DREAM

Data-dRiven PrEdictive FArMing in Telengana

Requirement Analysis and Specification Document - RASD

Christian Grasso - Filippo Lazzati - Chiara Magri

Year: 2021/2022

Contents

List of Tables

1 Introduction

1.1 Purpose

1.2 Scope

1.3 Definitions, Acronyms, Abbreviations

1.4 Revision history

1.5 Reference Documents

1.6 Document Structure

2 Overall description

2.1 Product perspective

2.2 Product functions

The main goal that Telengana's government wants to achieve with `DREAM` is to help policy makers formulating policies in the field of agriculture. In order to accomplish this objective, Telengana's government is asking for predictive models for food systems that can drive decisions exploiting huge amount of data. In this section, a list of the most important requirements of the system is provided; notice that they are just briefly described, since they will be analysed in-depth in chapter 3.

2.2.1 Data collection

Just like every data-driven decision, data is essential. `DREAM` must be able to manage different kinds of data coming from different sources:

1. agronomists and farmers insert into the system standard kind of information about the farms they respectively visit and own, but also other kinds of information like, for example, the types of products cultivated and the production volume for each product (kinds of information that allow the analytics to profile these users). `DREAM` also allows them to insert less-structured data, for instance full-text feedback about the inserted data (farmers can insert data about any problem they face);

2. data gathered by sensors deployed on the territory and by the water irrigation system are automatically collected by `DREAM` through effective interfaces interacting with such systems;

3. already existing systems are integrated with `DREAM`;

4. the last but not the least, farmers can interact through discussion forums, that are stored by `DREAM`.

2.2.2 Data analysis

The raw data collected by `DREAM` must be processed before being delivered to the end-user. Therefore, starting with the big volume of information "ingested", various kinds of analytics are performed in order to provide a more aggregate version of the data:

- the suggestions provided to the farmers are customized according to their information;
- information about farmers' production is used to distinguish among farmers who are performing well and those who are not;
- patterns and relations are identified among the data (for instance between weather forecasts and production volumes);
- analytics are also used for quantifying the improvement (if present) in the crop after having adopted agronomists' suggestions.

2.2.3 Forum

As previously mentioned, farmers are allowed to open discussion forums on `DREAM` with other farmers. This allows to keep in touch with other people doing the same job and, hence, to ask for advice in case of needing.

2.2.4 Request and supply of help

There are 4 different ways for a farmer to ask for help: ask for it directly to other farmers, ask in the forum, ask to to the agronomists or waiting for the system to recognize him as in trouble. As far as the first option is concerned, the service is provided through a GUI. On the other side, either farmers who are performing well or agronomists may be reached by the system on behalf of other farmers (or of the system itself) for a request of help. Then the system introduces them. It should be remarked that a farmer can contact only agronomists in the same area.

2.2.5 Daily plan

`DREAM` automatically devises a daily plan for each agronomist consisting in a list of farms that must be visited and provides a possible time schedule for such meetings. Every daily plan must be actively accepted by the agronomist before the involved farmer is notified. Anyway, the agronomist can apply some changes to the schedule, but the system will not approve them if some constraints are violated.

2.3 User characteristics

With regards to the possible actors of `DREAM`, three different main user classes can be identified:

Policy makers: their job is to devise policies to regulate the agricultural production. DREAM helps them providing aggregate data. Policy makers access the system and then receive high-level information about which farmers are performing well and which not, and possible reasons to this fact. Furthermore, DREAM helps to understand the impact of agronomists' clues in farmers' productions;

Farmers: they have a farm and some plots of land to cultivate. They access the system and insert data about themselves, helping DREAM to profile them. They insert data about their productions, the area where they work, eventual problems they must face, questions in the forum ... and so on. On the other side, DREAM puts them in touch with someone who can help them and provides data that may be relevant to them;

Agronomists: an agronomist has to find methods for increasing the production and the quality of the harvest. They insert the area they are responsible of and the they receive data about the best performing farmers and weather forecasts. They can answer to farmers' questions and they receive a raw daily plan about which farmers they should visit.

2.4 Assumptions, dependencies and constraints

3 Specific requirements

3.1 External Interface Requirements

3.2 Functional Requirements

DREAM allows its users to perform many tasks, and DREAM itself is able to interact with other different systems to achieve some results (see section 2.2). In order to provide a summary of the possible situations in which DREAM is involved and used, this paragraph first lists some concrete scenarios [1] and then abstracts from details and specificities showing the corresponding use cases.

3.2.1 Scenarios

A storm ruined the harvest Arin is a farmer who mainly cultivates potatoes, onions and tomatoes in the fields of his family. The harvest of March was really good, and Arin hoped that the same volume of production might have been repeated in April as well. Unfortunately, the North Indian Ocean cyclone season starts in April, and in the second week of this month Tauktae storm, the strongest storm of of the season, violently hit Arin's lands, halving the crop.
Consequently, Arin wants to share in the system (DREAM) the outcome of the storm. Arin opens its browser and access the system using its credentials. Next, he selects from a menu in the incoming webpage the entry for inserting data

[1] "A narrative description of what people do and experience as they try to make use of computer systems and applications" [M. Carrol, Scenario-based Design, Wiley, 1995]

about production and problems. Then he fills the form with a description about the storm and what it caused to its fields, and finally he sends it.
Arin hopes to receive some aid.

Great year for the potatoes Like many other farmers in India, Ikbal knows that the best period for cultivating potatoes is during the months from October to December. The weather is neither hot nor cold and the monsoons are nearly over at this time. Therefore, last October Ikbal planted many potatoes, and in February, with the help of some peasants who work for him, has picked up all of them.
In order to publish in `DREAM` the results of this year potatoes' harvest, he access the system through its computer, opens the form for inserting data about the production and then starts to fill it. The form is quite structured, and requires some data for each entry. After having inserted all the objective information about the crop, Ikbal would like to share something he has found out during this year, namely that planting the potatoes at depth x allows to retrieve substantial crops of this vegetable. Thus, Ikbal fills in also the optional text box and then sends it.

Registering to the system as an agronomist Dalbir worked for many years as an agronomist in Bihar, but recently he has been moved to Telengana. Telengana uses this innovative system, `DREAM`, to keep track of the harvests of farmers, to collect interesting data from sensors and many other things. All the agronomists in Telengana must work with `DREAM`, therefore Dalbir has to create an account.
Dalbir opens the webpage of `DREAM` and initiates the procedure for creating a new account. He follows the procedure tailored for agronomists, which allows him to create a new username and password. Next, Dalbir has to insert some data about his job in order to finish the registration. Among the others, Dalbir is asked to insert the area where he works.
His data will be verified later.

Visiting a farm The first task of Dalbir is to visit the farm of Arin. The date and hour of the meeting has been established by `DREAM`, which has notified also Arin about the meeting. Arin has been chosen because last month he performed very bad.
During the meeting, Arin has the opportunity of explaining all the problems he faced during the past month, one for all Tauktae (the strongest storm of the North Indian Ocean cyclone season). Arin takes Dalbir to the fields, where Dalbir can gather some sample from the terrain. At the end of the meeting, Dalbir gives Arin some advices, and then he moves to another farm.
Once in the office, Dalbir inserts in the system a report about his visit to Arin's farm.

A bad schedule Like every evening, Dalbir accesses `DREAM` for downloading the daily plan of the next day. Every daily plan provided by the system contains time and place of every meeting, in addition to some information about the farmer. Dalbir notices that Ikbal's farm is closer to his parents' house, and Dalbir knows that they would be really happy to if he had lunch with them.
However, the meeting with Ikbal is scheduled for the 3p.m o' clock. Therefore, Dalbir decides to exchange the meetings with Ikbal and Tarak, visiting Ikbal at 11.30a.m. and Tarak at 3p.m.. Through the user interface of `DREAM`, Dalbir exhanges the two meetings and once he has confirmed, `DREAM` notifies the farmers of the change.

Collecting data(?) `DREAM` is a software system designed to provide various kinds of analytics to its users, in particular to the policy makers. As a matter of fact, policy makers want to distinguish between well- and bad- performing farmers, as well as understand the impact of initiatives carried out by the agronomists and find patterns and relations among the data.
Therefore, periodically, `DREAM` interacts with sensors deployed on the territory in order to retrieve the humidity of the soil, with the water irrigation system to retrieve the amount of water used by each farmer and with an already existing website which provides up-to-date weather forecasts. In particular:

- 3 times a day the soil sensors are asked to provide data about the humidity of the soil;
- once a day the water irrigation system is asked to provide the water used by each farmer (when the data is available);
- the weather forecasts website is accessed monthly to compute some analytics on the data, but also on demand when agronomists or farmers ask for weather forecasts.

3.2.2 Use cases

USE CASE 1	
Name	InsertProblems
Actor	Farmer
Entry condition	The farmer faces a problem and decides to communicate it on `DREAM`
Event flow	1. The farmer accesses the system through its username and password; 2. in the homepage, the farmer selects from the menu the item ”insert problem”; 3. the farmer selects the kind of problem, when it occurred and how much it lasted; 4. the farmer gives a brief description of the problem; 5. the farmer selects whether he likes to meet as soon as possible an agronomist; 6. the farmer clicks on the ”send” button; 7. `DREAM` automatically detects where is the farm from the account of the farmer; 8. `DREAM` stores the data and eventually takes into account the request of a meeting with an agronomist.
Exit condition	The problem is saved in the `DREAM` database and, if requested, an agronomist is scheduled to meet the farmer in the next days. The farmer is notified about the meeting.
Exceptions	• If the kind of problem is not present, the farmer can add it by its own. • In case no agronomist can be scheduled for the current week, the farmer is notified about the problem and an agronomist will be found for the next week.
Special requirements	• The availability of an agronomist is checked in less than 3 second; • the page is refreshed in less than 1 second.

USE CASE 2	
Name	InsertProductions
Actor	Farmer
Entry condition	The farmer gets the results of a crop
Event flow	1. The farmer accesses the system through its username and password; 2. in the homepage, the farmer selects from the menu the item ”insert productions”; 3. the farmer inserts the picked product, when it was planted, the volume planted, the volume picked; 4. the farmer optionally gives a brief description of the harvest or an observation he made; 5. the farmer clicks on the ”send” button; 6. `DREAM` automatically detects where is the farm from the account of the farmer; 7. `DREAM` stores the data.
Exit condition	The data is saved in the `DREAM` database and the farmer is notified.
Exceptions	• If the kind of product is not present, the farmer can add it by its own.
Special requirements	• the page is refreshed in less than 1 second.

USE CASE 3	
Name	RegisterAgronomist
Actor	Agronomist
Entry condition	The agronomist does not have an account.
Event flow	1. The agronomist goes to the `DREAM` homepage; 2. the agronomist selects "create account", then "create agronomist account"; 3. the agronomist inserts his e-mail and devises a password; 4. the system verifies the e-mail of the agronomist; 5. the agronomist inserts personal data like name, surname, date of birth, specialization; 6. the agronomist selects the area where he will work; 7. the agronomist clicks on "done"; 8. `DREAM` verifies the identity of the agronomist; 9. the agronomist is notified that he can begin to work.
Exit condition	The agronomist's account is saved in the `DREAM` database and the identity of the agronomist has been verified.
Exceptions	• If the e-mail is already used, the agronomist is forced to change it; • If the password is too short, the agronomist is forced to change it; • if the identity of the agronomist cannot be verified, a mail notifying the error is sent to the agronomist.
Special requirements	• the identity must be verified within 3 days.

USE CASE 4	
Name	RegisterFarmer
Actor	Farmer
Entry condition	The farmer does not have an account.
Event flow	1. The farmer goes to the `DREAM` homepage; 2. the farmer selects "create account", then "create farmer account"; 3. the farmer inserts his e-mail and devises a password; 4. the system verifies the e-mail of the farmer; 5. the farmer inserts personal data like name, surname, date of birth, location of the farm; 6. the farmer clicks on "done"; 7. `DREAM` verifies the existence of the farm; 8. the farmer is notified that the account has been created.
Exit condition	The farmer's account is saved in the `DREAM` database and the existence of the farm has been verified.
Exceptions	• If the e-mail is already used, the farmer is forced to change it; • If the password is too short, the farmer is forced to change it; • if the existence of the farm cannot be verified, a mail notifying the error is sent to the farmer.
Special requirements	• the existence of the farm must be verified within 1 day.

USE CASE 5	
Name	VisitFarm
Actor	Agronomist, Farmer
Entry condition	The agronomist is scheduled to visit a farm. The farmer and the agronomist have been informed of the meeting. The time of the meeting occurs.
Event flow	1. The agronomist is scheduled to visit a certain farm at a certain time; 2. the farmer is notified about the meeting; 3. the agronomist accesses `DREAM`; 4. the agronomist selects "insert farm data"; 5. the agronomist selects the farm and inserts data with a comment; 6. the agronomist clicks on "done"; 7. `DREAM` stores the data; 8. the farm is removed from the to-visit farms.
Exit condition	The data about the meeting have been inserted by the agronomist and stored in `DREAM` and the farm is not in the to-visit farms.
Exceptions	• If the farmer does not show up at the meeting, the agronomist specifies so when inserting data about the meeting.
Special requirements	None

USE CASE 6	
Name	GenerateDailyPlan
Actor	Agronomist, Farmer
Entry condition	The next day is a working day and the agronomist accesses the system.
Event flow	1. The agronomist receives a daily plan for the following day; 2. if it is ok for the agronomist, he clicks on ”accept”, otherwise he clicks on ”change it”; 3. the agronomist selects some changes, then he clicks on ”accept”; 4. `DREAM` checks if the new daily plan is acceptable; if not, an error is notified to the agronomist, who has to re-accept (or re-change) the schedule until `DREAM` accepts it; 5. the farmer is notified about the meeting.
Exit condition	The daily plan has been confirmed and the farmer has been notified.
Exceptions	• In case the agronomist does not accept the daily plan, the farmers are not notified about the meeting.
Special requirements	• Every farmer must be visited at least twice a year; • bad performing farmers must be visited more times than the others.

USE CASE 7	
Name	GetSoilSensorsData
Actor	SoilSensors
Entry condition	A timer has expired.
Event flow	1. `DREAM` asks the soil sensors for some data; 2. the soil sensors answer with the data; 3. `DREAM` stores the data in its database.
Exit condition	`DREAM` has stored in its database the data coming from the sensors.
Exceptions	• In case a sensor is not reachable, `DREAM` stores that the data of that sensor at that time is not available because ”the sensor is not reachable”.
Special requirements	• The sensor must answer in less than 5 seconds; • `DREAM` must try to contact the sensor 3 times before giving up.

USE CASE 8	
Name	GetWaterIrrigationSystemData
Actor	WaterIrrigationSystem
Entry condition	A timer has expired.
Event flow	1. `DREAM` asks the water irrigation system for some data; 2. the water irrigation system answers with the data; 3. `DREAM` stores the data in its database.
Exit condition	`DREAM` has stored in its database the data coming from the water irrigation system.
Exceptions	• In case the water irrigation system is not reachable, `DREAM` stores that the data of the water irrigation system at that time is not available because "the water irrigation system is not reachable".
Special requirements	• The water irrigation system must answer in less than 5 seconds; • `DREAM` must try to contact the sensor 3 times before giving up.

USE CASE 9	
Name	VisualizeAggregateData
Actor	PolicyMaker
Entry condition	A policy maker accesses the platform and enters the *Aggregate Data* section.
Event flow	1. The policy maker accesses the system; 2. the policy maker selects the *Aggregate Data* menu item; 3. the policy maker chooses the date range they wish to analyze; 4. `DREAM` generates aggregate statistics for all areas in the database; 5. `DREAM` shows the data to the user.
Exit condition	The user finishes analyzing the data.
Exceptions	• If the system has not yet collected any data for the specified period, a warning is returned.
Special requirements	• The data generation must be completed in less than 3 seconds.

USE CASE 10	
Name	VisualizeBestFarmers
Actor	Agronomist
Entry condition	An agronomist accesses the platform and enters the *Farms* section.
Event flow	1. The agronomist accesses the system; 2. the agronomist selects the *Farms* menu item; 3. optionally, the agronomist chooses how the farms should be ranked (e.g. most productive, most resilient to bad weather, ...); 4. `DREAM` shows a list of the best farms ordered according to the selected parameter.
Exit condition	The user finishes analyzing the data.
Exceptions	• If the system has not yet collected any data in the agronomist's area, a warning is shown.
Special requirements	• The data must be shown in less than 1 second.

USE CASE 11	
Name	VisualizeForecasts
Actor	Agronomist
Entry condition	An agronomist accesses the platform and enters the *Weather Forecasts* section.
Event flow	1. The agronomist accesses the system; 2. the agronomist selects the *Weather Forecasts* menu item; 3. the agronomist chooses the date for which forecasts should be shown; 4. `DREAM` connects to the Telengana government website to fetch forecasts for the chosen date; 5. `DREAM` shows a map with the forecast.
Exit condition	The user finishes analyzing the data.
Exceptions	• If the Telengana government website is currently unreachable, an error is shown to the user. • If no weather data is available for the chosen date, an error is shown to the user.
Special requirements	• The Telengana government website must serve requests in less than 5 seconds; • The Telengana government website is assumed not to change in a backwards-incompatible manner without notice.

USE CASE 12	
Name	VisualizeFarmerForecastsAndSuggestions
Actor	Farmer
Entry condition	A farmer accesses the platform.
Event flow	1. The farmer accesses the system; 2. `DREAM` connects to the Telengana government website to fetch forecasts for the following days; 3. `DREAM` automatically shows forecasts and suggestions for the farmer.
Exit condition	The user logs out.
Exceptions	• If the Telengana government website is currently unreachable, an error is shown to the user. • If there are no suggestions for the farmer, a notice is shown.
Special requirements	• The Telengana government website must serve requests in less than 5 seconds; • The Telengana government website is assumed not to change in a backwards-incompatible manner without notice; • If an issue occurs with the weather forecasts, the suggestion section should not be impacted.

USE CASE 13	
Name	GetWeatherForecasts
Actor	OuterWeatherSource
Entry condition	Either (a) a user requests weather data; or (b) `DREAM` needs to access weather information for statistical purposes.
Event flow	1. `DREAM` connects to the Telengana government website to fetch the required forecasts; 2. The data is stored and analyzed as needed.
Exit condition	The weather forecast data is collected by the system.
Exceptions	• If the Telengana government website is currently unreachable, the process is aborted and an error is returned.
Special requirements	• The Telengana government website must serve requests in less than 5 seconds; • The Telengana government website is assumed not to change in a backwards-incompatible manner without notice.

3.3 Performance Requirements

3.4 Design Constraints

3.5 Software System Attributes

4 Formal analysis using Alloy

5 Effort spent

6 References

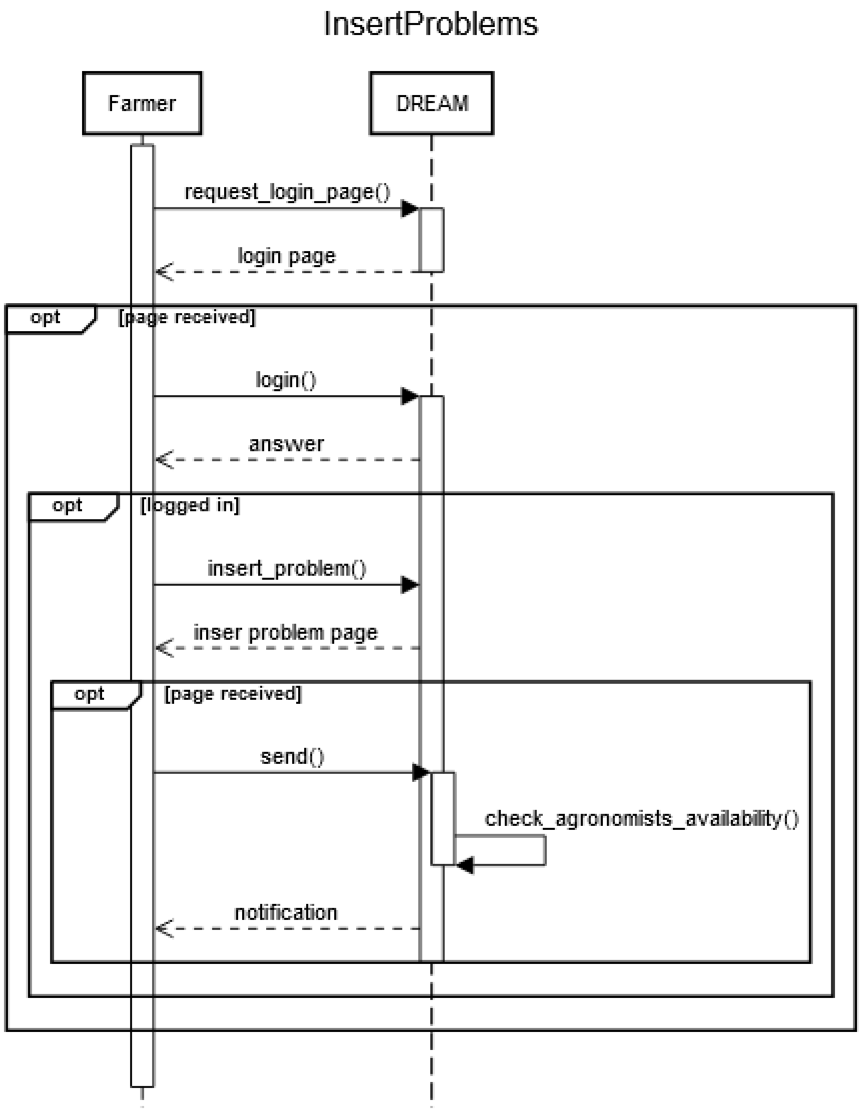

Figure 1: Sequence diagram for the InsertProblems use case

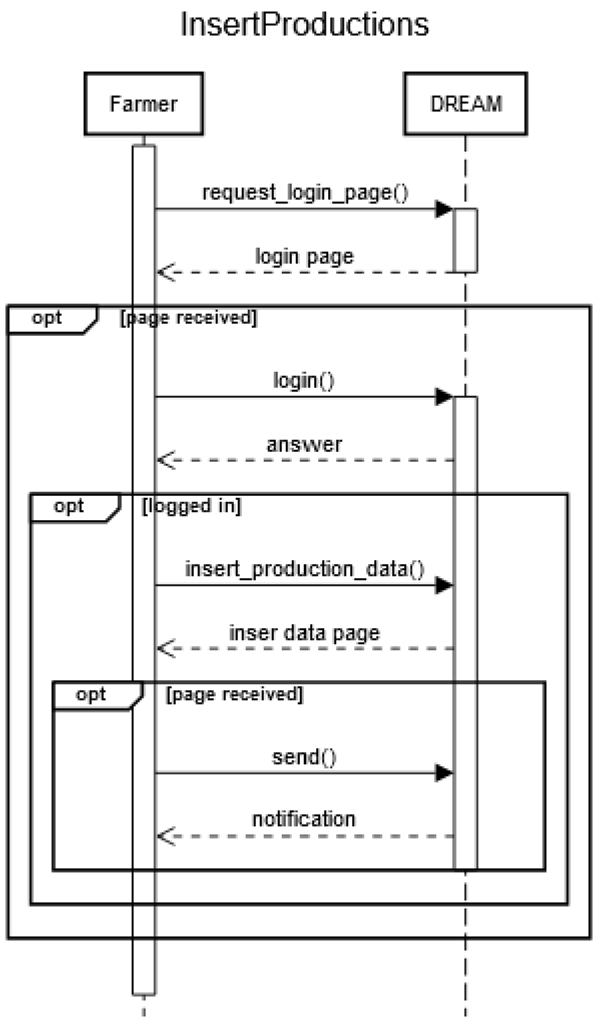

Figure 2: Sequence diagram for the InsertProductions use case

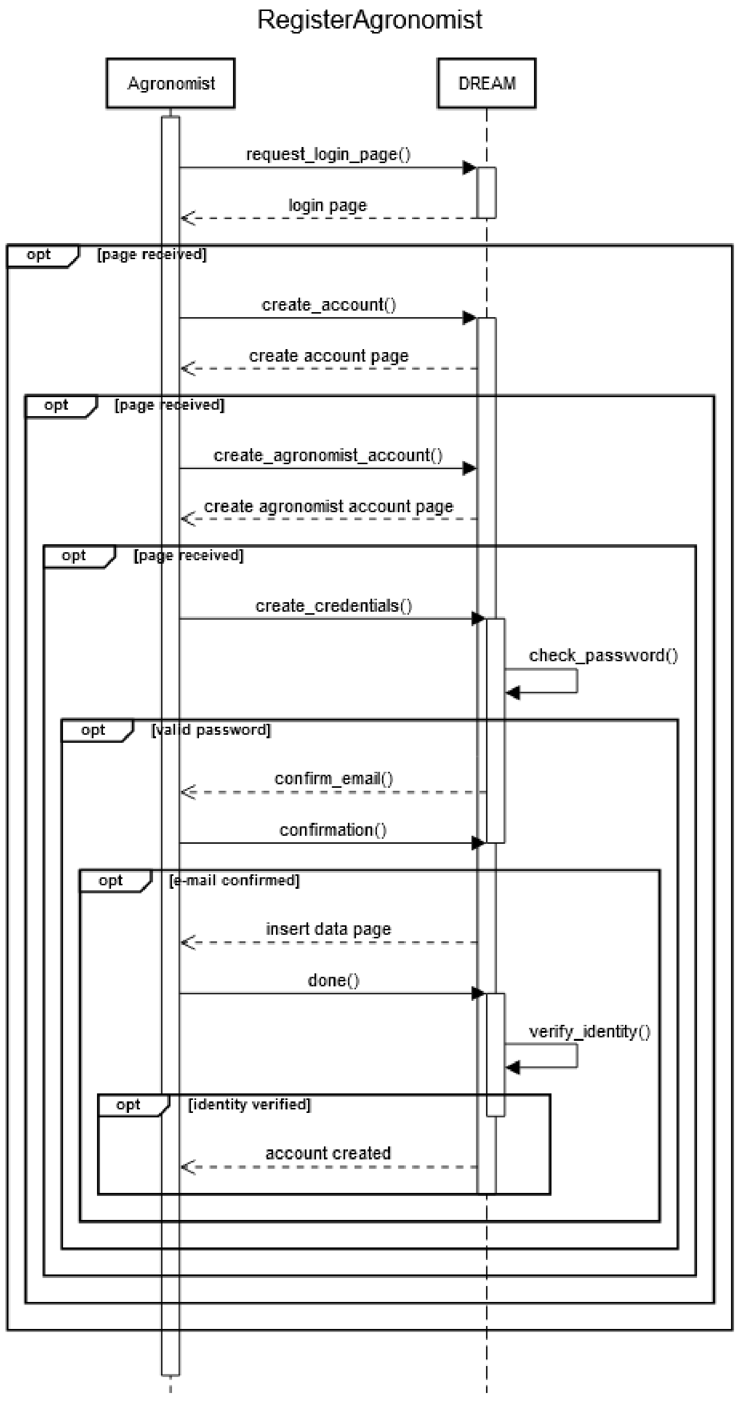

Figure 3: Sequence diagram for the RegisterAgronomist use case

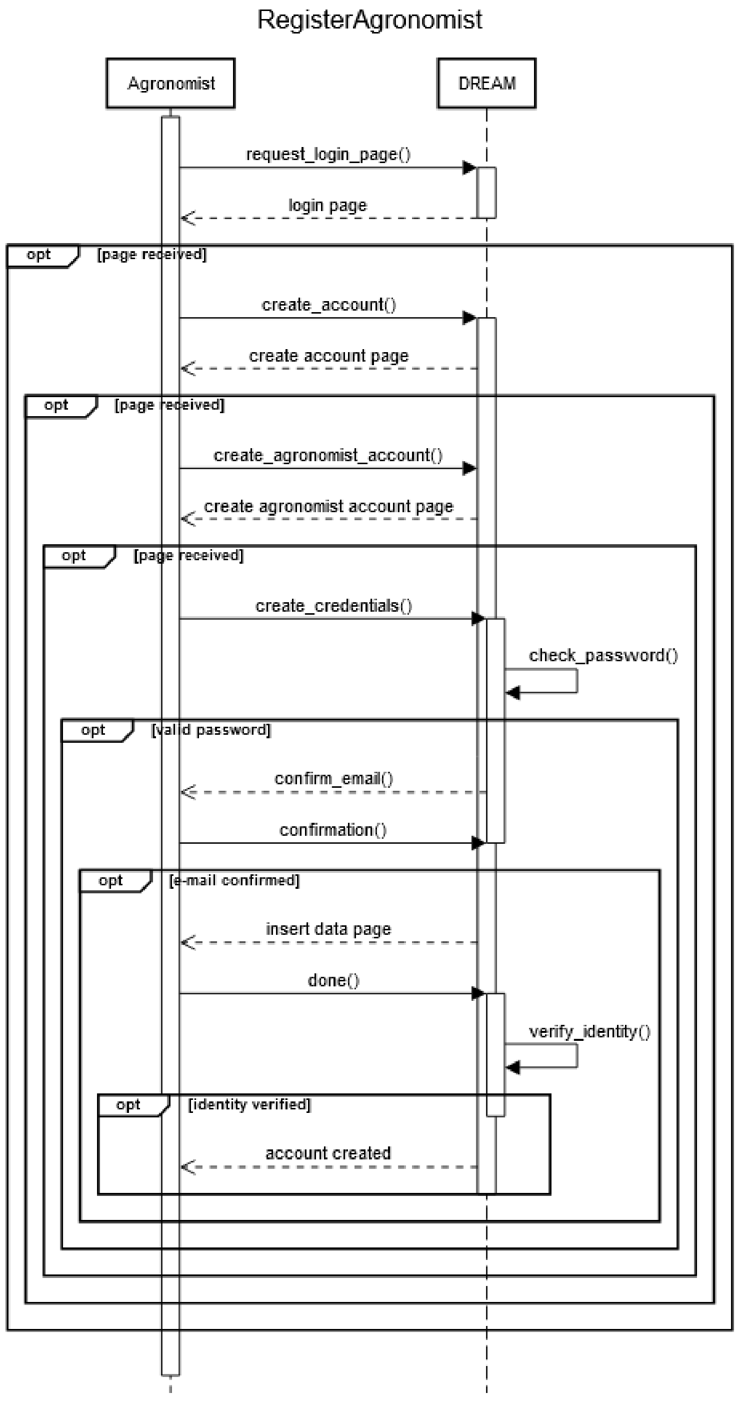

Figure 4: Sequence diagram for the RegisterFarmer use case

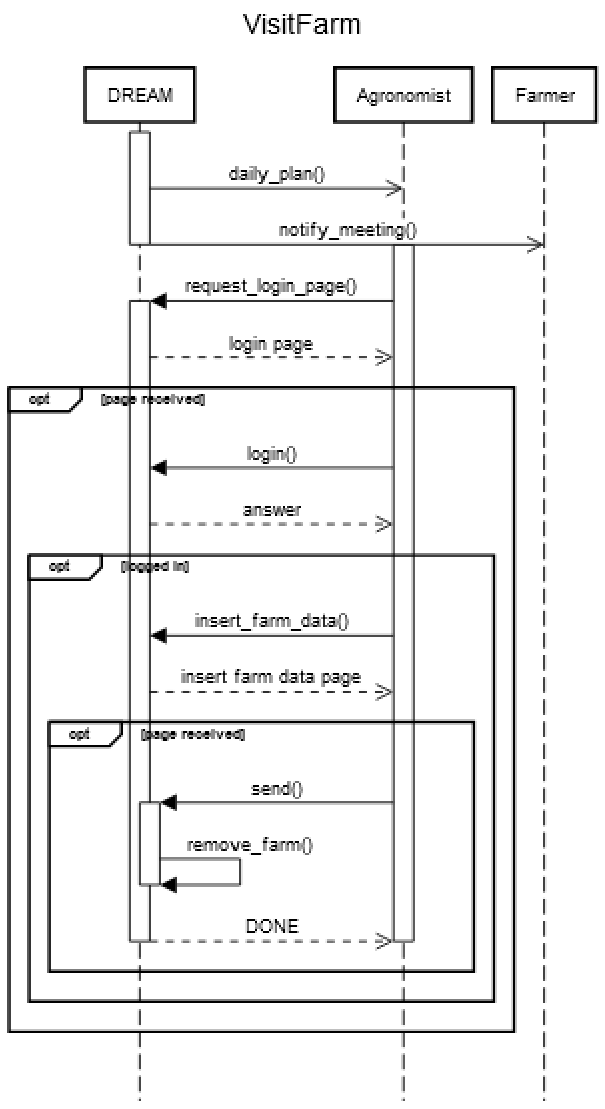

Figure 5: Sequence diagram for the VisitFarm use case

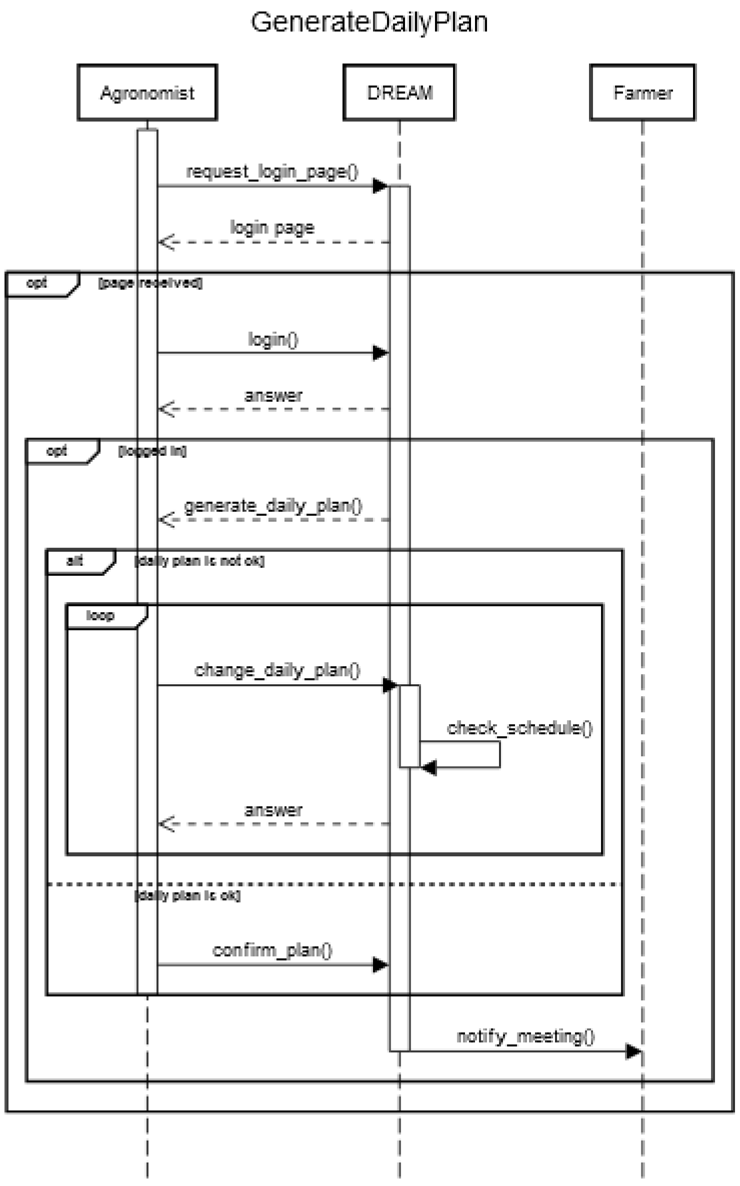

Figure 6: Sequence diagram for the GenerateDailyPlan use case

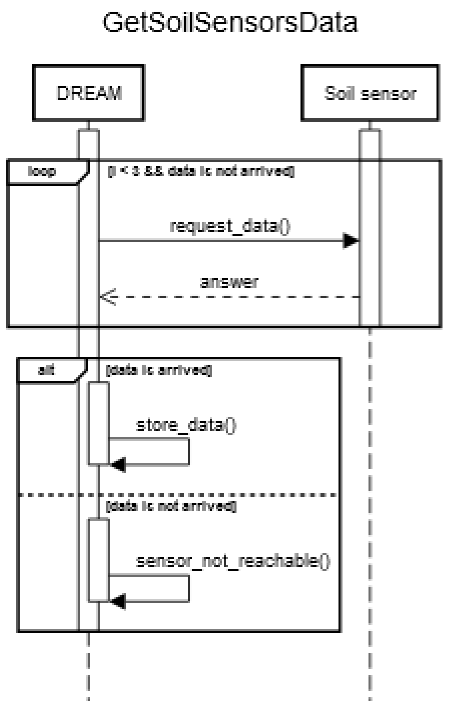

Figure 7: Sequence diagram for the GetSoilSensorsData use case

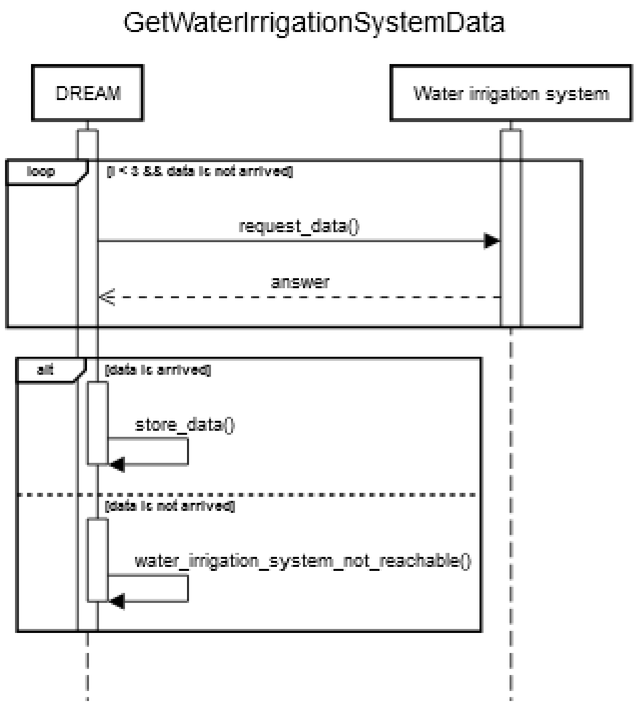

Figure 8: Sequence diagram for the GetWaterIrrigationSystemData use case

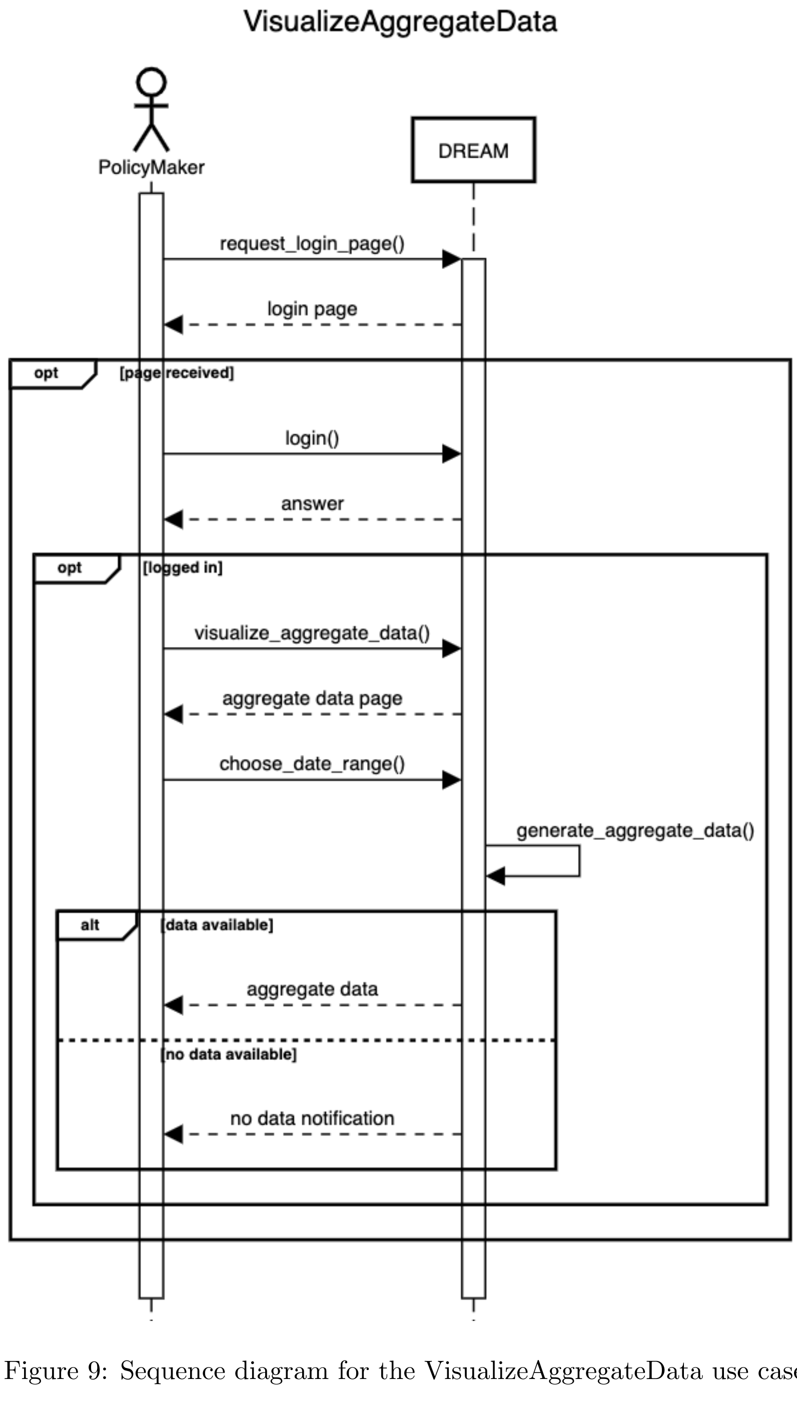

Figure 9: Sequence diagram for the VisualizeAggregateData use case

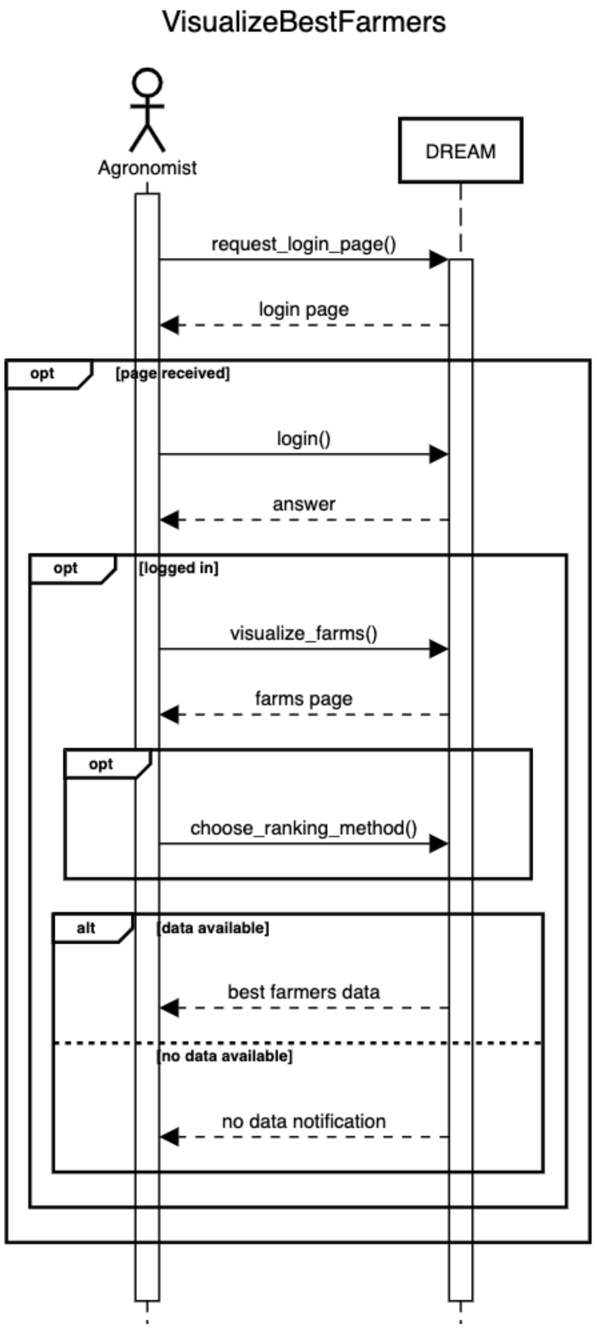

Figure 10: Sequence diagram for the VisualizeBestFarmers use case

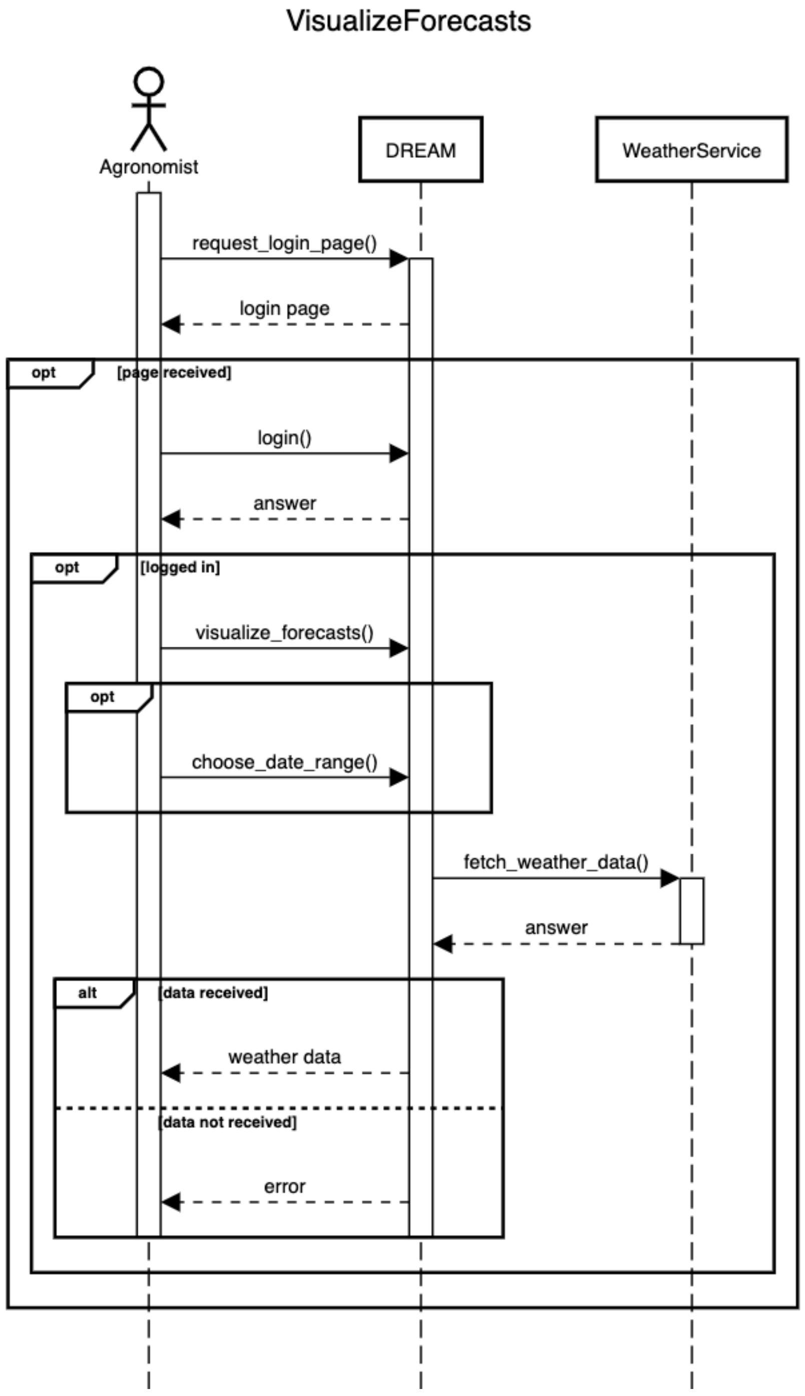

Figure 11: Sequence diagram for the VisualizeForecasts use case