clauses. 163ms.
  Instance found. Predicate is consistent. 239ms.
```

Figure 4.2: Alloy Analysis Results

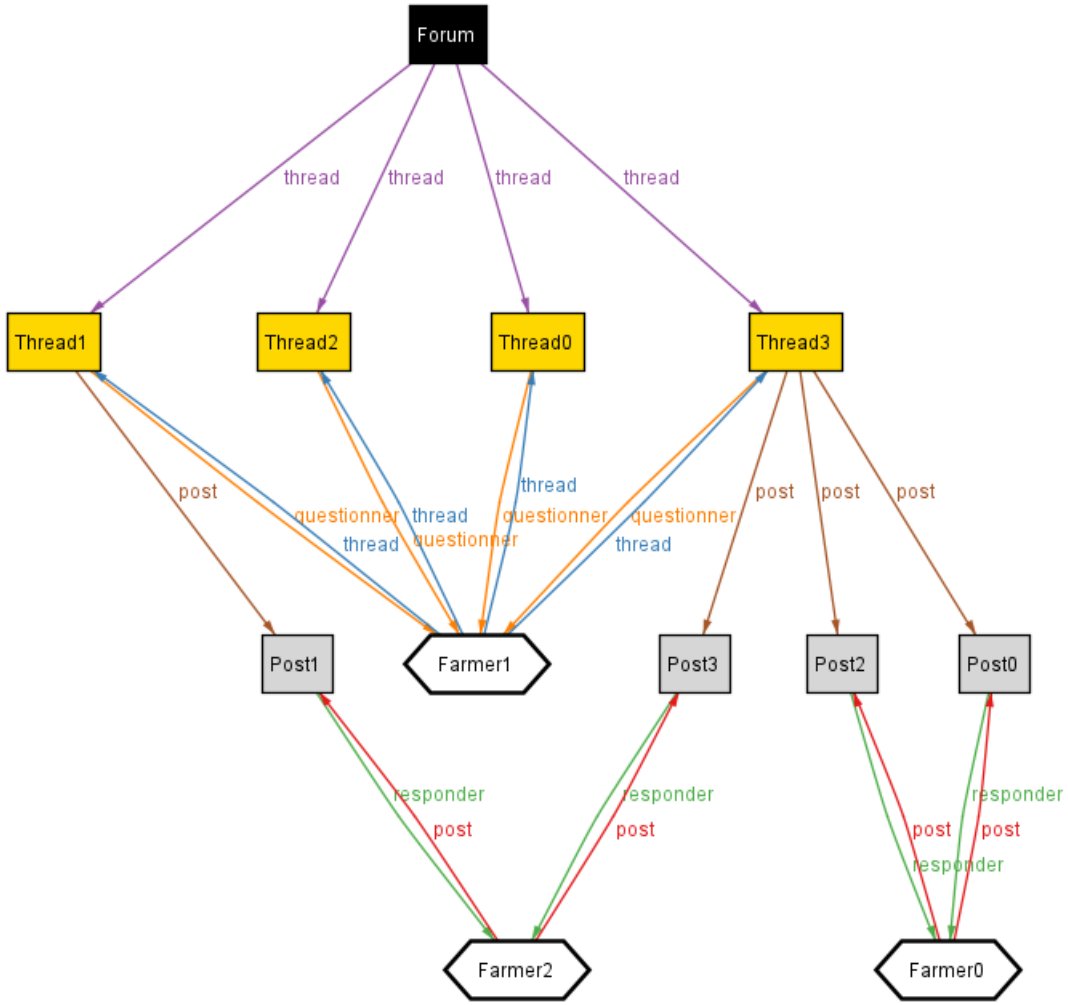

Figure 4.3: Generated World0 - Discussion Forum

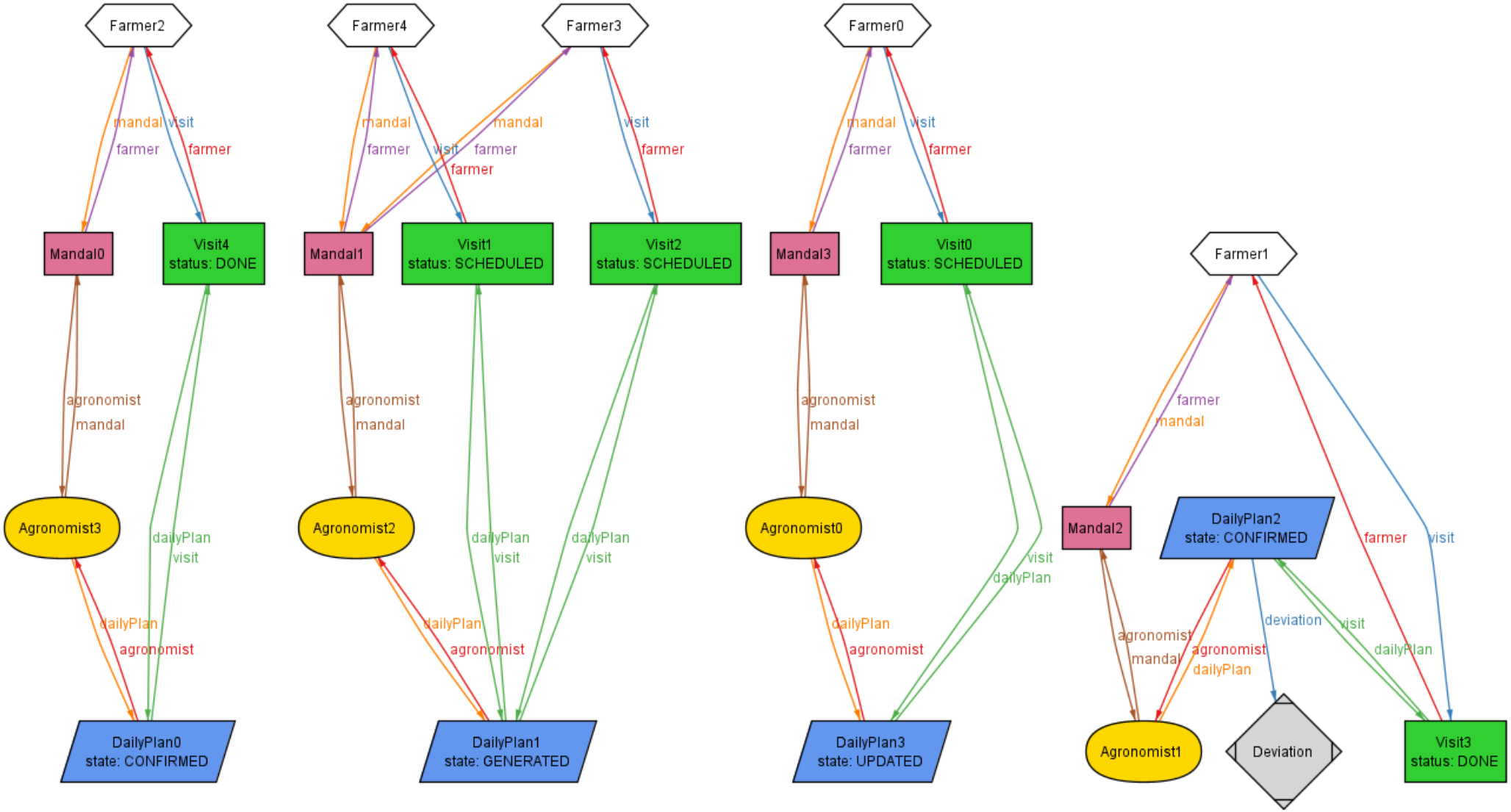

Figure 4.4: Generated World1 - Daily Plan

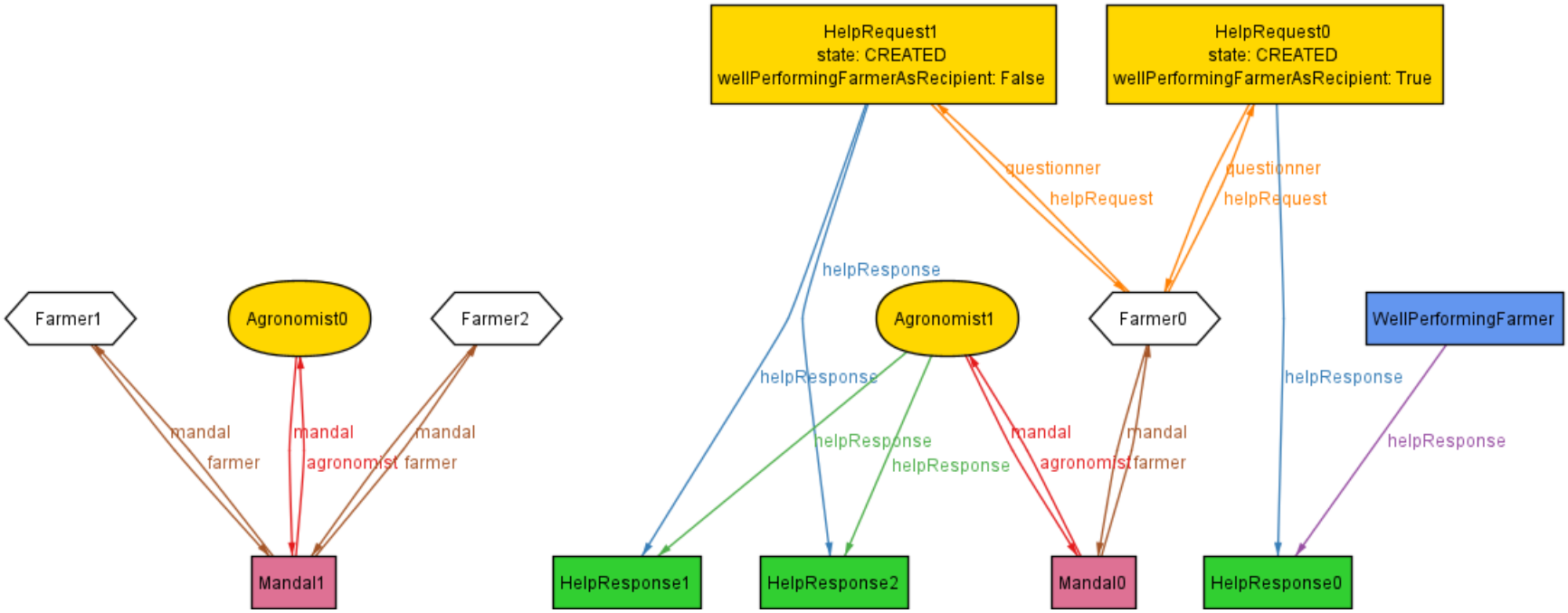

Figure 4.5: Generated World2 - Help Request

Chapter 5

Effort Spent

5.0.1 Ali Arslan

Table 5.1: Effort spent - Ali Arslan

Task	Hours
Case study comprehension for Q&A session with stakeholders	0.45
Attendance of Q&A session with stakeholders	1.00
Document restyling	0.40
Goals section	0.30
World and Shared phenomena	1.00
Class Diagram	1.30
State Diagrams	0.30
Scenarios	1.00
Domain assumptions	1.00
Alloy revision	0.30
Mock-up	8.00
Document revision	2.00

5.0.2 Servidio Elisa

Table 5.2: Effort spent - Servidio Elisa

Task	Hours
Case study comprehension for Q&A session with stakeholders	0.45
Attendance of Q&A session with stakeholders	1.00
Latex document template	0.15
Purpose section	0.35
Goals section	0.30
World and Shared phenomena	1.30
Class Diagram	1.30
State Diagrams	0.50
Definitions, Acronyms, Abbreviations	0.20
Scenarios	2.00
Assumptions, dependencies and constraints	0.25
Product Functions	2.00
Use Cases	2.00
Sequence Diagrams	2.00
Requirements	1.30
Traceability Matrix	0.30
Alloy Model	7.00
Document revision	2.00

5.0.3 Suriano Federica

Table 5.3: Effort spent - Suriano Federica

Task	Hours
Case study comprehension for Q&A session with stakeholders	0.45
Attendance of Q&A session with stakeholders	1.00
Goals section	0.30
Scope section and World and Shared phenomena	2.30
Class Diagram	1.30
State Diagrams	0.30
Document Structure section	0.20
User Characteristics	0.15
Scenarios	2.00
Product functions	2.00
Use cases	2.00
Sequence diagrams	2.00
Requirements	1.30
Performance requirements	0.20
Design constraints	0.20
Software system attributes	0.20
Alloy revision	0.30
Document revision	2.00