POLITECNICO
DI MILANO

Version 1.0.0

Requirements Analysis and Specification Document

Authors:
Marco Zanghieri - 10588478
Gabriele De Santis - 10820992
Michele Terziani - 10617090

23.12.2021

Contents

1 Introduction

A Purpose

The purpose of the Requirements Analysis and Specification Document (RASD) is to provide a description of the platform by defining the main goals, the domain and its representation through models, but also the list of the most important requirements and specifications that characterize the development of the software described below.

This document has the purpose to guide the developer in the realization of the DREAM platform - Data-dRiven PrEdictive FArMing, whose purpose is to support different stakeholders in the food system by deploying new models to acquire and combine data.

The DREAM system is an application whose aim is to support the agricultural system of the Indian state of Telangana, by collecting data and analyzing them in order to support local workers.
The platform has to provide specific data to the farmers in Telangana in order to receive suggestions about specific crops or fertilizers to be used. Those suggestions are given by an agronomist which has to receive, gather and manage all the data about the land they are responsible of. Thanks to that, they can plan which farms they are going to visit, in order to get into direct contact with the farmers. Finally, the policy makers analyze the data about each land, keeping in consideration the general events, as the meteorological conditions, and the behaviour of the agronomist on a specific land. Their purpose is to identify which are the farmers performing especially well (and the ones who needs help) in order to share useful practice.

B Scope

The software is aimed to create a platform which will be able to reach communication and coordination between different workers in the agriculture sector.
These goals are reached through the collection and review of data retrieved by actors, external services and IT providers.
In first place, IT providers look at the meteorological and geological data to support the validity of the information. The most important is the information about the weather by the Telangana State Development Planning Society and the soil moisture data from Climate Change Initiative, part of the ESA program. They aim to acquire and forward:

- Data concerning meteorological short-term and long-term forecasts.
- Information obtained by the water irrigation system concerning the amount of water used by each farmer.
- Information obtained by sensors deployed on the territory and measuring the humidity of soil.

Given these data, DREAM involves in the project three main actors:

- **Farmer**: Farmer can visualize useful data on weather forecast and specific suggestion about crops to plant and fertilizer, sorted by their location and type of production. This actor can also insert data and issues related to their production and possibly ask for help and suggestions to the agronomist. In addiction to that, farmers can create and participate to discussion forums with other farmers in order to share and receive solutions for any problem they encounter.

- **Agronomist**: An agronomist can define the area they are responsible for, create or update daily plans to visit farms of their area. Therefore, the software acquires and combines this specifics in order to visualize data concerning weather forecasts in that area and a ranking of the best performing farmers working there. The purpose of this actor is to collect information provided by the farmers about their production (types of products, produced amount per product) and their performance so that it can be sent to the government. In addition, agronomist can receive request for help from the farmers and answer to these requests.

- **Telangana's policy maker**: A Telegana's policy maker can manage and analyze all the data gathered by the Agronomist. Their scope is to have a complete overview of the farmers working in each land, in order to keep track of their performance taking into consideration the meteorological adverse events too. By doing that, they want to incentive the most productive farmers through special privilege and encourage them to share their useful practice with the less productive farmers. Finally, the policy makers also monitor the initiatives carried out by the agronomist to check whether they are optimal and efficient to reach the desired result on a specific land.

B.1 World Phenomena

	World Phenomena
WP1	Farmers encounter some problems with their crops
WP2	Farmers decide to plant some particular plant
WP3	Not all Farmers have a personal communication device
WP4	Agronomist decide to visit some farmers
WP5	A critical weather condition is imminent
WP6	Policy Makers identify farmers with bad performance

B.2 Shared Phenomena

	Shared Phenomena	Control
SP1	Farmer send a ticket for help or suggestion	World Controlled
SP2	Farmer receive a suggestion from the Agronomist	Machine Controlled
SP3	Farmer send a message in the Farmers' forum	World Controlled
SP4	Farmer receive a message in the Farmers' forum	Machine Controlled
SP5	Farmer insert data about his crops (water, plants, fertilizer, etc)	World Controlled
SP6	Farmer request the visit of an agronomist	World Controlled
SP7	Farmer visualize info about weather	Machine Controlled
SP8	Farmer receive an alert from the Agronomist	Machine Controlled
SP9	Agronomist create, modify or delete daily plans	Machine Controlled
SP10	Agronomist create a report of his daily plan	World Controlled
SP11	Agronomist chooses the area he is responsible of	Machine Controlled
SP12	Agronomist visualize data about the farmer and the land	Machine Controlled
SP13	Agronomist visualize data about the weather	Machine Controlled
SP14	Agronomist receive a report from the Policy Maker	Machine Controlled
SP15	Agronomist receive a ticket from a Farmer	Machine Controlled
SP16	Agronomist answer to the Farmer's ticket	World Controlled
SP17	Agronomist send an alert to a land	World Controlled
SP18	Agronomists create, delete or update lists of Farmers	World Controlled
SP19	Policy Maker send a report to an agronomist	World Controlled
SP20	Policy Maker visualize data about agronomist, farmer and land	Machine Controlled
SP21	Policy Maker visualize data about weather	Machine Controlled
SP22	Policy Makers create, delete or update lists of Farmers	Machine Controlled

B.3 Goals

The goals for this project focus on 5 points:

	Goals
G1	Ensure correct receiving of data to policy makers for governance models
G2	Provide an overview analysis about a specific land
G3	Allow farmers and agronomist to exchange suggestions
G4	Provide farmers' performance evaluation
G5	Help farmers to improve their production

C Definitions, Acronyms, Abbreviations

C.1 Acronyms

	Acronyms
DREAM	Data-dRiven prEdictive fArMing
RASD	Requirement Analysis and Specification Document
ESA	European Space Agency

C.2 Abbreviations

	Abbreviations
WPn	World Phenomenon number n
SPn	Shared Phenomenon number n
Gn	Goal number n
Rn	Requirement number n
Dn	Domain number n
PM	Policy Maker

D Revision history

Date	**Modification**
01/11	First Version

E Reference Documents

- Specification Document: "DREAM Project Assignment.pdf"
- Slides of the lectures

F Document Structure

- **Chapter 1** gives an introduction about the purpose of the document and the development of the application, with its corresponding specifications such as the definitions, acronyms, abbreviation, revision history of the document and the references. Besides, are specified the main goals, world and shared phenomena of the software.
- **Chapter 2** contains the overall description of the project. In the product perspective are included the statecharts of the major function of the application and the model description through a Class diagram. In user characteristic are explained the types of actors that can use the application. Moreover, the product function clarified the functionalities of the application. Finally, are included the domain assumption that can be deducted from the assignment.
- **Chapter 3** presents the interface requirement including: user, hardware, software and communication interfaces. This section contains the core of the document, the specification of functional and non-functional requirements. Functional requirements are submitted with a list of use cases with their corresponding sequence diagrams and some scenarios useful to identify specific cases in which the application can be utilised. Non-functional requirements included: performance, design and the software systems attributes.
- **Chapter 4** includes the alloy code and the corresponding metamodels generated from it, with a brief introduction about the main purpose of the alloy model.
- **Chapter 5** shows the effort spent for each member of the group.
- **Chapter 6** includes the reference documents.

2 Overall Description

A Product perspective

Here we include scenarios and further details on the shared phenomena and a domain model (class diagrams and statecharts)

A.1 Scenarios

1. **Farmer asking for help**
   Dhruv is a farmer in Telangana. He noticed during the last week that some plants in his field began to spoil. The plants in question have a brownish color

and after some day they just die. He thinks it is due to a plant disease, but he doesn't know which kind of disease neither which kind of treatment to use. Dhruv takes the tablet and inserts his credential on the DREAM platform. After that, he compiles a ticket form for the agronomist to ask for help. He writes what he noticed in the last days and he attaches some pictures. After 24 hours, he receives a reply from the agronomist which suggest to use a specific treatment and to keep him up to date about the progress.
Unluckily, the plants doesn't recover even with the treatment suggested by the agronomist. Dhruv send another request for help to him which promptly plan a visit to Dhruv's farm by using DREAM. He inserts the day and the time and the notification visit arrives to Dhruv.

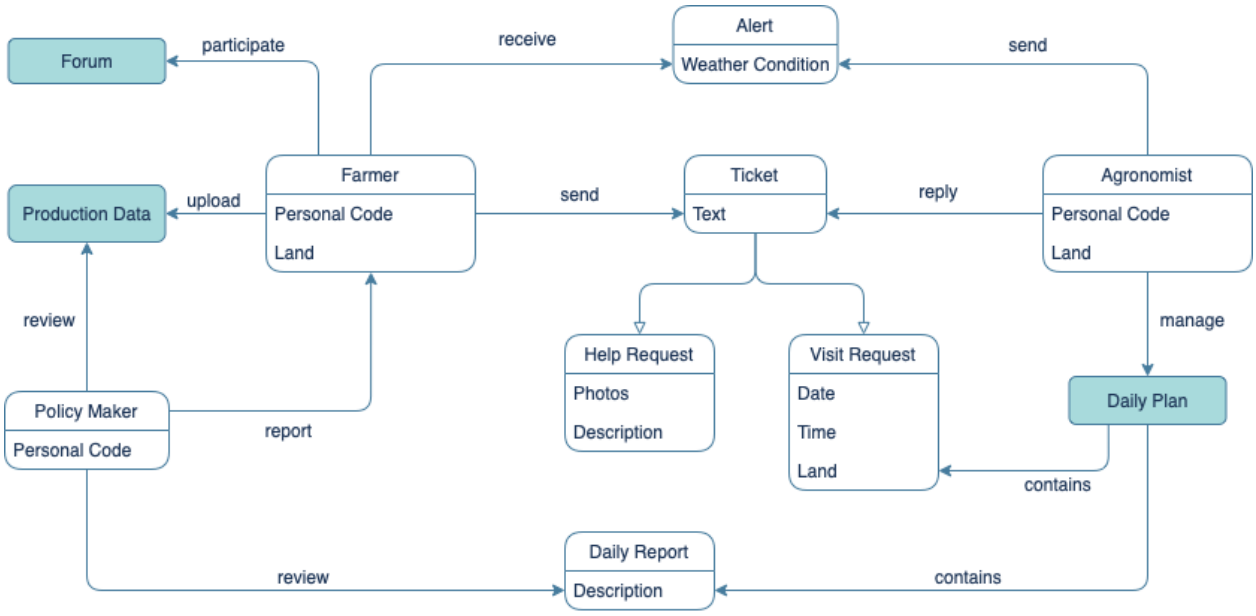
2. **Farmer visualizing relevant data and Forum discussions**
It is almost time to plant the seeds and Chetan, an Indian farmer, knows very well you need the proper weather condition to ensure a flourishing growth of the plants. Indeed, as all the farmers knows, planting is suggested according to specific weather condition. Chetan, thanks to the DREAM platform, can visualize real time weather forecast. Thanks to that, he decides the best time to plant the seeds is in three days.
Finally, since it is the right time to plant, he can read in the Forum page all the suggestion exchanged with the other farmers in that land in order to plant. Indeed, he read a farmer suggested to use a specific fertilizer for that crop which is resilient to a fungal disease which is spreading in other lands.

3. **The daily plan of an agronomist**
Kapi is an agronomist and his working day is just beginning. First of all he inspects through the DREAM platform the farmers, their crops and how each one is performing. He also reads the help request he received. After that, he decide to make the daily plan for the next day: he decides to visit first the farmers performing bad that requested for help and the ones requesting for help not considering their performance. He schedules the time and, once he finished, he confirms it.
The next day he start the visits following the daily plan he made. Unlikely, because of an unexpected event, Kapi can't continue the visit he planned. Because of that, he access the DREAM application and he cancels the other visits in that day. A notification on the DREAM application arrives to the farmers which were waiting for him.

4. **Agronomist alerting for an incoming flood**
In the next days heavy rain are expected. Naresh, an agronomist of a land in Telangana, inspect the forecast. Thanks to the good knowledge he has of the land he recognize there is a flood risk in a specific area near a river. Because of that, he instantly access to the DREAM platform and he alerts all the farmers in that area, suggesting them how to protect the plants before the flood come. Since there is a big risk of loosing a good part of the crops, Naresh decide to plan a visit after the rainy days by using the application.

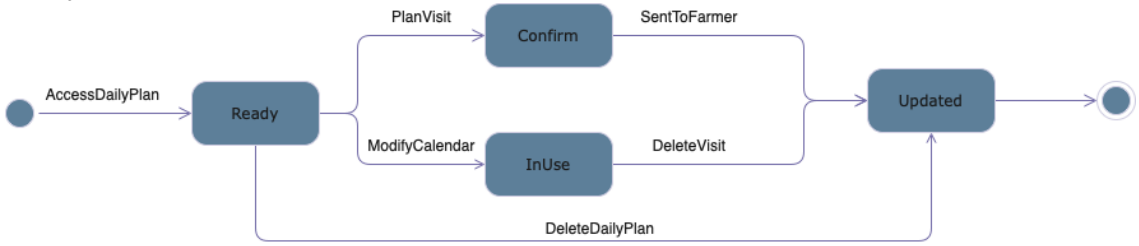
5. **Policy Makers reporting bad performing farmers**
   Darpak is one of the policy makers in Telangana. He has a complete view of all the farmers in all the lands and the corresponding agronomist. He notice through the data of the crops of a single farmer, that he didn't performed very well in the last months. Because of that, Darpak decide to inspect whether the farmer had some unexpected problem which caused a bad performance. Unlikely, all the data doesn't show any problem: the weather was good and the fertilizer used was the proper one. Finally, he found the problem: the farmer didn't gave enough water to the plant since the water irrigation system show that a little quantity of water was used considering the time period into consideration. Since the bad performance was caused by an improper behaviour of the farmer and not to external causes, the policy maker decide to report him to the agronomist so that he can be helped.
   Finally, the policy maker insert the farmer into a list of farmer to inspect more often.

A.2 Class Diagram

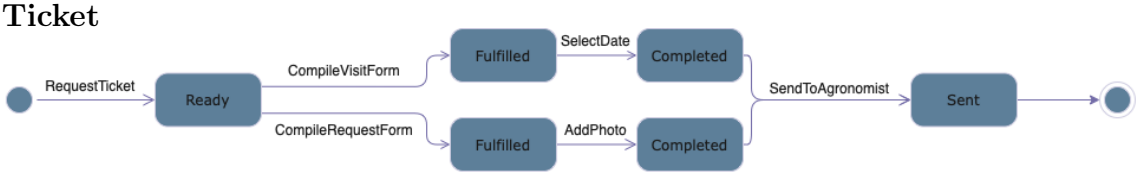


A.3 State Charts

- **Daily Plan**



- **Ticket**



B Product functions

Here are described the functions of the software. We are going to divide this section for each one of the three actors.

B.1 Policy Maker

- **Visualize data about farmers and weather**
  The main functionality in the Policy Maker dashboard is to visualize and analyze all the relevant data about the farmers, the lands (such as quantity, status and type of crops, quantity of water used, type of fertilizer used etc) and also about the meteorological weather. Thanks to that, the PM can cross-reference the data to identify the best and the worst performing farmers.
  In this page, the PM can also visualize the interaction between the farmers and their agronomist, so that it's possible to identify also possible problems due to agronomist mistake.

- **Send report to Agronomist(s)**
  Once the PM have identified the best and worst performing farmers in the dashboard, he can report them to the agronomist. For each one of the farmer, the PM will click on the report button and a form page will appear to him. In this page, he'll write which incentives will be given to the farmer or how to help him if he's not performing well. Once the report page will be submitted, the Agronomist will receive these information. Of course, since the PM analyze all the farmers regardless the area in which they are, not all the agronomist will receive these information (or, at least, not always).
  The PM can also use the report to communicate to the Agronomist what they think about the initiatives they have with good performing farmers. Indeed, the PM can also suggest initiatives to carry out to help the farmers in need of help.

B.2 Agronomist

- **Choose the area they are responsible of**
  Once the Agronomist has been chosen by the government, he can access the website thought a personal code and a key. The first time an Agronomist will access the platform, he will have to choose which area he is responsible of. Once the choice is made, it cannot be changed.

- **Visualize data about farmers and weather**
  The Agronomist can visualize a similar dashboard to the one of the Policy

Maker. Thanks to that, they can identify the farmers who needs help and, if necessary, contact or visit them.

- **Messaging page**
  The Agronomist have a messaging page in which he can receive the reports from the PM and receive or answer to the tickets of the farmers. The messaging page is organized as a mail: each message can have a subject and a text body. It can be also possible to send videos or photos.
  It's necessary to point out that the messaging system with the PM is one-way (only the PM can send messages to the Agronomist) while with the Farmers is two-way (the Agronomist and the Farmers can bot receive or send messages). The communication with the Farmers is essential. Indeed, if the agronomist notice a Farmer is doing something bad for the harvest (for example, he's giving too much water), he can send a message to the Farmer to inform him about that and suggest him the correct way to do that.

- **Manage lists of farmers**
  When the Agronomist visualize the data about the Farmers, he can decide to insert a specific Farmer into a list. There can be multiple list. The default ones are "best performing" and "worst performing". The lists can be useful for the Agronomist to remember which are the farmers to observe.

- **Manage daily plan in the calendar**
  The Agronomist have to periodically create a daily plan. All the daily plans can be visualized in a calendar page. It's possible to choose to visualize the calendar page daily, weekly, monthly or annually.
  The Agronomist, when creating a daily plan, have to specify a day, a time and a farm or a farmer. When choosing the farm to visit, it's possible to visualize also the list of the worst performing farmers, so that the Agronomist can choose a farm of one of them. Indeed, the Agronomist should visit more often the under-performing farms. When choosing the farm, near to the farm name there is also a number which shows the number of times that specific farm has been visited in the current year. The default value is 0 and each one of the farm has to be visited at least twice per year.
  Once the event has been confirmed, all the farmers in the farm will be notified. The Agronomist can change the daily plan up to 24 hours before the visit.
  If there were some deviations or if the Agronomist had made some changes of the daily plan in the last 24 hours, he has to manually compile a form where he specifies what happened. Once all the visits of a specific day are completed and (if necessary) the form is compiled, the Agronomist has to press a "confirm" button in the daily page.

B.3 Farmer

- **Insert data about crops**
  Each Farmer can insert data about the crops they have in the land.
  In the insert data page the Farmer has to choose the type of crop he's writing about (among all the crops created in the past), or eventually create a new

type of crop.
After that, the Farmer writes in that form all the information he want to send.
Finally, the Farmer can send the form to the Agronomist.

- **Create a message (ticker or visit request)**
  Each Farmer can communicate with the Agronomist through messages which can be of two types: ticket and visit request.
  Tickets are used to ask for help or suggestion to the Agronomist while the visit request are used to ask the Agronomist to visit the land.
  First of all, the Farmer have to login with his credentials. After that, the Farmer choose if he wants to send a ticket or a visit request and compiles a form in which he writes about the kind of suggestion he needs or the reason for the visit. Once he compiles the form, the Farmer can send the ticket which will be visualized in the messaging page of the Agronomist.

- **Interact in the farmer forum**
  The Farmers can interact with Farmers of other lands through a forum.
  The forum is composed of a series of topics. Each Farmer can create a new topic or reply to past topics.
  First of all the Farmer have to login with his credentials. After that, he can visualize all the title of the past topics and eventually click one of them to read all the messages shared among the Farmers. If he wants to, he can reply to the topic with a message through a form.

- **Visualize data about the weather**
  The Farmer can visualize the weather condition through a weather page. The weather condition are the same visualized by the Agronomist and the Policy Makers since they take the weather data from the same API.

C User characteristics

The system involves 3 main characters:

- **Farmer**: Anyone involved in the field of agriculture. It interacts with the system by updating the data and asking for help.

- **Agronomist**: Anyone responsible for a single land in the region. With the system, he manage the farmers in his land and eventually helps them.

- **Policy Maker**: Anyone that has the overall view of the Telengana region and use the system to evaluate and intervene in case of bad performance of both farmers and agronomists.

D Assumptions, dependencies and constraints

D.1 Domains

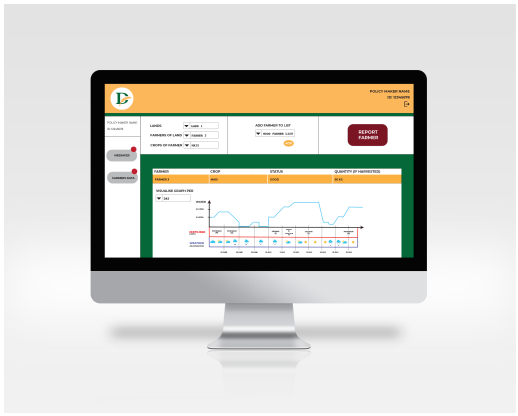
	Domain Assumptions
D1	Information about weather and lands are correctly received by IT providers
D2	Each land has a serialized tablet
D3	Each tablet has a camera
D4	Each tablet is connected into a network
D5	The tablet provides tickets and forms to submit
D6	The tablet receives updates and alerts
D7	An agronomist manages only one land with all the farmers in it
D8	An agronomist visits each land at least twice a year
D9	An agronomist notifies a daily plan before 24 hours
D10	An agronomist writes a daily report of his visits
D11	For every weather emergency, the agronomist alerts the farmers about the condition
D12	Each agronomist correctly view the overall status of his land
D13	All farmers uses the tablet
D14	Each actor has a personal code
D15	A policy maker correctly receives overviews of lands and farmers
D16	A policy maker correctly reviews the initiative of the agronomists
D17	Each daily plan can't have more than 5 visits

3 Specific Requirements

A External Interface Requirements

A.1 User Interfaces

Figure 1: Log in



(a) Computer Dashboard

(b) Tablet Dashboard

A.2 Hardware Interfaces

- **Farmers**
  Because of the extension of the region and the high expenses required to monitor every farmer, every land is associated with a serialized industrial tablet.

Every Farmer can use the tablet of its land. This device has been chosen because of its durability and its low cost.

- **Agronomists and Policy Makers**
  Agronomist and Policy Makers can access the software through a web application. It can be accessed through a computer or through its own smartphone. The web application can be accessed also by the farmers if they posses a personal device.

A.3 Software Interfaces

Here there are the external software interfaces used.

- **Weather**
  The platform uses an external weather API to visualize the live data. One of the possible APIs which can be used is the one of the Telengana Government which provide useful maps and daily reports.

- **Calendar**
  To manage the Daily Plan of the Agronomist it can be used an external API which provides a calendar. For example, the Google Calendar API lets you display, create and modify calendar events. It can be a useful solution also to display the Agronomist planned visit in the industrial tablet of the land.

A.4 Communication Interfaces

All the actors in the systems have to have an Internet connection. While Policy Makers and Agronomist could have (but not necessarily) a cabled connection, this is very difficult for the Farmers. Each industrial tablet in the land have to be connected to Internet through a wireless connection, for example by using 3G or 4G connection. Among all the providers, the telecommunication company Airtel has been identified because of its national/international long distance services and its trust. Its' services cover entirely the state of Telengana.

B Functional Requirements

	Requirements
R1	The system allows farmers to publish data about their land
R2	The system provides up-to-date weather forecast
R3	Farmers can consult in a dedicated page all the suggestions made by the agronomist
R4	The system allows farmers to send a help request

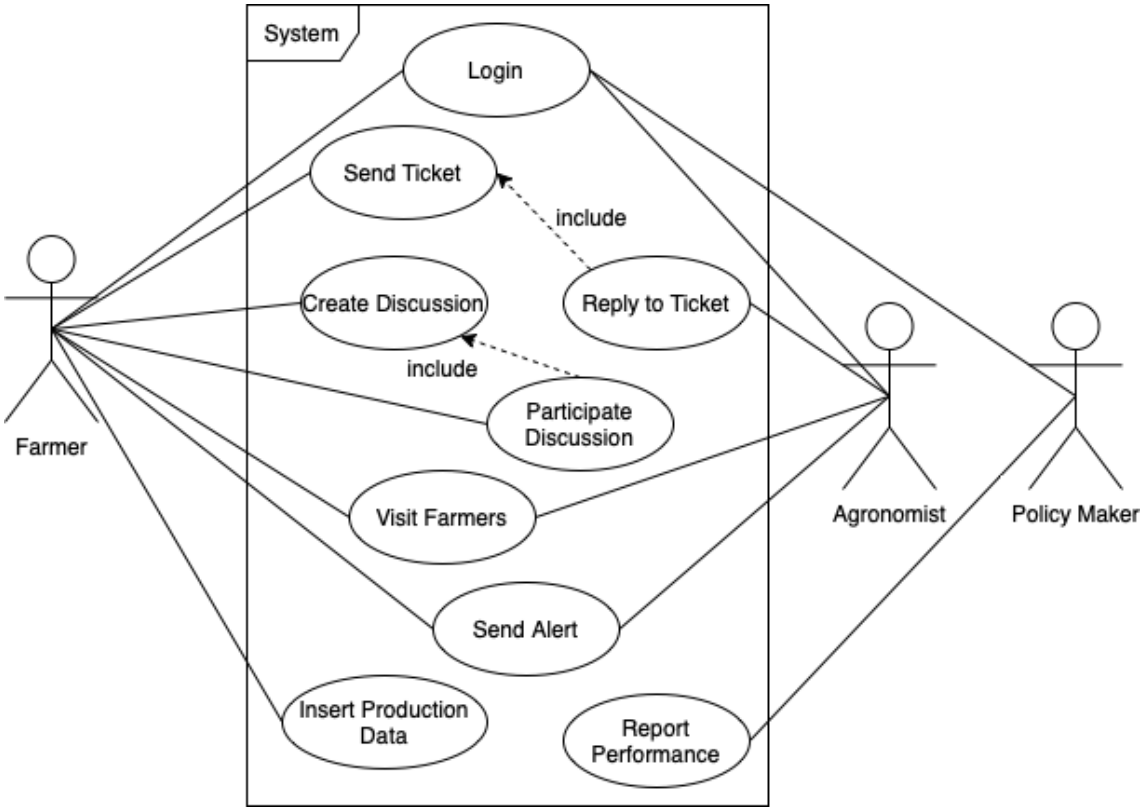
R5	After specifying the assigned land, the system allows agronomist to fetch all farmers' data of that land
R6	The system sends to farmers agronomist's visit details
R7	Agronomists can create a new daily plan
R8	The system won't allow agronomists to plan more than one visit per daily plan
R9	The systems saves the daily plan of agronomists
R10	After creating a new daily plan, the system allows agronomists to modify that plan
R11	When creating a visit in the daily plan, the systems notifies the farmer of the visit
R12	The system sends alerts made by agronomist to the farmers
R13	When a help request is made, the system shall allow agronomists to answer with their suggestions
R14	All the request with their relative answers are stored by the system
R15	The system allows agronomists to delete their daily plan
R16	After deleting a daily plan, the system sends a notification to the farmers involved
R17	Farmers can fetch all the request with their relative answers
R18	The system permits to users to register with their personal code
R19	The system receives updates for the farmers by the agronomist
R20	The system manages notifications
R21	The system offers different services according to the user type
R22	The system provides data about all the land to the policy makers
R23	The system allow policy makers to review the initiative performed by agronomists
R24	The system allow policy makers to report bad performing farmers to agronomist
R25	The systems provides performance data about all the farmers to policy makers
R26	The system allows a daily plan to be modified or deleted at least 24 hours before the visits
R27	The system allows a daily plan to be modified or deleted 24 hours before the visits only by submitting a form

R28	The system stores all the agronomists' daily reports

B.1 Mapping

G1	**Ensure correct receiving of data to policy makers for governance models**
	- R1, R5, R13, R16, R19, R20, R21, R22, R23 - D8, D10, D13 , D14, D15, D16
G2	**Provide an overview analysis about a specific land**
	- R1, R2, R5, R8, R16, R18, R19, R20, R21, R23 - D1, D2, D3 , D7, D10, D13, D15, D17
G3	**Allow farmers and agronomist to exchange suggestions**
	- R3, R4, R6, R7, R8, R9, R10, R11, R12, R13, R14, R15, R16, R17, R18 - D2, D3, D4 , D5, D6, D7, D8, D9, D10, D11
G4	**Provide farmers' performance evaluations**
	- R1, R5, R13, R16, R19, R21, R22, R23 - D8, D10, D13
G5	**Help farmers to improve their production**
	- R3, R4, R6, R11, R15, R16, R17, R18, R19, R20, R21, R22, R23, R24, R25 - D1, D8, D9, D14, D17

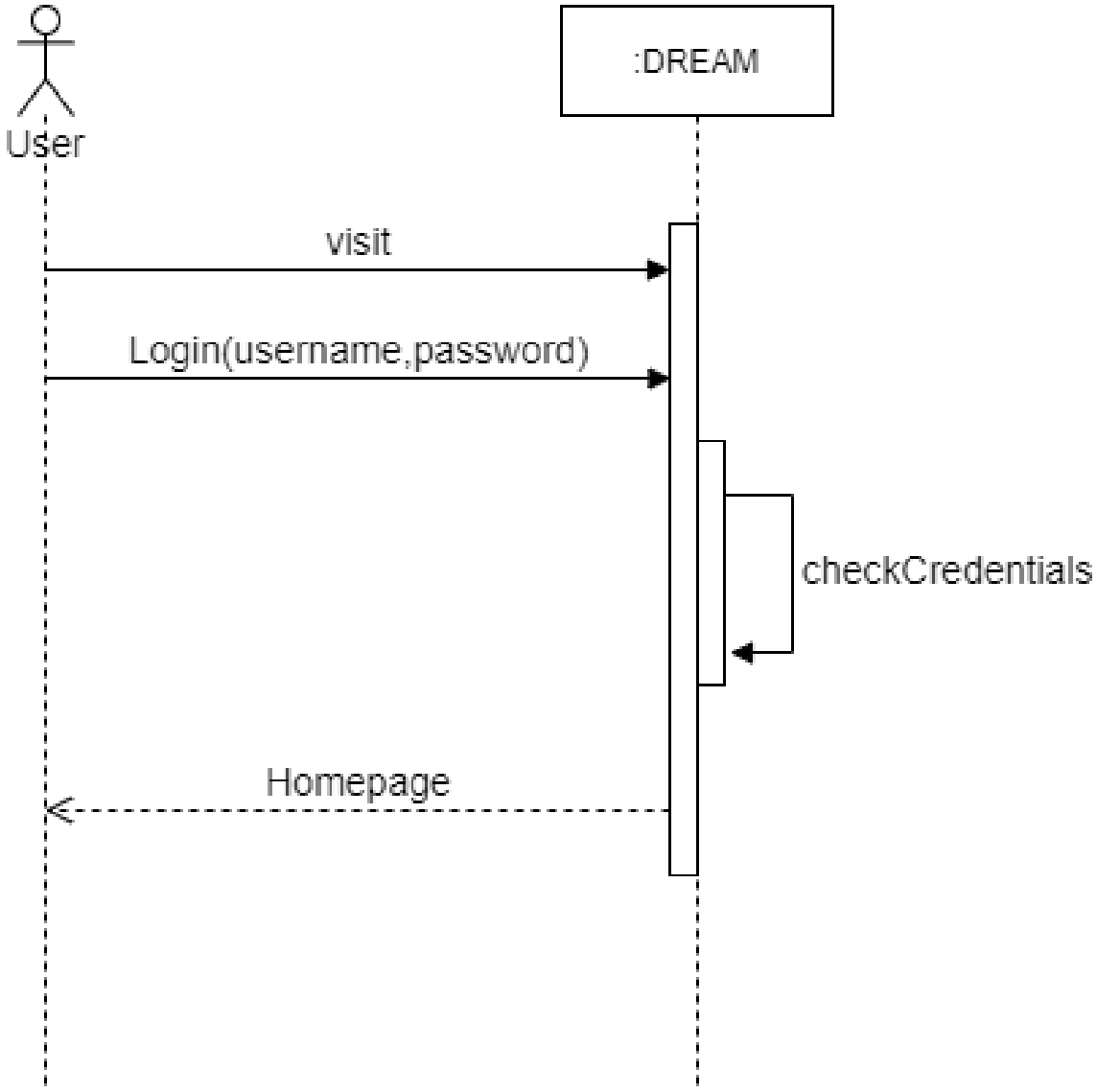
B.2 Use Cases Diagram



B.3 Use cases

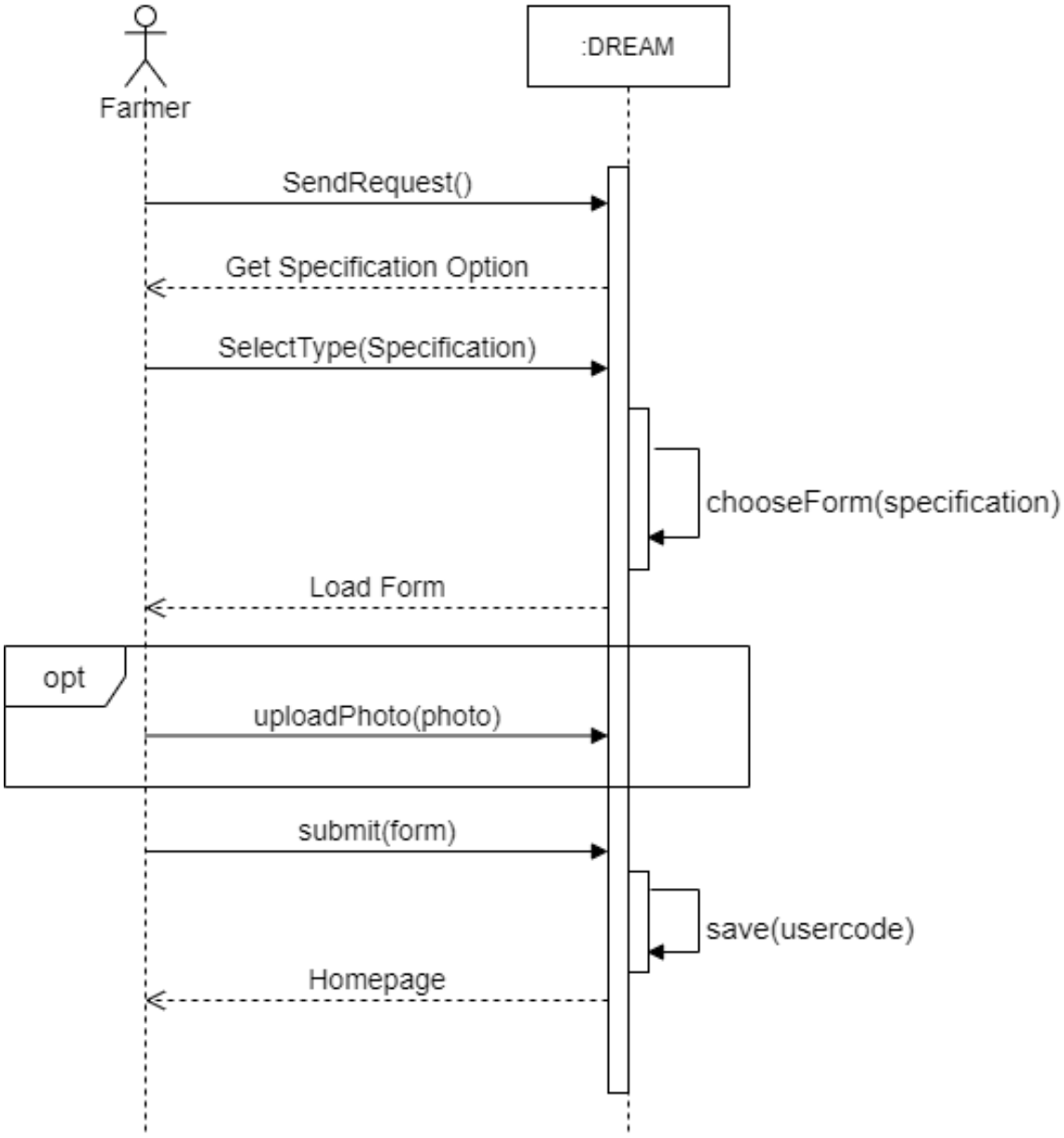
Name	Log in
Actor	Farmer, Agronomist, Policy Maker
Entry Condition	User need to access the app
Event Flow	1. Dream displays a login page to access 2. User insert his personal code and password 3. Dream redirect the user to the Homepage
Exit Condition	System display the Homepage
Exception	

Login


User
:DREAM
visit
Login(username,password)
checkCredentials
Homepage

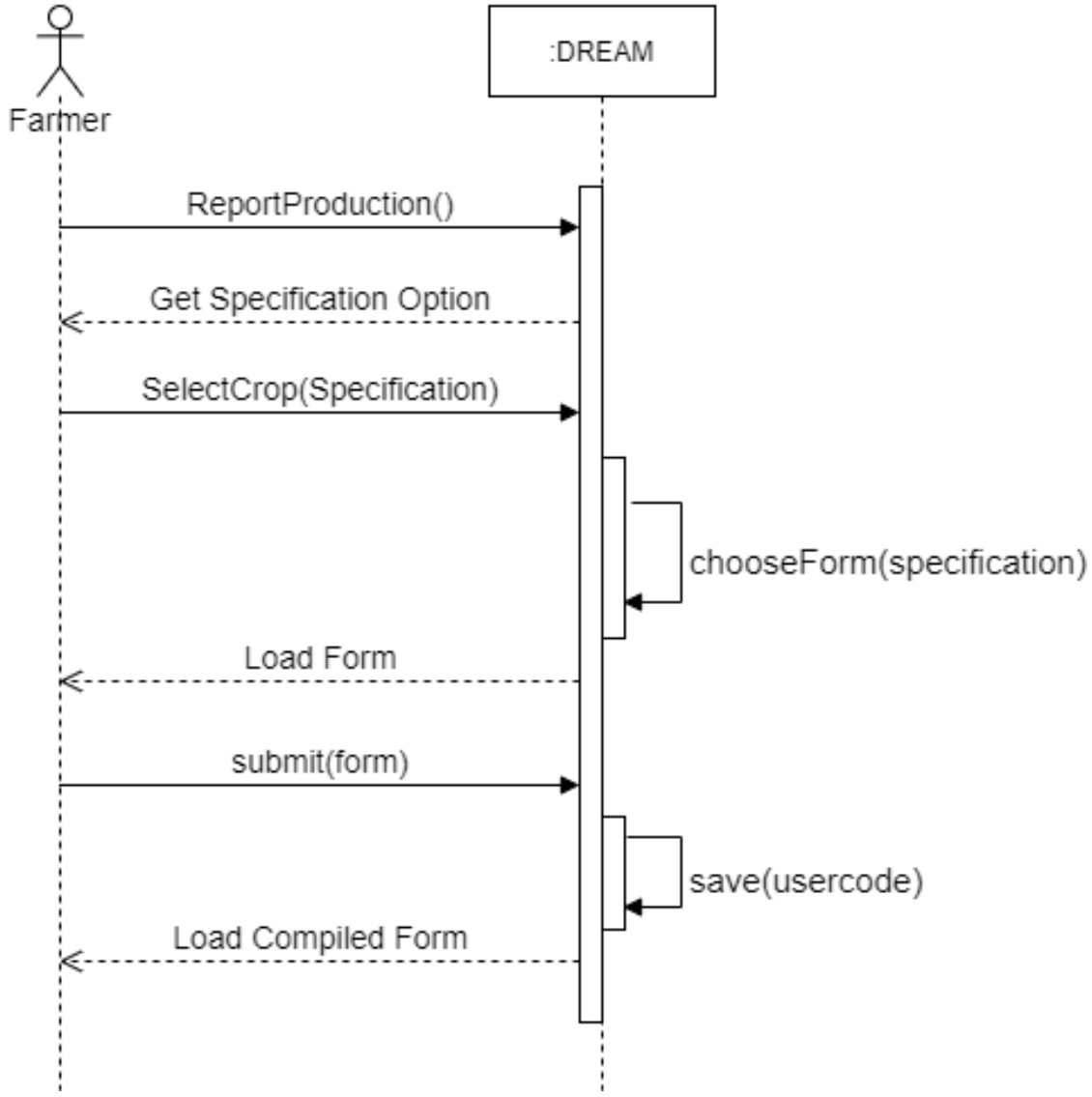
Name	Send a ticket
Actor	Farmer
Entry Condition	Farmer need Agronomist's intervention
Event Flow	1. In HomePage, Farmer click on the button "Send Request" 2. DREAM shows a page asking to specifies the type of request 3. Farmer specifies the type of request 4. DREAM displays a form to submit according to its type 5. Farmer compiles the forms fields 6. Farmer can upload photos if necessary 7. Farmer submit the form 8. DREAM saves the request assigning it to the sender personal code
Exit Condition	System confirms the request has been sent and goes back to the Homepage
Exception	If Farmer does not log in, it cannot access the form page. Farmer must fill all the fields of the form, otherwise the system does not allow the submit

Send a ticket



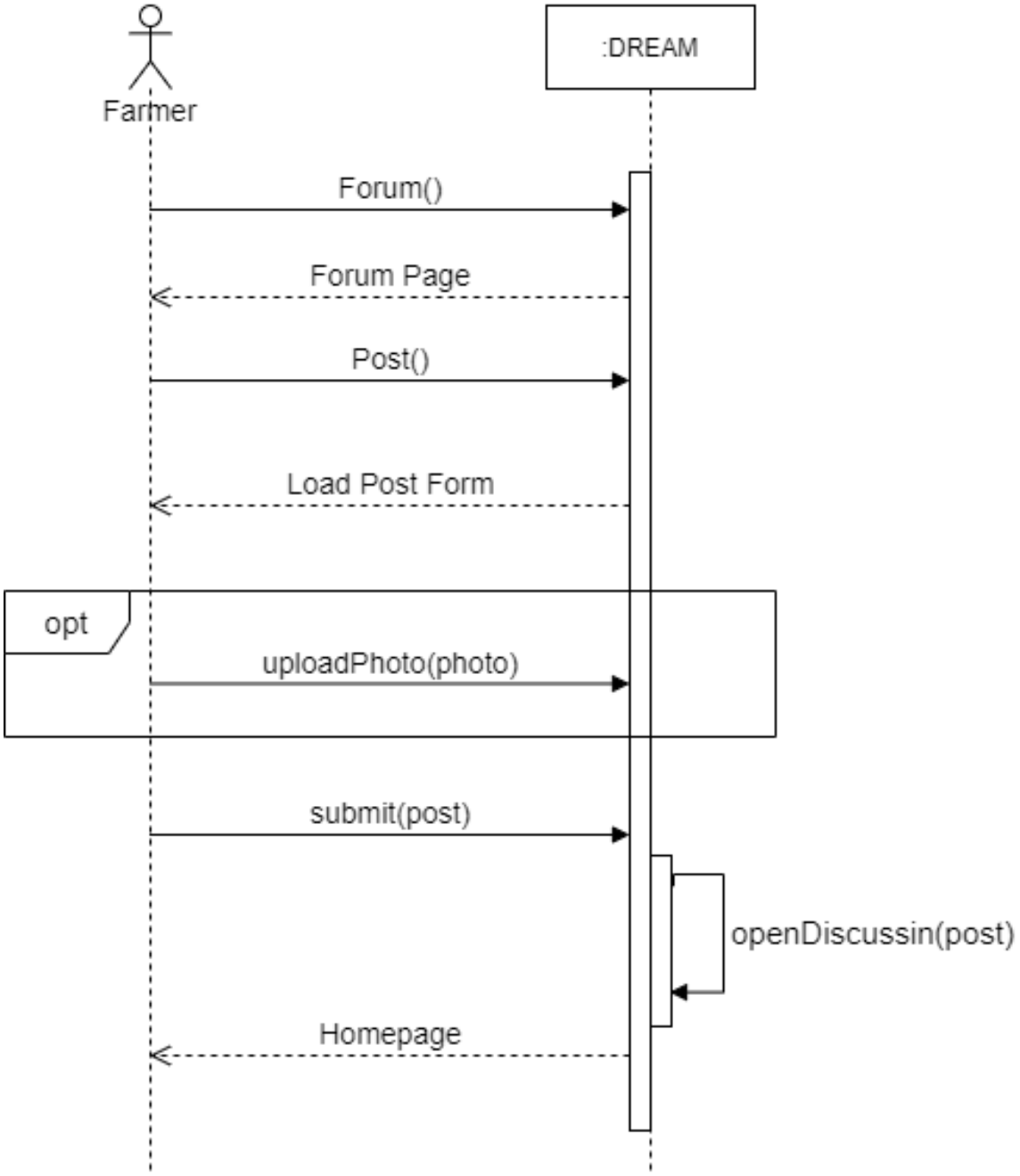
Name	Insert Production Data
Actor	Farmer
Entry Condition	Farmer finish its production and has to report it
Event Flow	1. In HomePage, Farmer click on the button "Report Production" 2. Dream shows a page asking the Farmer to specify the type of crop 3. Farmer select the type of crop 4. DREAM displays a form to compile 5. Farmer compiles the forms fields 6. Farmer submit the form 7. DREAM saves the form assigning it to the sender personal code
Exit Condition	System confirm that the report has been sent and displays the form compiled
Exception	If Farmer does not log in, it cannot access the form page. Farmer must fill all the fields of the form, otherwise the system does not allow the submit

Insert production data



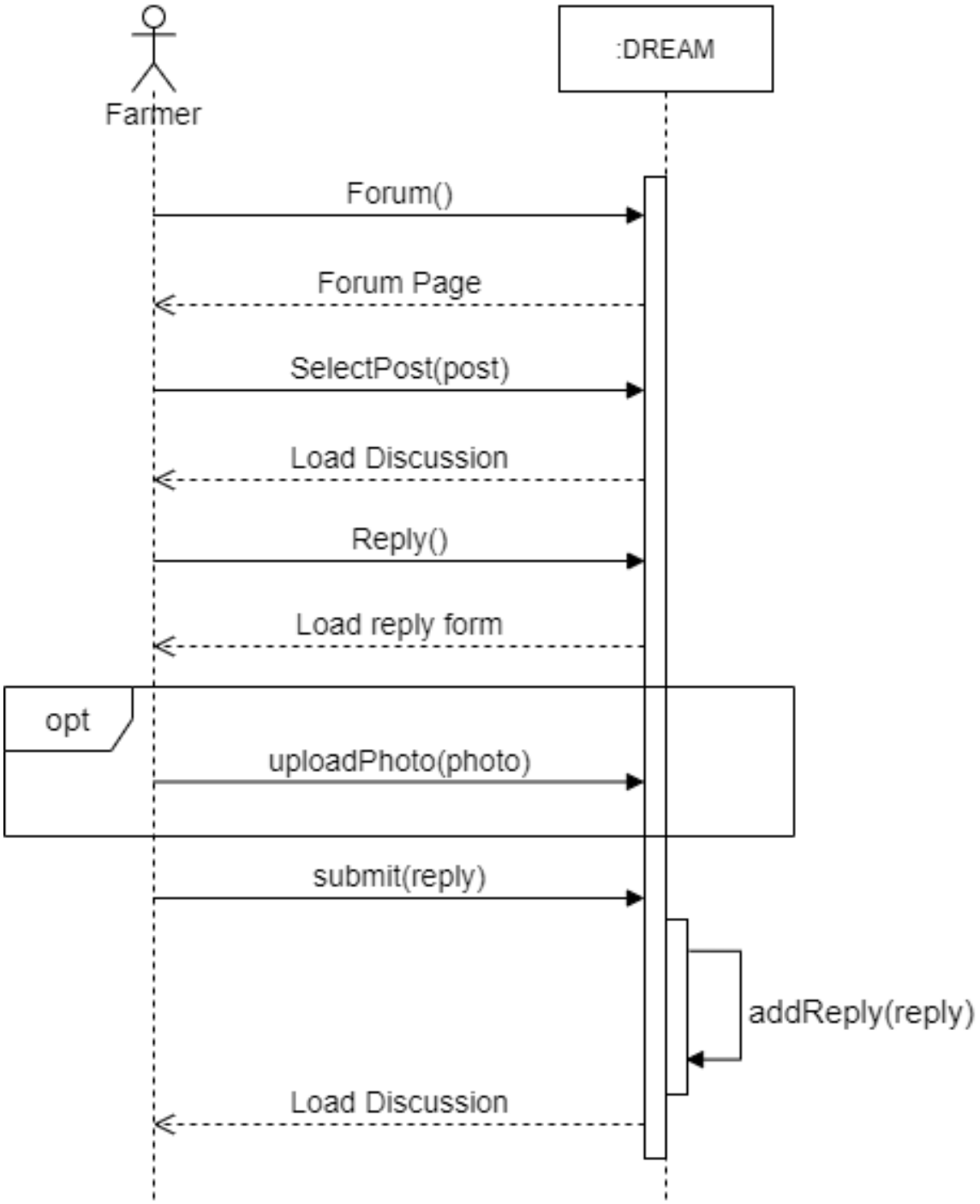
Name	Create a discussion
Actor	Farmer
Entry Condition	Farmer wants to post in the forum
Event Flow	1. In HomePage, Farmer click on the button "Forum" 2. Dream displays the Forum page with all the open discussion 3. Farmer click on "Post" 4. DREAM displays a form with an object and a description field 5. Farmer compile all the fields 6. Farmer can upload a photo if necessary 7. Farmer submit the form 8. DREAM open a new a discussion on the forum
Exit Condition	System displays the updated list of open discussion
Exception	If Farmer does not log in, it cannot access the forum page. Farmer must fill all the fields of the form, otherwise the system does not allow the submit

Create a discussion



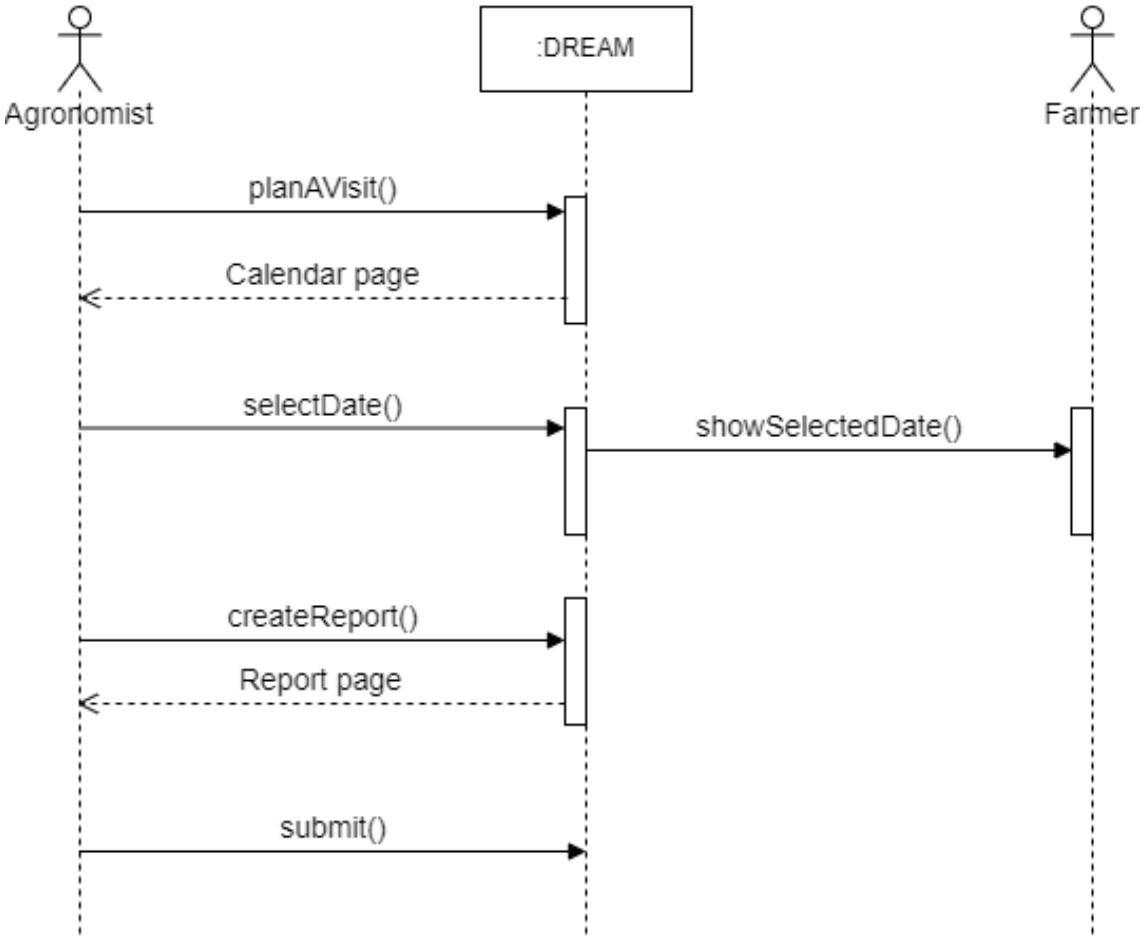
Name	Partecipate to a discussion
Actor	Farmer
Entry Condition	Farmer wants to reply to a open discussion
Event Flow	1. In HomePage, Farmer click on the button "Forum" 2. Dream displays the Forum page with all the open discussion 3. Farmer select an open discussion 4. Dream shows the selected discussion with all the replies to it 5. Farmer click on "Reply" 6. DREAM displays an answer textbox 7. Farmer write the answer 8. Farmer can upload a photo if necessary 9. Farmer send the answer 10. DREAM add the reply to the open discussion
Exit Condition	System displays the discussion post and the updated list of replies
Exception	If Farmer does not log in, it cannot access the forum page. Farmer must fill all the fields of the form, otherwise the system does not allow the submit

Partecipate to a discussion

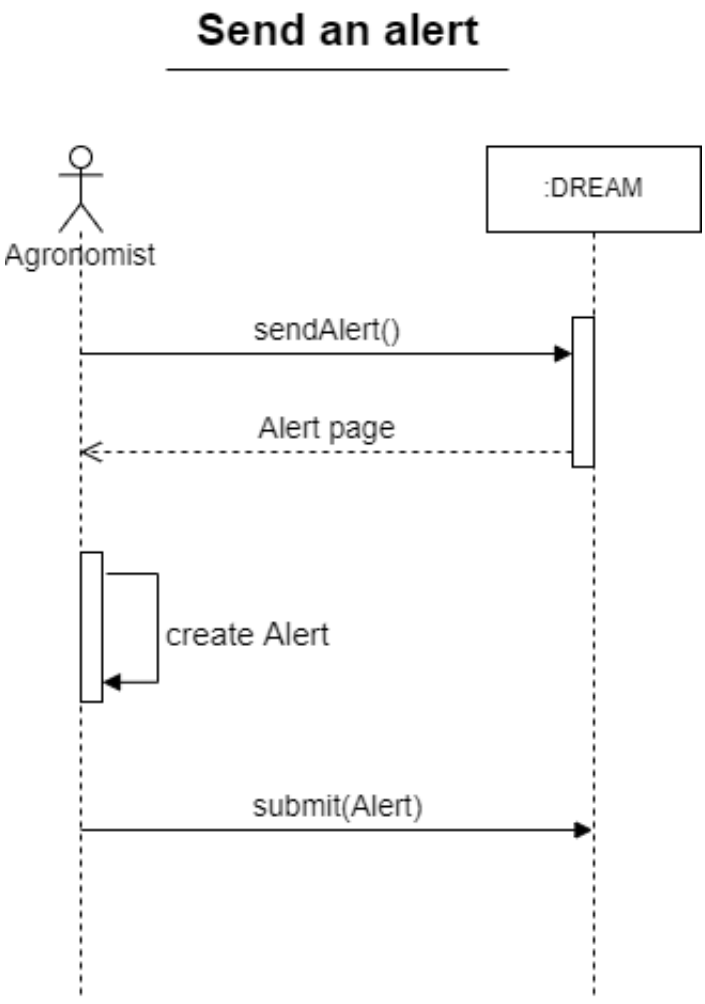


Name	Visit farmers
Actor	Agronomist, Farmer
Entry Condition	Agronomist wants to visit a farm
Event Flow	1. Dream displays the Agronomist page 2. Agronomist click on the button "Plan a visit" 3. DREAM displays a monthly calendar to show the available dates 4. Agronomist select the date for the visit 5. Dream sends a notification to the Farmer 6. Farmer receives visit details 7. After visiting, Agronomist compile the report of the daily plan 8. DREAM saves the report
Exit Condition	System update Agronomist's daily plan
Exception	If user does not log in, it cannot access the page. The system does not allow the Agronomist to plan more than one visit at the time

Plan a visit

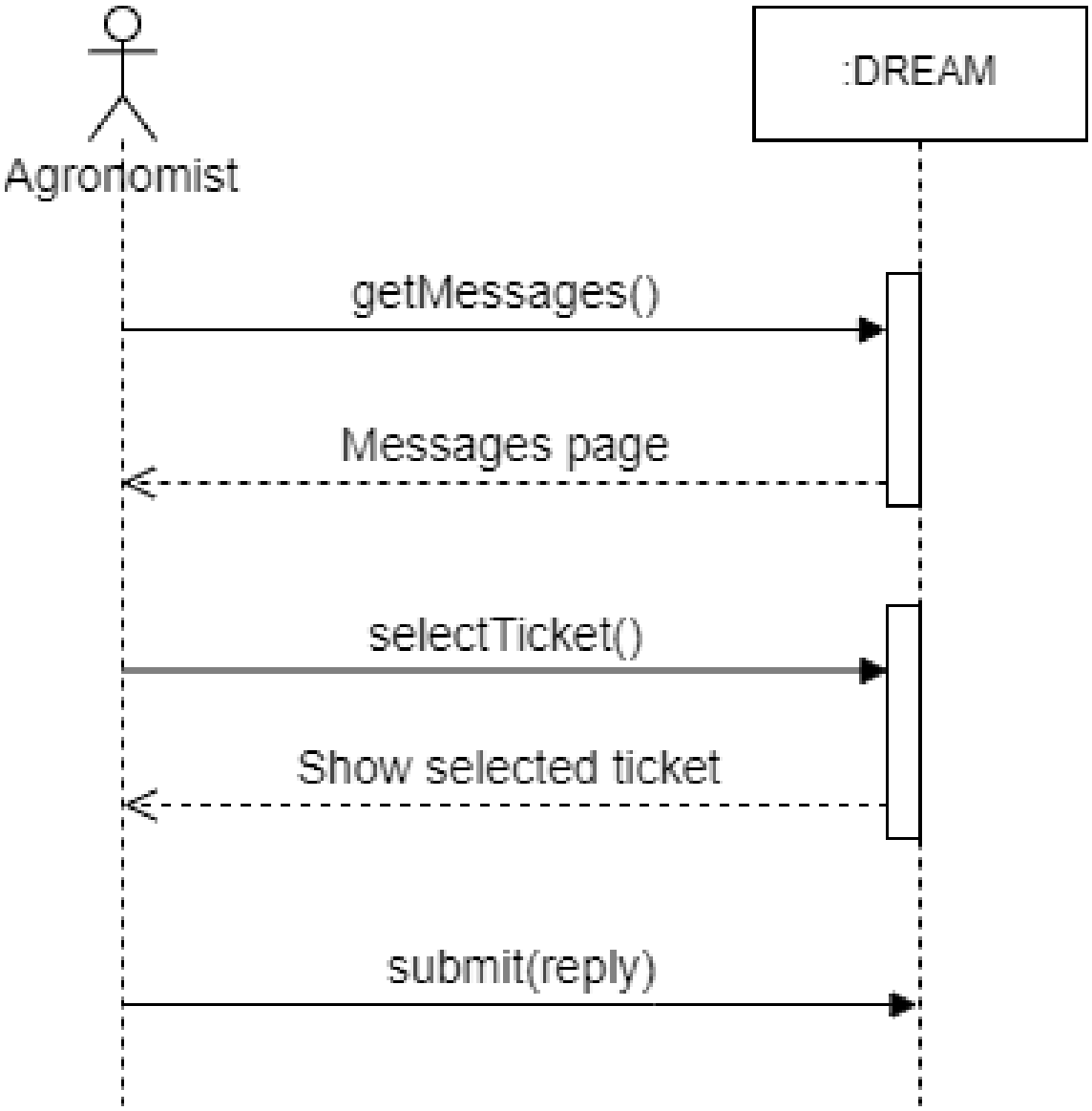

:DREAM
Agronomist
Farmer
planAVisit()
Calendar page
selectDate()
showSelectedDate()
createReport()
Report page
submit()

Name	Send an alert
Actor	Agronomist, Farmer
Entry Condition	Agronomist wants to notify an emergency
Event Flow	1. Dream displays the Agronomist page 2. Agronomist click on the button "Send Alert" 3. DREAM shows a page to create the alert 4. Agronomist describe the incoming emergency 5. Agronomist click "submit"
Exit Condition	System dispatches the alert to all land's farmers
Exception	If Agronomist does not log in, it cannot access the form page. Agronomist must specify the land he manages, otherwise the alert cannot be sent



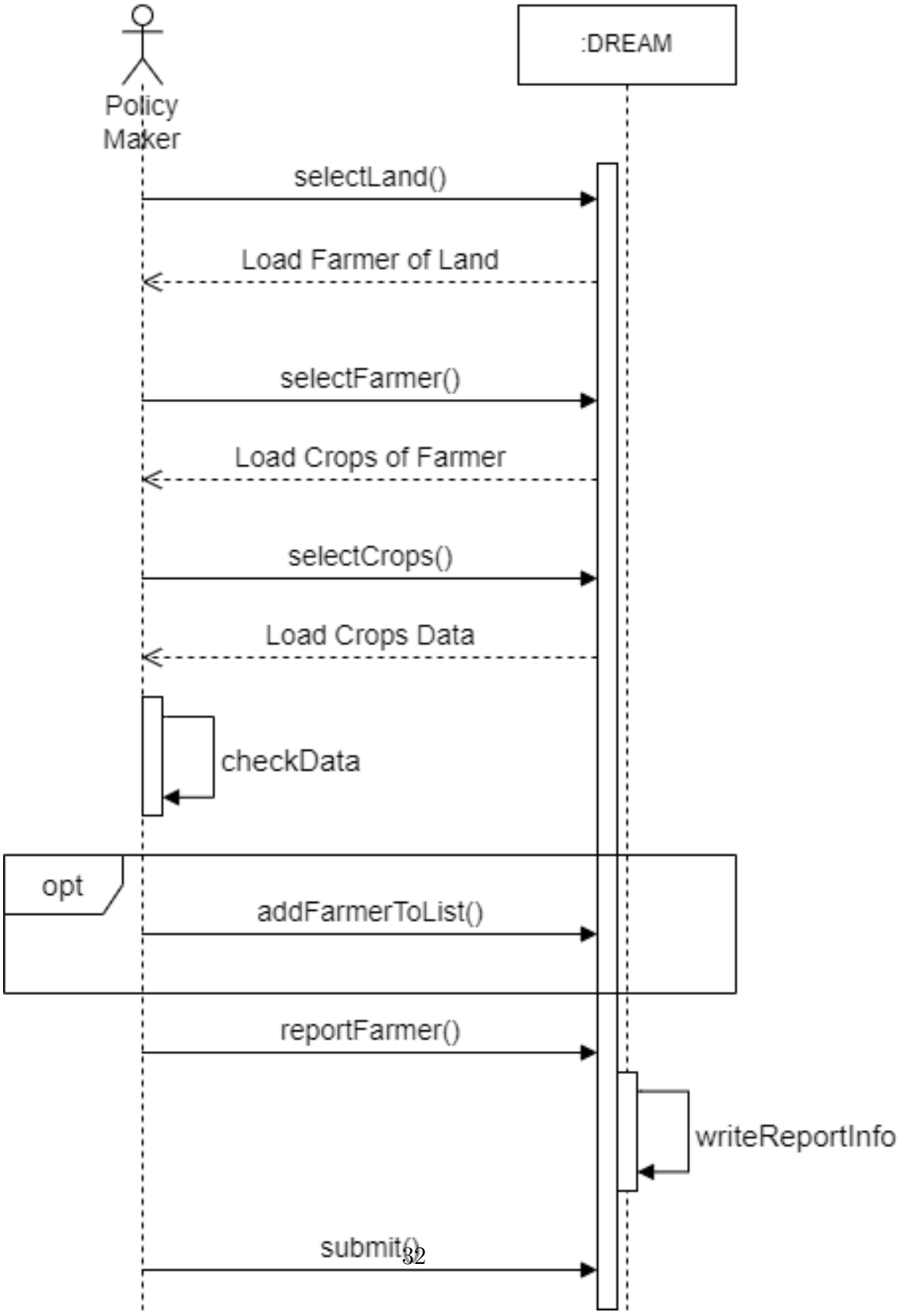
Name	Reply to a ticket
Actor	Agronomist
Entry Condition	Agronomist receive a ticket from a Farmer
Event Flow	1. Dream displays the Agronomist page 2. Agronomist click on button "Messages" 3. Dream displays all the upcoming tickets 4. Agronomist select a ticket to reply 5. Dream opens the selected ticket showing its content 6. Agronomist insert the reply to the ticket 7. Agronomist click "Submit"
Exit Condition	System dispatches the reply to the ticket's sender
Exception	If Agronomist does not log in, it cannot access the page. If an Agronomist has already replied to a ticket, it cannot reply again to that ticket

Reply to a ticket



Name	Report performance
Actor	Policy Maker
Entry Condition	Policy Maker wants to notifies a farmer's performance
Event Flow	1. Dream displays the Policy Maker page, with an overview of all the farmers 2. The PM can click on a farmer to visualize all the data and statistics associated 3. Optionally, the PM can visualize the past weather condition 4. The PM identify the well or bad performing farmers 5. Optionally, the PM can insert the well or bad performing farmers on a list 6. Policy Maker click on "Report" 7. Dream displays a form page of the Policy Maker 8. The PM write the send to the Agronomist 9. The PM submit the report
Exit Condition	A report to the Agronomist has been sent
Exception	If Policy Maker does not log in, it cannot access the page.

Report performance



C Performance Requirements - Non-functional Requirements

1. The land visit by Agronomist have to be notified at least 24 hours in advance.
2. At least a farmer in every land has to daily check the tablet to keep in track with eventual notifications or alerts.

D Design Constraints

D.1 Hardware limitations

In order to maintain the tablet operating on a long term, it is necessary to have a central charge unit in every land.

D.2 Any other constraint

Every tablet has a serial number registered that keep track of the devices in use on the region, in order to monitor its status and avoid case of theft.

E Software System Attributes

E.1 Reliability

The system must be fault tolerant. The architecture should be redundant in order to guarantee the reliability. For example, backup data and duplication of the running process might be kept, in order to quickly recover in case of damage.
Agronomist should have backup industrial tablet. Indeed, it may happen an industrial tablet in a land broke up. The Agronomist have to be able to detect the loss of connection and bring the backup tablet to the land as soon as possible.

E.2 Availability

Considering both critical (alerting about bad weather condition) and non-critical aspects (farmer who inserts production data), the system should be available at least the 99.9% of the time.

E.3 Security

All the personal code and password used for access the system by the Policy Makers, Agronomists and Farmers should all be encrypted.
The system should be protected against non authorized access.

E.4 Maintainability

The application must have an high level of maintainability. Appropriate design pattern have to be used, in order to make it easy fixing and modifying the code.
An error logging system must be provided in order to quickly solve any problem faced.

E.5 Portability

The system must be able to run on different platform. Indeed, the Farmer will access it from the industrial tablet, while the Policy Makers and the Agronomists can choose whether access it from a smartphone or a desktop computer. The access from the smartphone is particularly important for the Agronomist since he has to be able to change the daily plan from everywhere he is in case of an unexpected event.

4 Formal Analysis using Alloy

This section is dedicated to a model of the project developed with the Alloy tool. The model focus on the main entities of the system: Agronomist, Policy Maker, Farmer, Ticket Request, Daily Plan, Alert, Report. Entities like Calendar, Land and Tablet are modelled accordingly to their use in relation to the main entities.

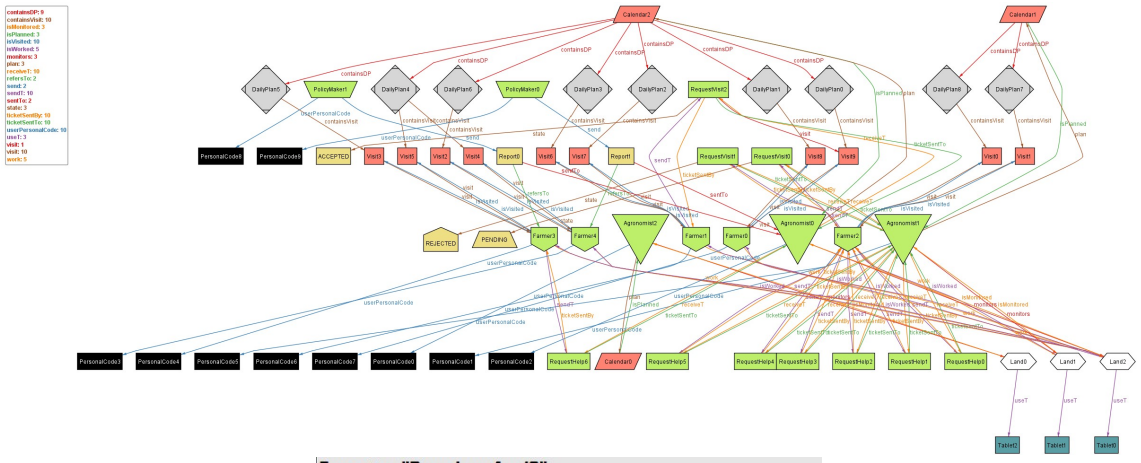
```
sig PersonalCode{}

abstract sig User{
        userPersonalCode: one PersonalCode
}

sig Farmer extends User{
        sendT: set Ticket,
        work: one Land,
        isVisited: set Visit
} {#isVisited > 1}

sig Agronomist extends User{
        monitors: one Land,
        sendA: set Alert,
        plan: one Calendar,
        receiveT: set Ticket
}

sig PolicyMaker extends User{
        send: some Report
}

sig Report{
        refersTo: one Farmer,
        sentTo: one Agronomist
}

sig Land{
        isWorked: set Farmer,
        isMonitored: one Agronomist,
        useT: one Tablet
}

abstract sig State{}
sig ACCEPTED extends State{}
sig PENDING extends State{}
sig REJECTED extends State{}


abstract sig Ticket{
        ticketSentBy: one Farmer,
        ticketSentTo: one Agronomist
}

sig RequestVisit extends Ticket{
        state: one State,
```

```
        visit: lone Visit
}

sig RequestHelp extends Ticket{}

sig Visit{
        visit: one Farmer
}

sig DailyPlan{
        containsVisit: set Visit,
} {#containsVisit ≤ 5}

sig Calendar{
        containsDP: set DailyPlan,
        isPlanned: one Agronomist
}

sig Tablet{}

sig Alert{
        alertSentTo: one Land
}

////////////////////// FACTS \\\\\\\\\\\\\\\\\\\\

fact {
        plan= ~isPlanned                //no Agronomist plan different Calendars
        isVisited = ~visit              //no Farmer visited by different Visit
        isWorked =  ~work               //no Farmer that works in differents lands
        isMonitored = ~monitors         //no Agronomist monitors different Lands
        ticketSentBy = ~sendT           //no Ticket sent from different Farmers
        ticketSentTo = ~receiveT        //no Ticket sent to different Agronomist
}

//every User's personal code is unique and belongs to only one user
fact noEqualPersonalCode{
        all disj u,u':User | u.userPersonalCode ≠ u'.userPersonalCode
}

//every personal code must be related to an User
fact noPersonalCodeWOUser{
        all pc:PersonalCode | let u=User | (pc in u.userPersonalCode)
}

// A Report sent by a Policy Maker have to be sent to the Agronomist of the Farmer
fact reportToCorrectAgronomist{
        all r:Report | all f:Farmer | (f in r.refersTo) implies (f.work.isMonitored
            ↪  in r.sentTo)
}

 //A Ticket sent by a Farmer can only be sent to the Agronomist of the Farmer's
    ↪ Land
fact ticketToCorrectAgronomist{
        all f:Farmer, t:Ticket | (t in f.sendT) implies (t in receiveT[isMonitored[
            ↪ f.work]])
}

//There is no Report without a PolicyMaker
fact noReportWOPolicyMaker{
        all r:Report | one p:PolicyMaker | r in p.send
}

//There is no Calendar without an Agronomist
fact noCalendarWOAgronomist{
        all c:Calendar | one a:Agronomist | c in a.plan
}

//There is no Dailyplan withouth a Calendar
fact noDailyPlanWOCalendar{
```

```
        all dp: DailyPlan | one c: Calendar | dp in c.containsDP
}

//There is no Visit without a DailyPlan
fact noVisitWODailyPlan{
        all v: Visit | one d: DailyPlan | v in d.containsVisit
}

//There is no Alert without an Agronomist
fact noAlertWOAgronomist{
        all al: Alert | one ag: Agronomist | al in ag.sendA
}

//There is no Tablet without a Land and the Tablet is unique
fact noTabletWOLand{
        all disj l1,l2: Land | l1.useT ≠ l2.useT
        all t: Tablet | one l: Land | t in l.useT
}

fact alertAgronomist{
        //An Alert can't be sent by two different Agronomist
        all al: Alert | (no disj a1,a2: Agronomist | al in a1.sendA and al in a2.
            ↪ sendA)
         //The Alert has to be sent to the Land monitored by the Agronomist who
             ↪ sent the Alert
        all al: Alert | all l: Land | (al in l.isMonitored.sendA) implies (al.
            ↪ alertSentTo in l)
}

//Create a Visit  to visit the Farmer if the VisitRequest has been ACCEPTED
fact createVisitIfRequestAccepted{
        all r: RequestVisit |  r.state = ACCEPTED implies (one v: Visit, f: Farmer | v
            ↪  in r.visit and v in f.isVisited)
}

fact noStateWTwoRequest{
        //A State can't be of two different VisitRequest
        all s: State | (no disj r1,r2: RequestVisit | s in r1.state and s in r2.state
            ↪ )
         //Each State has to belong to a unique VisitRequest
        all s: State | one r: RequestVisit | s in r.state
}

fact visitFarmer {
        //A Visit can't be contained in two different DailyPlans
        all v: Visit | (no disj d1,d2: DailyPlan | v in d1.containsVisit and v in d2.
            ↪ containsVisit)
and     // In a DailyPlan, there can be only one Visit per Farmer
        all f: Farmer | all dp: DailyPlan | (f in dp.containsVisit.visit) implies (no
            ↪  disj v1,v2: Visit | v1 in dp.containsVisit and v2 in dp.
            ↪ containsVisit and f in v1.visit and f in v2.visit)
and     // The Visit to a Farmer has to be done only from the Agronomist of the
    ↪ Farmer
        all v: Visit | all f: Farmer | (f in v.visit) implies (v in f.work.
            ↪ isMonitored.plan.containsDP.containsVisit )
}

pred show{
        #PolicyMaker = 2
        #Agronomist = 3
        #Farmer = 5
        #RequestVisit = 3
        #ACCEPTED = 1
}

run show for 10



////////////////////// ASSERTIONS \\\\\\\\\\\\\\\\\\\\
```

```
// Visiting the same Farmer more than once in the same DailyPlan is not allowed
assert noVisitSameFarmerSameDailyPlan{
        no f:Farmer | all d:DailyPlan |(all disj v1,v2:Visit | v1 in f.isVisited
            ↪ and v2 in f.isVisited  and v1 in d.containsVisit and v2 in d.
            ↪ containsVisit)
}

// An Agronomist can't visit a Farmer which does not work in the Land he monitors
assert noVisitFromAgronomistOfOtherLand{
        no f:Farmer | all a:Agronomist | f.work not in a.monitors and a.plan.
            ↪ containsDP.containsVisit in f.isVisited
}

// A Visit can't be contained in two different DailyPlan simultaneously
assert noVisitInTwoDailyPlan{
        no v:Visit | (all disj d1,d2:DailyPlan | v in d1.containsVisit and v in d2.
            ↪ containsVisit)
}

// A Ticket of a Farmer can't be sent to an Agronomist which monitors a Land in
    ↪ which the Farmer does not work
assert noTicketSentToAgronomistOfOtherLand{
        no t:Ticket | all f:Farmer | t in f.sendT and f.work not in t.ticketSentTo.
            ↪ monitors
}

// A Report of a PolicyMaker can't be send to an Agronomist which monitors a Land
    ↪ in which the Farmer (subject of the report) does not work
assert noReportSentToAgronomistOfOtherLand{
        no r:Report | r.sentTo.monitors not in r.refersTo.work
}

// A Visit generated by an ACCEPTED RequestVisit can't be done to a Land in which
    ↪ the Farmer (who sent the RequestVisit) does not work
assert noVisitDueToRequestVisitToFarmerOfOtherLand{
        no v:Visit | one r:RequestVisit | one a:Agronomist | one f:Farmer | a not
            ↪ in f.work.isMonitored and a in r.ticketSentTo and f in r.
            ↪ ticketSentBy and v in r.visit and f in v.visit
}

check noVisitDueToRequestVisitToFarmerOfOtherLand


////////////////////// PREDICATES \\\\\\\\\\\\\\\\\\\\

pred policyMakerCreateReport[p:PolicyMaker, r:Report, a:Agronomist]{
        p.send = r
        r.sentTo = a
        one f:Farmer | (f in a.monitors.isWorked) implies r.refersTo = f
}

pred isDailyPlanNotFull[d:DailyPlan]{
        #(d.containsVisit) < 5
}

pred agronomistCreateAVisit[a:Agronomist, d:DailyPlan, v:Visit, f:Farmer]{
        isDailyPlanNotFull[d]
        v in d.containsVisit
        f.work.isMonitored = a
        a.plan.containsDP = d
        implies
        f.isVisited = v
}

pred agronomistCreateAlert[ag:Agronomist, al:Alert, f:Farmer, l:Land]{
        ag in l.isMonitored
        f in l.isWorked
        ag.sendA = al
```

```
        al.alertSentTo = l
}

pred farmerMakeARequestHelp[f: Farmer, r: RequestHelp, a: Agronomist]{
        a in f.work.isMonitored
        r.ticketSentTo = a
        f.sendT = r
}

pred farmerMakeARequestVisitWhichIsAccepted[f: Farmer, r: RequestVisit, a: Agronomist,
    ↪ v: Visit]{
        a in f.work.isMonitored
        r.ticketSentTo = a
        f.sendT = r
        r.state = ACCEPTED
        r.visit = v
        v.visit = f
}

run farmerMakeARequestVisitWhichIsAccepted
```

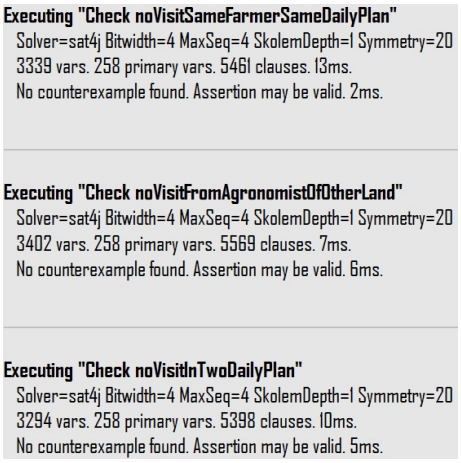


Executing "Run show for 10"
Solver=sat4j Bitwidth=4 MaxSeq=7 SkolemDepth=1 Symmetry=20
58950 vars. 2460 primary vars. 100523 clauses. 141ms.
Instance found. Predicate is consistent. 267ms.

A Assertions

The Alloy model wants also to prove the following Assertions

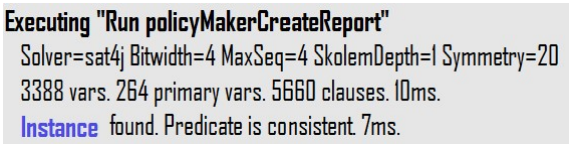
- a Visit to the same Farmer more than once per Daily Plan is not allowed
- an Agronomist can't Visit a Farmer which does not work in the Land he monitors
- a Ticket of a Farmer can't be sent to an Agronomist which monitors a Land in which the Farmer does not work
- a Report of a Policy Maker can't be sent to an Agronomist which monitors a Land in which the Farmer (subject of the Report) does not work
- a Visit due to an Accepted Visit request can't be done in a Land in which the Farmer (who sent the Visit Request) does not work

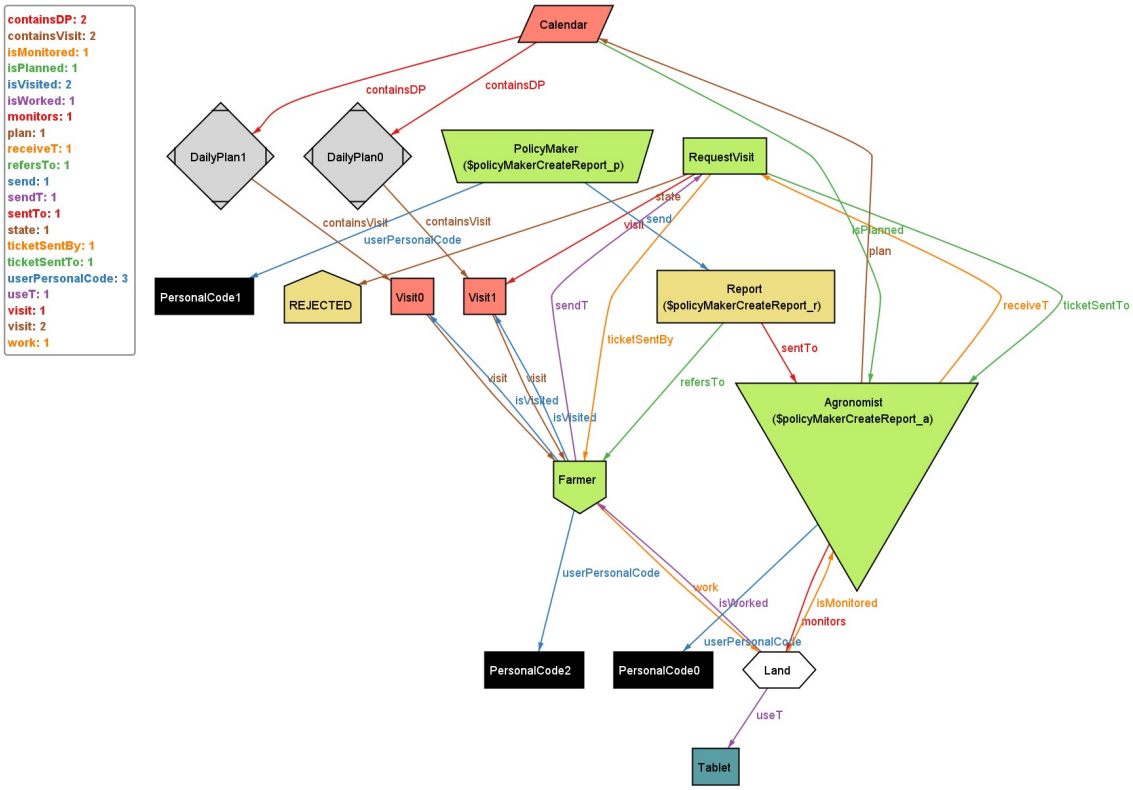


B Predicates

The features the Alloy model is going to show through Predicates are

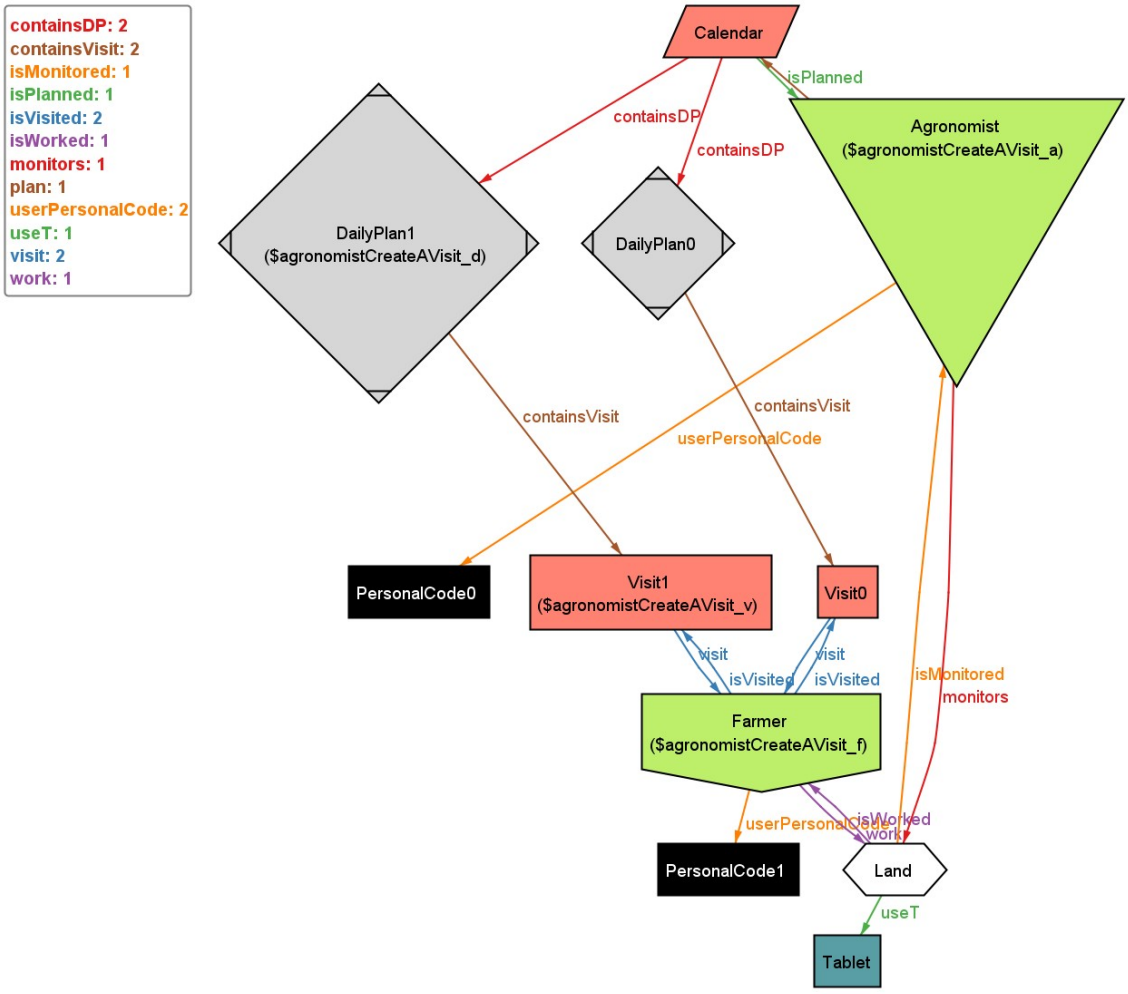
- A Policy Maker creates a Report
- An Agronomist creates a Visit to visit a farmer
- An Agronomist creates an Alert to be sent to the Land he monitors
- A Farmer make a Request of Help
- A Farmer make a request of Visit which is Accepted by the Agronomist; that means, the Agronomist has a Visit event with the Farmer who made the request.





Executing "Run agronomistCreateAVisit"

Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
3429 vars. 267 primary vars. 5735 clauses. 10ms.
Instance found. Predicate is consistent. 10ms.

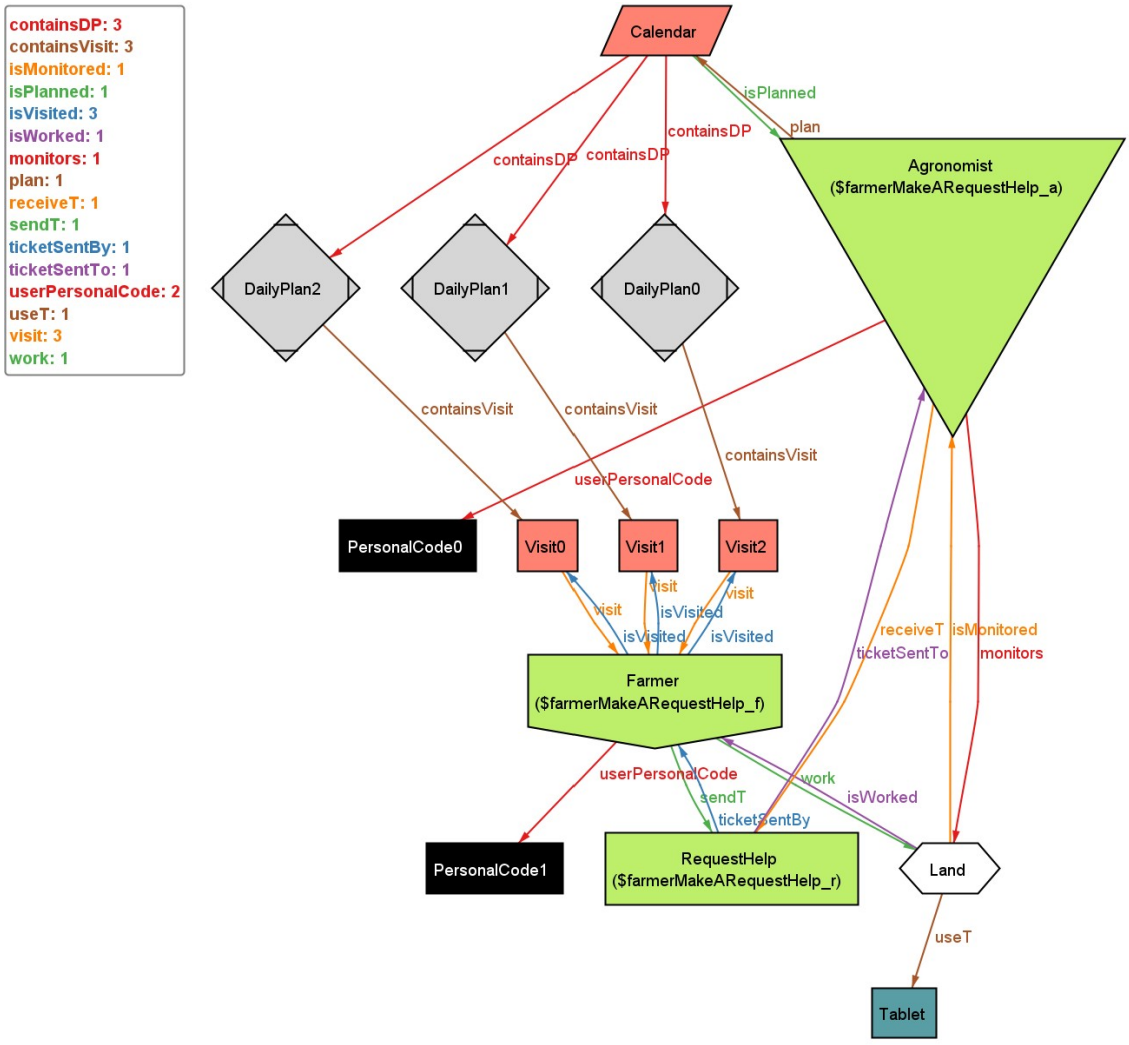


Executing "Run farmerMakeARequestHelp"

Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20

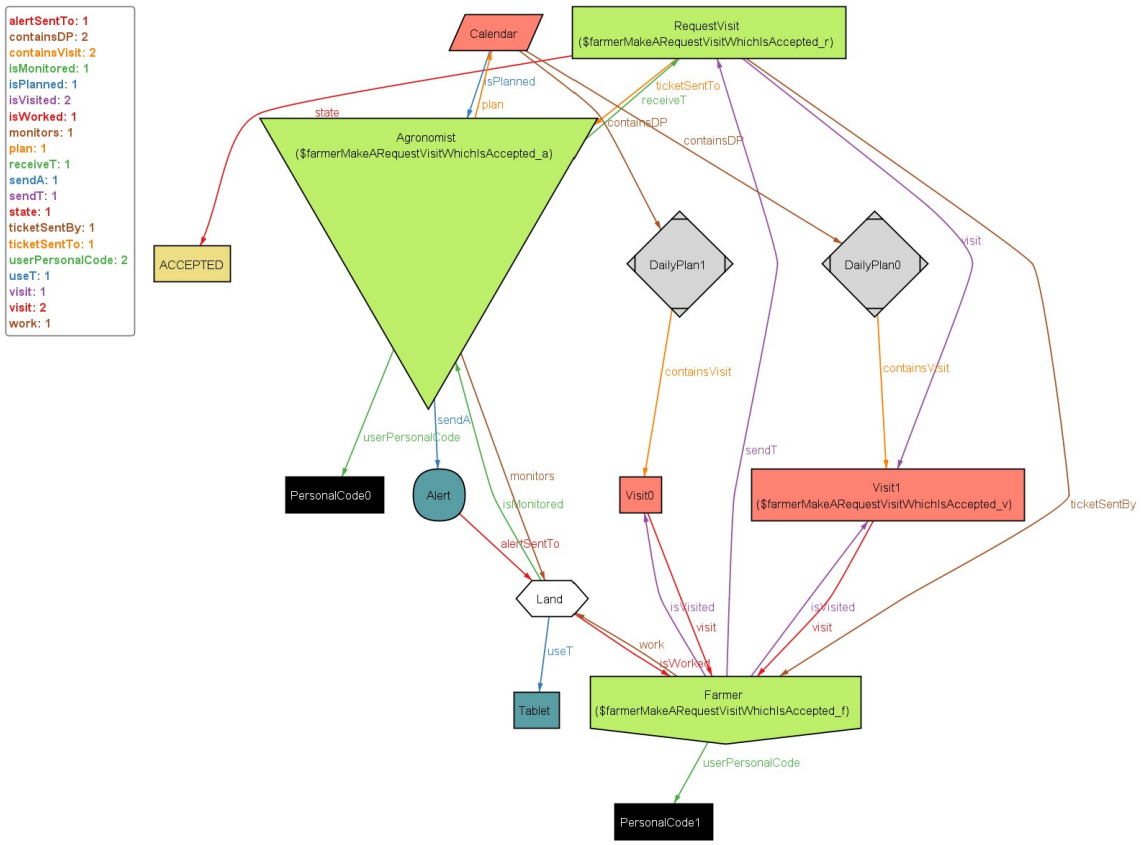
3363 vars. 264 primary vars. 5546 clauses. 10ms.

Instance found. Predicate is consistent. 6ms.



Executing "Run farmerMakeARequestVisitWhichIsAccepted"

Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
3449 vars. 267 primary vars. 5742 clauses. 10ms.
Instance found. Predicate is consistent. 14ms.



5 Effort Spent

In this section you will include information about the number of hours each group member has worked for this document.

RASD					
Gabriele		Marco		Michele	
Topic	Ore	Topic	Ore	Topic	Ore
General Reasoning	7	General Reasoning	7	General Reasoning	7
Purpose & Scope	2	Purpose & Scope	2	Purpose & Scope	3
Class Diagram	2	Class Diagram	2	Class Diagram	3
Statechart	1	Statechart	1	Statechart	2
Product Function	10	Product Function	5	Product Function	5
Domain Assumptions	3	Domain Assumptions	4	Domain Assumptions	4
Functional Requirements	3	Functional Requirements	4	Functional Requirements	4
Use Cases & Use Cases Diagrams	3	Use Cases & Use Cases Diagrams	4	Use Cases & Use Cases Diagrams	5
Sequence diagrams	8	Sequence diagrams	5	Sequence diagrams	4
Alloy	33	Alloy	5	Alloy	5
Document organization	3	Document organization	5	Document organization	3

6 References

Specification document: "RDD Assignment AY 2021-2022.pdf"