RASD

December 23, 2021

Contents

1 Introduction

1.1 Purpose

Recently, the importance and necessity of resilient food systems have become evident with the large disturbances caused by the Covid-19 pandemic. This however has always been the case as the continuing threat of climate change to the agriculture sector demands a change in how we produce food.

With this in mind, the state of Telengana aims at developing a new system in order to aid an anticipatory, data driven governance model. It will combine information from several sources, such as water irrigation systems and the farmers themselves, in order to track the performance of individual farmers and analyze the impact of steering initiatives. The new system, DREAM, will moreover act as a platform where farmers and agronomists can communicate and share their knowledge.

This document will further expand on the goals and requirements put on the system to be with the purpose of guiding the development.

1.1.1 Goals

Goal	Description
G1	Allow Policy Makers to identify the performance of farmers
G2	Allow Policy makers to track the results of steering initiatives
G3	Allow Farmers to create and participate in discussion forums
G4	Allow Farmers to request assistance from other farmers or agronomists
G5	Allow Farmers to view data relevant to them, e.g. weather forecast or crop suggestions
G6	Allow Farmers to provide data about their production and problems.

1.2 Scope

While there are several stakeholders to consider, this document only concerns itself with that of policy makers and farmers. Occasionally agronomists are mentioned as well but they should not be considered part of the project scope and are only mentioned to clarify or to help in understanding.

Furthermore, the system only concerns itself with farmers of Telengana and policy makers of the same state. This might mean that certain decisions, or requirements, might be inapplicable for other geographical areas.

1.2.1 World Phenomena

Identifier	Description
WP1	Farmer harvesting
WP2	Policy maker decides on steering initiatives
WP3	Incidents at farm (e.g. pests)

1.2.2 Shared Phenomena

Identifier	Description
SP1	Farmer inserts data about their production
SP2	Farmer requests assistance
SP3	Farmer answering help request
SP4	Farmer requests to view personalized data
SP5	Farmer creates a discussion forum thread
SP6	Farmer posts comment in existing forum thread
SP7	Policy maker viewing performance of farmers
SP8	Farmer registering an account
SP9	System visualizes personalized data based on farmers location and production type
SP10	System sends notification to user
SP11	Farmer inserts description of problems faced
SP12	Policy maker requesting to view steering initiative

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

Definition	Description
Performance	Performance w.r.t to farmers is used as a general term that for example could mean productivity. The performance metric can be decided on later is the idea.
Farm Identifier	To identify a specific farm, this could be e.g. a plot number or an organization number.

1.3.2 Abbreviations

Abbreviation	Description
RASD	Requirements Analysis and Specification Document
WP	World Phenomena
SP	Shared Phenomena
GX	Goal number X
DX	Domain assumption number X
RX	Requirement number X
DREAM	Data-dRiven PrEdictive FArMing in Telengana

1.4 Revision History

1.5 Reference Documents

- The specification document "01. Assignment RDD AY 2021-2022.pdf"

1.6 Document Structure

This document is composed of six sections, detailed below.

In the first section the problem is introduced together with the associated goals of the project. Additionally the scope of the project is specified along with the various phenomena occurring. Lastly, the necessary information to read the report is presented, such as definitions and abbreviations.

Section two contains an overall description of the system, including a detailing of its users and main functions. Moreover there is the class diagram, descriptions of several scenarios, some statecharts and finally the domain assumptions made in this report.

In section three the requirements on the system is specified. This includes functional requirements, non-functional requirements and requirements on external interfaces. Furthermore use cases are described, with accompanying use case and sequence diagrams. Section three also contains mappings of functional requirements to the goals of the system, and to the use cases.

Section four contains a formal analysis with the help of Alloy. Together with the Alloy code, the analysis objective is described.

In section five there is a presentation of the project members total effort spent.

Section six contains the references used.

2 Overall description

2.1 Product perspective

2.1.1 Scenarios

1. **Farmer wants to start using the system**

   The farmer Bertil decides that he wants to register to the service to take advantage of the various offerings it provides, such as weather data and the discussion forum. He launches the service and selects the option to sign up. Bertil inputs all the relevant information and is granted access to the service.

2. **Farmer requesting help**

   The farmer Lars-Erik has troubles with his farm. His crop production has gone down and the measures he has taken have not had any effect. Lars-Erik realizes that he needs help and decides to use the DREAM platform. So he launches the service, to which he is already registered and logged in, and selects 'Need help'. Lars-Erik enters the information about the type of problem he is facing and the measures he already has tried without success. Lars-Erik finally enters a title and confirms. Lars-Erik receives a success message and the help request is posted in the list of all the other help requests.

   Stefan, another farmer, who really likes helping other people notices that a new help request has been posted and after reading it, he decides he wants to help. Stefan sends a response, with a suggestion of what worked for him when he was in a similar situation a year ago. Lars-Erik reads the response and asks some follow up questions to Stefan. After Stefan answers, Lars-Erik sends a final thank you and informs him that he will try his suggestion the next day.

3. **Farmer wants to discuss topic on the forum**

   After a long working day the farmer Johnny sits down at his computer and logs onto the DREAM website. Johnny is interested in discussing with other farmers about a new planting method he tried today. So, Johnny selects the option to create a new forum post. The system asks Johnny what the title of the post should be, and what should be in the body of the post. Johnny takes a couple of minutes to think and then fills in the requested information and submits. He can now see his post among the others, and so now he is excitedly waiting for someone to respond.

4. **Farmer viewing personalized data**

   The farmer Sonny wakes up early in the morning and goes to sit down at the kitchen table. While eating his breakfast, he opens up his laptop and

opens the DREAM website. He logs in, and selects the option to view personalized data. The website redirects to a new page and displays the weather forecast of today, the current soil humidity at his farm as well as suggestions for crop and fertilizer to use. Based on this information, Sonny makes a plan for the day, e.g. deciding on how much water to dispense and where.

5. **Farmer inserting performance data about production**

   After a day of hard work the farmer Örjan does the usual thing he does monthly, which is to file a report to the DREAM website. He launches it, selects the current day and an option to insert data. He inserts information about the amount of crops he harvested. This month Örjan sadly had some problems at the farm, parts of the crops had been damaged by pests. He included this when filing the report.

6. **Policy Maker evaluating performance of farmers**

   Sven, the Telengana policy maker, opens up the DREAM website and logs in with his administrative credentials. Lately the weather has been rough in Telengana and Sven wants to know whether or not this has impacted the state's farmers. So, he selects the option of viewing farmer performance on the website. The system displays the default view with a paginated list of all the state's farmers, ranked by their productivity in the last 60 days. As Sven is interested in the overall performance difference, he selects the aggregate option. The system displays a graph of the average performance of the state's farmers over the last 60 days.

7. **Policy Maker evaluating steering initiative**

   The policy maker Holger wants to analyze the effectiveness of a certain steering initiative offered by the agronomists. Holger launches the system and presses the button "steering initiatives". He is presented with a list of all steering initiatives. After selecting an initiative, the website shows which farmers adopted the initiative. As previously mentioned, Holger wants to view the effectiveness of the initiative, this is presented by a performance graph, with a marker showing the adoption of the initiative. Holger can select between a performance graph for a single selected farmer, or an aggregated graph of all.

2.1.2 Class diagram

This is the class diagram of the system. It should be noted that associations with the User class is dependent on the role. A user can be responsible or associated to a farm if and only if the user is a farmer. The same is true for the relation with ForumThread, Comment and HelpRequest. On the other hand, the relation with SteeringInitiatve requires that the user has the role of

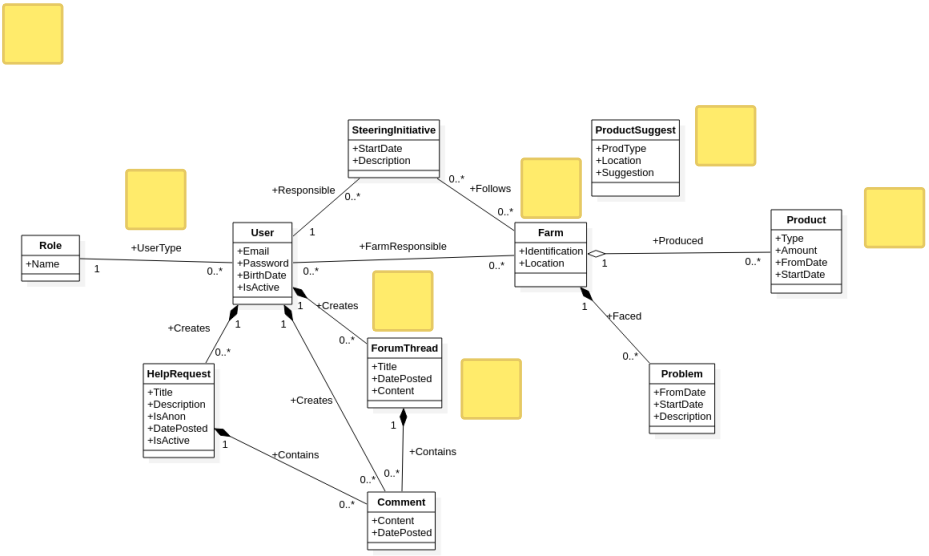

Figure 1: Class Diagram of DREAM. ProductSuggest is standalone, and is assumed to be managed by agronomists.

Policy maker (or Agronomist). ProductSuggest will be used to find personalized suggestions of crops and fertilizers. The idea is the farmers product history and location is used to personalize the suggestion. The suggestions (based on location + product type) are inserted by experts (i.e. agronomists).

2.1.3 Statecharts

In this section statecharts of some aspects of the system is presented. Those left out were so mainly due to their simplicity.

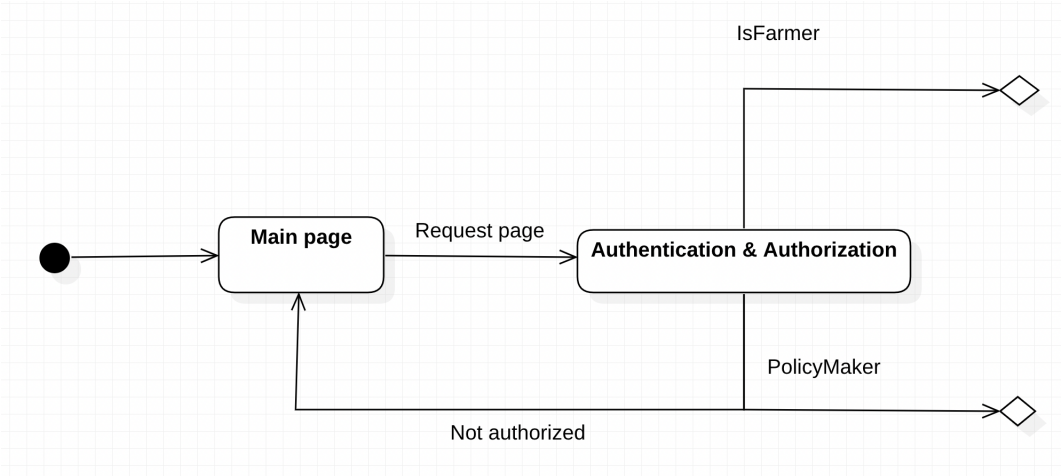

Figure 2: Overview state diagram. This diagram is meant to capture the general state flow of the system. The idea is that after a page has been requested, the user is first authorized and then depending on their role there are different possibilities of what page it was they requested (and the diagram continues). If they are not authorized the state will go back to the main page.

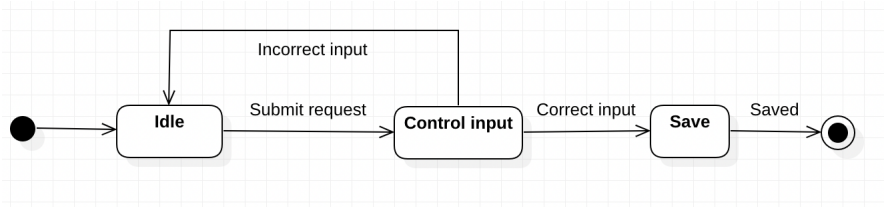

Figure 3: State Diagram of creating a help request

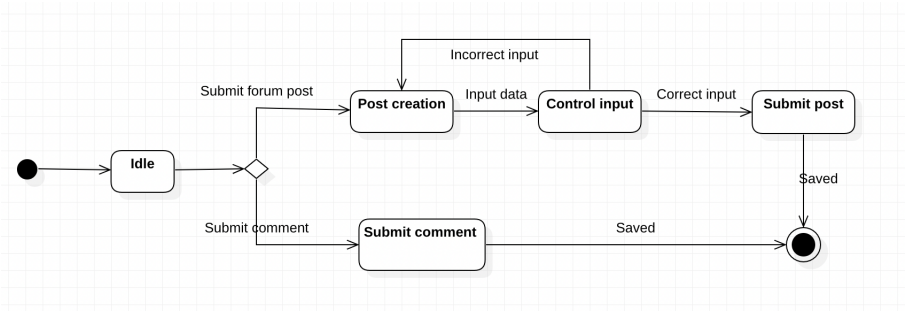

Figure 4: State Diagram of the forum

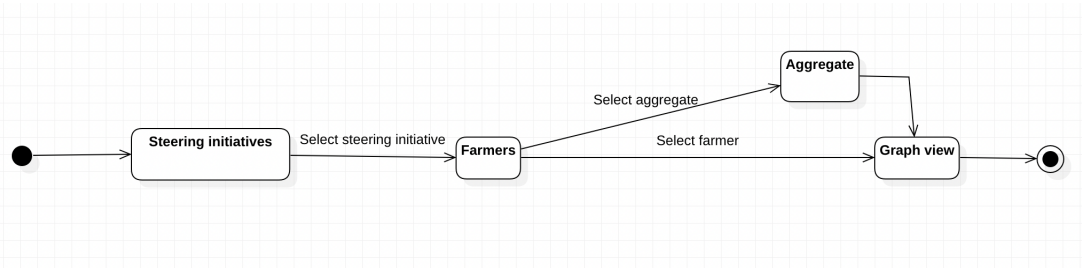

Figure 5: State Diagram of steering initiative

2.2 Product functions

In this section the main functionalities of DREAM is presented and described in more detail.

2.2.1 Identification of well and poor performing farmers

One of the most important aspects of the system is the ability of identifying farmers that are outliers in terms of them performing particularly well or poorly with respect to other farmers. To achieve this the system will, on request, calculate the performance of every farmer and display the results in an ordered list. Furthermore the option to aggregate the performance is available, e.g. the average productivity of all farmers in some area, in order to investigate more broadly.

The performance of a farmer could be calculated with different aspects in mind. Measuring performance solely focusing on productivity (i.e. amount of crops produced) is one way. Another way is to also factor in the farmer's water usage per produced amount for example.

This functionality is only available to Policy Makers, and is reliant on several data points that are either inserted by the farmers themselves, or collected by different sensors. This collection of data is of course one of DREAM's most important functions as well.

2.2.2 Knowledge sharing and communication among farmers

The system will provide two tools in order to facilitate knowledge sharing and communication among farmers. The first tool is the discussion forum, where any registered farmer can create new forum posts and participate in existing

ones by making comments. From the start the forum will be moderated by policy makers, and as time goes on responsible farmers and agronomists will also be added as moderators.

The second tool that is available is the help request function. With this tool farmers will be able to anonymously, unlike the forum, request and receive help from others. This will lower the threshold for those who may not be comfortable asking their questions publicly.

2.2.3 Tracking effect of Steering initiatives

An important aspect of the system is the ability to track the impact of a steering initiative on the performance of farmers. These steering initiatives are created by the experts, agronomists. The agronomists also manages which farms adopts which initiatives. It is the policy makers who are main responsible and will monitor the effectiveness of the steering initiatives.

Policy makers will have a list of all the active steering initiatives. After choosing one of them they will see which farms have adopted it and can from there view the performance trends of a single farmer or on an aggregated level.

2.3 User characteristics

The following three actors are considered in the DREAM system.

1. **Unregistered farmer**

   A farmer that needs to register to the DREAM platform before being able to use any of its functionalities.

2. **Policy maker**

   A registered user that can use the system to get an overview of the performance of farmers in the Telengana province and to evaluate steering initiatives.

3. **Farmer**

   A registered user who is associated with a farm and uses the system in order to e.g. improve their own production resilience. Can for example communicate with other farmers, send anonymous help requests, insert data about their production and visualize relevant data to them.

2.4 Assumptions, dependencies and constraints

2.4.1 Domain assumptions

Identifier	Description
D1	There exists APIs where correct soil moisture and water irrigation data can be retrieved for every farm in DREAM
D2	Each farm has a unique identifier, which can be used for external APIs
D3	There exists an API where a farmers connection to a farm can be verified, e.g. by a (birth date/fiscal, farm id) tuple
D4	Steering initiatives are created by and administered by agronomists (not considered in this project)
D5	Agronomists assign farms to steering initiatives
D6	Farmers follow the steering initiatives they are assigned to
D7	Farmers insert truthful data about their production
D8	Policy makers have access to an account on DREAM with privileges
D9	Farmers insert data about their production
D10	Crop and fertilizer suggestions are managed by an agronomist or policy maker.

3 Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

The user interface of DREAM is a website that will be used by both farmers and policy makers. It should be available to as many as possible, regardless of technical background. Therefore the website has to be easy to use. Additionally, to accommodate farmers (assuming policy makers have access to an office PC), DREAM has to be user-friendly on mobile devices.

3.1.2 Hardware Interfaces

The system revolves around being able to input and visualize data gathered from different actors and sources. Since the system is fully operable on the internet, the only hardware interface needed to access DREAM is a working device with a web browser, such as a computer or a mobile phone.

The system also relies on the use of different sensors deployed on farms to obtain data, such as measuring the humidity of soil. However, it is assumed that the sensors are managed externally and that an API is provided to retrieve the data.

3.1.3 Communication Interfaces

Many of the functionalities offered by the system relies on communication with other services. The communication is either of the type where DREAM retrieves data, or where DREAM queries the external service. This is in line with the goal of Telengana to combine information from several sources.

Specifically, there are four different interfaces that DREAM utilizes, possibly through Web APIs. Below we list the required function offered by each interface. Note that the description of each interface are assumptions made in this project.

- **Retrieval of data on water irrigation system**

  The interface is be able to respond with a history of water usage for a specific farm, based on for example a farm identification number and a time period.

- **Retrieval of data on soil humidity**

  Similar to the previous item, this interface provides a history of the soil humidity for a specific farm in some time period.

- **Authentication of farmer to a farm**

  During the registration process, DREAM requires that the farmer be paired with a farm. To ensure that only actual farmers of the state can register, we need to authenticate that the registering farmer in fact is associated to the farm that they specify. This interface provides that functionality based on a (farmer, farm) tuple.

- **Retrieval of weather forecast**

  Lastly, this interface is used by DREAM when visualizing personalized data for a farmer. In that data report a weather forecast will be provided to the farmer, and for that we interface with an external service.

3.2 Functional Requirements

Requirement	Description
R1	The system shall allow an unregistered farmer to register an account
R2	The system shall allow a registered farmer to insert data about their production
R3	The system shall allow a registered farmer to insert descriptions of problems they faced
R4	The system shall allow a registered farmer to request help
R5	The system shall allow a registered farmer to comment on help requests
R6	The system shall allow a registered farmer to create a forum thread
R7	The system shall allow a registered farmer to comment on forum threads
R8	The system shall allow a registered farmer to view data relevant to them
R9	The system shall allow a registered policy maker to view farmer performances
R10	The system shall allow a registered policy maker to view all steering initiatives
R11	The system shall allow a registered policy maker to view a farmer's performance before and after steering initiative
R12	The system shall allow a registered policy maker to view the aggregated performance of farmers before and after steering initiative
R13	When a farmer registers, the system must authenticate that the farmer is associated to the specified farm
R14	The system must allow a farmer that creates a help request to be anonymous
R15	The author of a help request must be allowed to mark it as done
R16	The author of a forum thread must be allowed to remove it
R17	The author of a thread comment must be allowed to remove it
R18	The system must allow the policy maker to select farmer performance metric
R19	The system must allow the policy maker to select the display order on farmer performance list
R20	The system must be able to notify user of exception
R21	The system must be able to notify user on successful action
R22	The system must store the history of farmers productivity
R23	The system must allow registered farmers to login
R24	The system must allow registered policy makers to login

3.2.1 Mapping on Goals

Goal	Domain assumption	Requirement
G1	D1, D2, D7, D8, D9	R2, R9, R18, R19, R20, R22, R24
G2	D1, D2, D4, D5, D6, D7, D8, D9	R2, R9, R10, R11, R12, R18, R19, R20, R22, R24
G3	D3	R1, R6, R7, R13, R16, R17, R20, R21, R23
G4	D3	R1, R4, R5, R13, R14, R15, R20, R21, R23
G5	D1, D2, D3, D7, D9, D10	R1, R2, R8, R13, R23
G6	D3	R1, R2, R3, R13, R20, R21, R23

3.2.2 Use cases

1. **Farmer registration**

Actor	Farmer
Entry conditions	The farmer does not have an account and is on the initial view of the system
Event flow	1. The farmer presses the "Register account" button 2. The farmer enters name, birth date, email address and password 3. The farmer inserts identification data and location for an associated farm 4. DREAM processes the information and displays a success message
Exit condition	An account is created
Exceptions	1. The farmer does not enter all mandatory data 2. Farm not found (i.e. incorrect identification) 3. Farmer is not associated to specified farm • In all cases DREAM will notify the user

2. **User Login to DREAM**

Actor	User (Farmer or Policy maker)
Entry conditions	User is registered, not logged in and on the DREAM main page
Event flow	1. User presses login button 2. User insert email address and password 3. User submits 4. DREAM processes the information and displays a success message
Exit condition	User is logged in to DREAM
Exceptions	1. User do not enter password or email before submitting 2. User enters invalid email/password combination • In all cases DREAM will notify the user

3. **Request help**

Actor	Farmer
Entry conditions	The farmer is registered, logged on to DREAM and on the initial view
Event flow	1. The farmer presses the Help requests button 2. The farmer selects the Request help option 3. The farmer enters a title and a description of the help request 4. The farmer then selects the option of being anonymous 5. DREAM processes the information and displays a success message
Exit condition	A new anonymous help request has been created
Exceptions	1. The farmer does not enter all mandatory data 2. The farmer enters invalid data for some field • If any of the above happens, DREAM notifies the farmer with an error message

4. **Answer help request**

Actor	Helper (Farmer or agronomist)
Entry conditions	The helper is registered, logged on to DREAM and on the initial view
Event flow	1. The helper presses the Help requests button 2. The helper selects one of the open requests from the list of requests 3. The helper writes a response message to the selected help request and submits 4. DREAM shows success message
Exit condition	An response has been added to a help request
Exceptions	1. The helper does not provide a response message before submitting • DREAM notifies with an error message

5. **Farmer create forum thread**

Actor	Farmer
Entry conditions	The farmer is registered, logged on to DREAM and on the initial view
Event flow	1. The farmer presses the "Forum" button. 2. Then the farmer selects "Create new thread" option 3. The farmer insert a title and a description for the post 4. The farmer presses submit button.
Exit condition	Forum thread is created
Exceptions	1. Description and/or title is missing 2. Farmer does not enter correct data for either field • If any of the above happens, DREAM notifies the farmer with an error message

6. **Farmer comment on forum thread**

Actor	Farmer
Entry conditions	The farmer is registered, logged on to DREAM and on the initial view
Event flow	1. The farmer presses the "Forum" button. 2. The farmer selects one of the open forum thread 3. The farmer inserts a comment 4. The farmer submits
Exit condition	A comment is posted to a forum thread
Exceptions	1. The farmer submits before inserting comment. 2. The farmer inserts invalid data in comment • DREAM notifies the farmer with an error message

7. **Farmer insert data about their production**

Actor	Farmer
Entry conditions	The farmer is registered, logged on to DREAM and on the initial view
Event flow	1. The farmer selects option to Insert Data on front page 2. The farmer selects a time period, selects some crops and specifies the produced quantity for each selected crop 3. Then the farmer writes a description of problems he faced in this period 4. DREAM saves the data and sends a success message back.
Exit condition	Productivity (number of crops produced) data is inserted into DREAM as well as a description of problems faced.
Exceptions	1. Farmer submits with invalid time period (e.g. in the future) 2. Farmer submits without adding harvest amount for all crops 3. Farmer submits without adding any crop *or* any problem description (i.e. one is enough) • If any of the above happens, DREAM notifies the farmer with an error message

8. **Farmer view personalized data**

Actor	Farmer
Entry conditions	The farmer is registered, logged on to DREAM and on the initial view
Event flow	1. The farmer selects option to "Show personalized data"
Exit condition	The farmer is redirected to a view where personalized data such as weather forecast and crop suggestions is visualized
Exceptions	1. None

9. **Policy maker view farmer performance**

Actor	Policy Maker
Entry conditions	Policy maker is logged in on an account with policy maker privileges and on the initial view
Event flow	1. The policy maker select option to view performance 2. The policy maker selects a time frame and that the list should be ranked by productivity 3. The policy maker then selects order by descending
Exit condition	A list of the states' farmers are shown, in order of productivity descending
Exceptions	1. Policy maker enters invalid time frame 2. No farmer has inserted data in the time frame • If any of the above happens, DREAM notifies

10. **Policy maker evaluate steering initiative of single farmer**

Actor	Policy Maker
Entry conditions	Policy maker is logged in on an account with policy maker privileges and on the initial view
Event flow	1. The policy maker selects option "Steering initiatives" 2. The policy maker selects a steering initiative from the list 3. The policy maker selects a farmer from the list of farmers displayed 4. The policy maker selects productivity as performance metric
Exit condition	A graph is shown of productivity of a farmer having adopted said steering initiative
Exceptions	1. Farmer selected has not inserted any production data 2. No farmers have adopted the steering initiative 3. No steering initiative in system • If any of the above happens, DREAM notifies

11. **Policy maker evaluate steering initiative aggregated**

Actor	Policy Maker
Entry conditions	Policy maker is logged in on an account with policy maker privileges and on the initial view
Event flow	1. The policy maker selects option "Steering initiatives" 2. The policy maker selects a steering initiative from the list 3. The policy maker selects the aggregated option 4. The policy maker selects productivity as performance metric
Exit condition	A graph is shown with the aggregated productivity of all farmers having adopted said steering initiative
Exceptions	1. None of the farmers have inserted production data 2. No farmers have adopted the steering initiative 3. No steering initiative in system • If any of the above happens, DREAM notifies

3.2.3 Use case diagrams

1. **Unregistered Farmer**

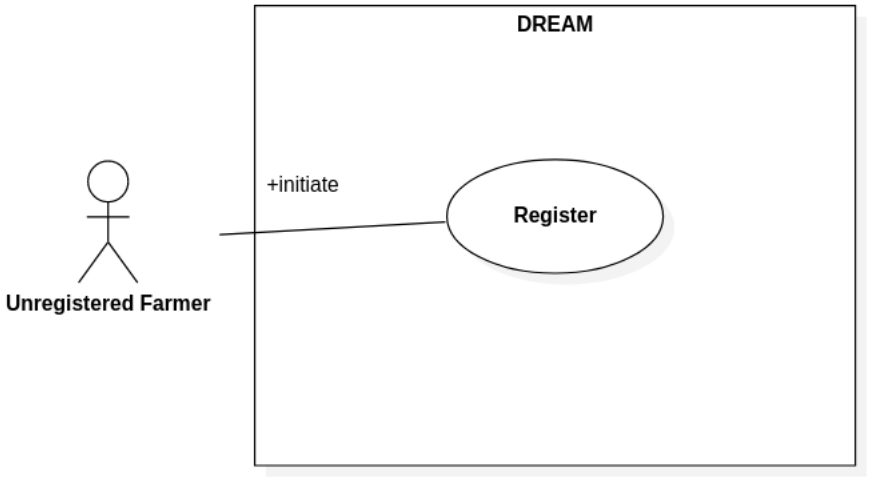

Figure 6: Use case diagram for an unregistered user

2. **Policy Maker**

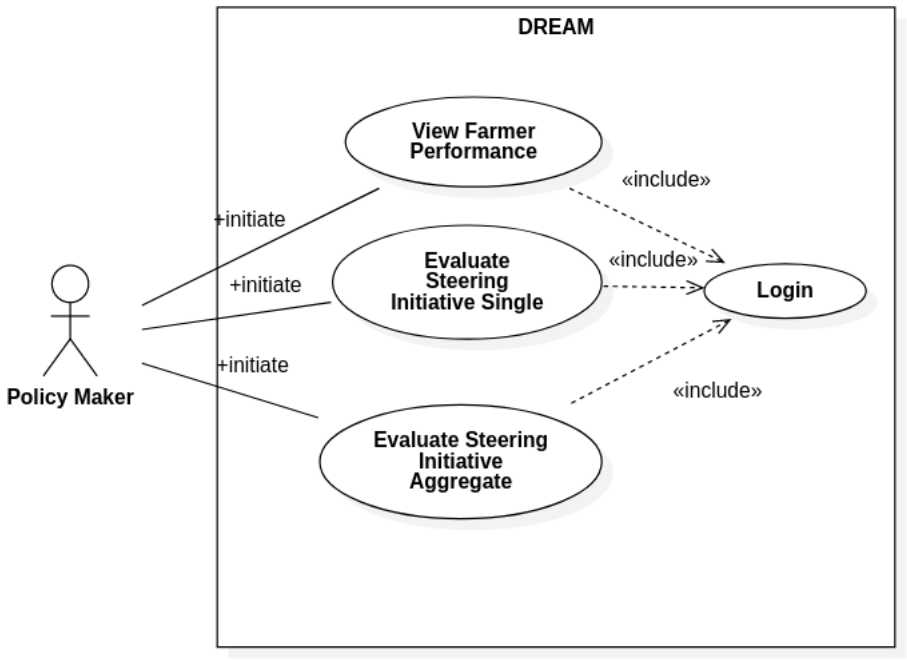

Figure 7: Use case diagram for a policy maker

3. **Farmer**

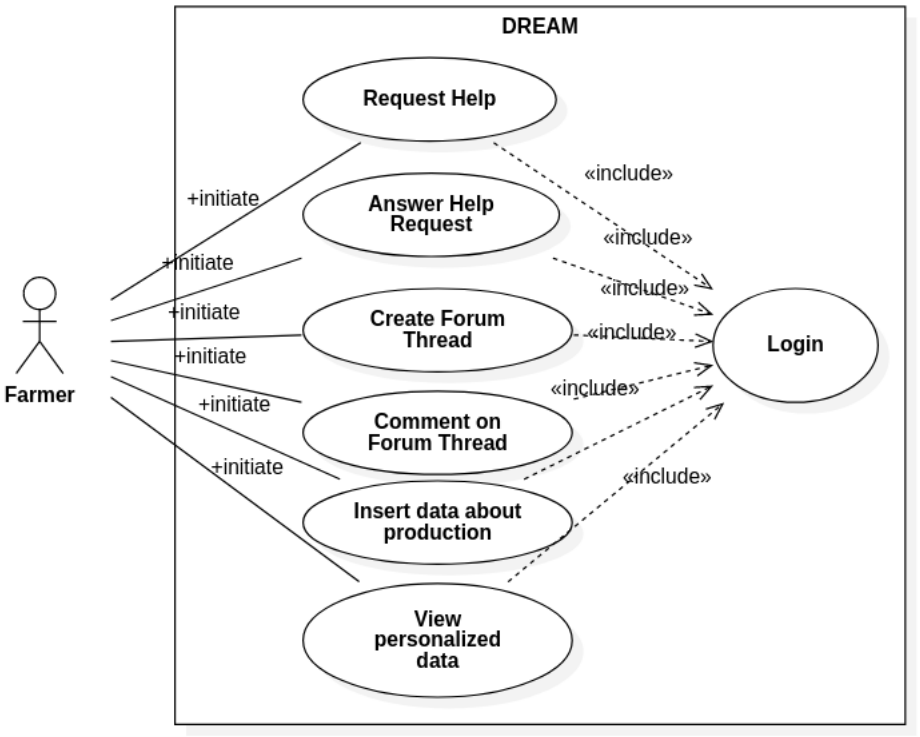

Figure 8: Use case diagram for a farmer

3.2.4 Sequence diagrams

In this section the corresponding sequence diagrams for the use cases are presented. Overall, what should be noted is that we consider the actor to be logged in all cases except for "Farmer registration" and "User login to DREAM".

1. **Farmer registration**

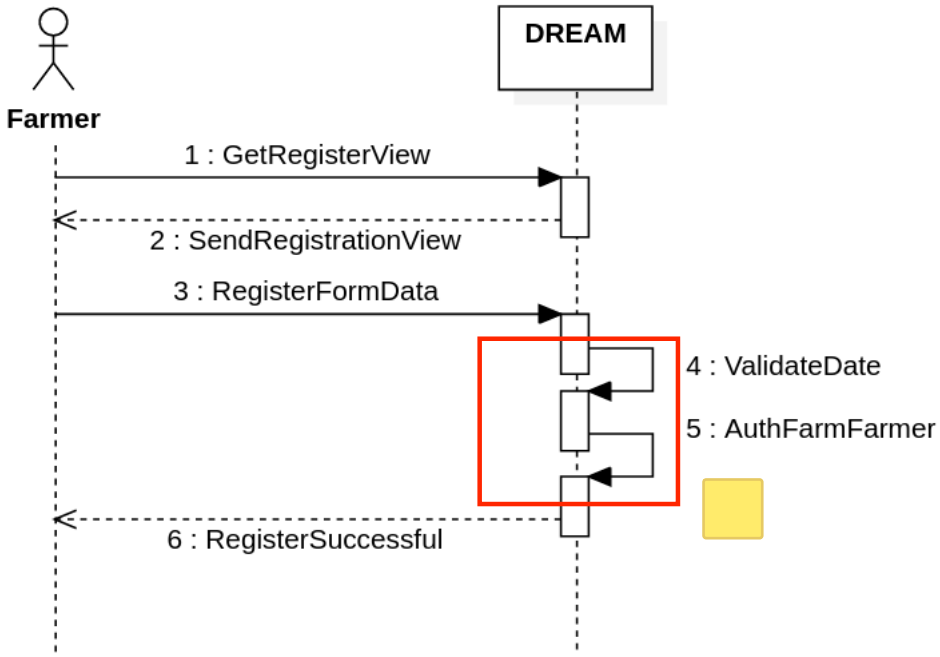

Figure 9: Sequence diagram for the registration process for a farmer. "AuthFarmFarmer" referring to authenticating that the farmer is associated to the specified farm.

2. **User Login to DREAM**

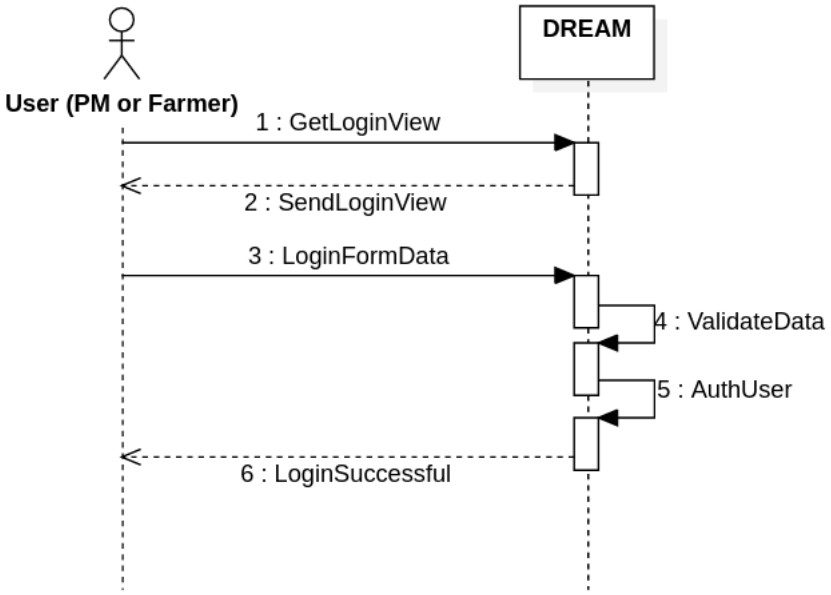

Figure 10: Sequence diagram for Login process

3. **Request help**

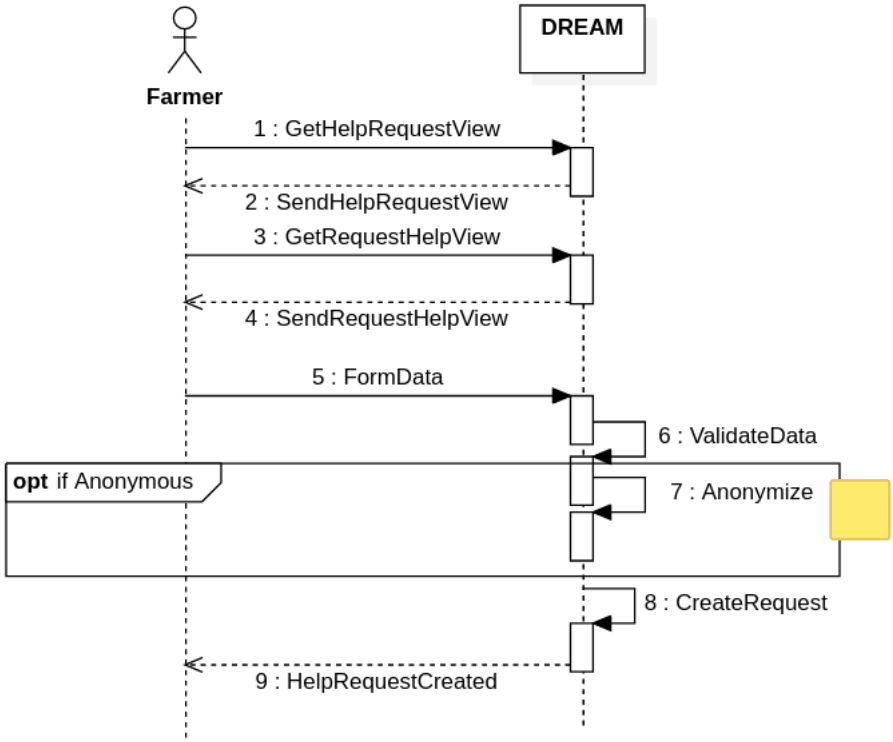

Figure 11: Sequence diagram for the process of when a farmer is creating a help request. The possibility of being anonymous is modelled in the diagram.

4. **Answer help request**

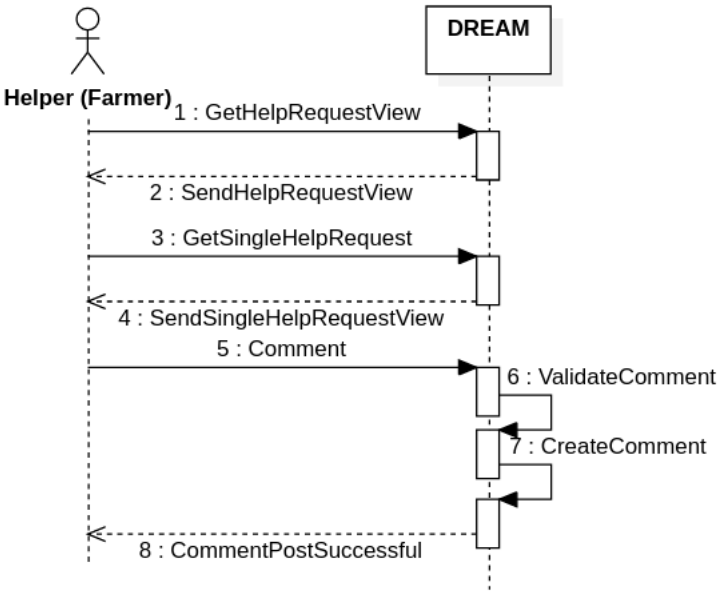

Figure 12: Sequence diagram for answering a help request. "Helper" used as it may be extended to different users instead of only other farmers (e.g. agronomist)

5. **Farmer create forum thread**

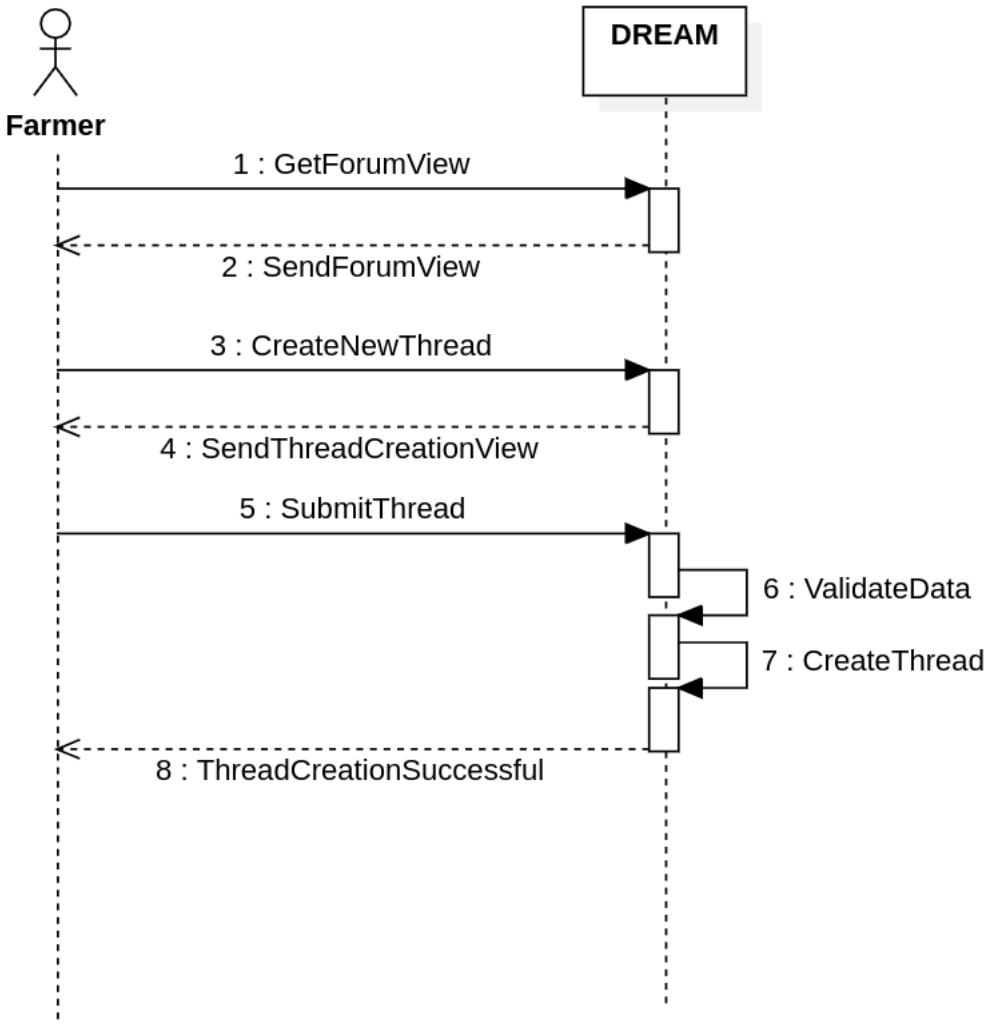

Figure 13: Sequence diagram for forum thread creation.

6. **Farmer comment on forum thread**

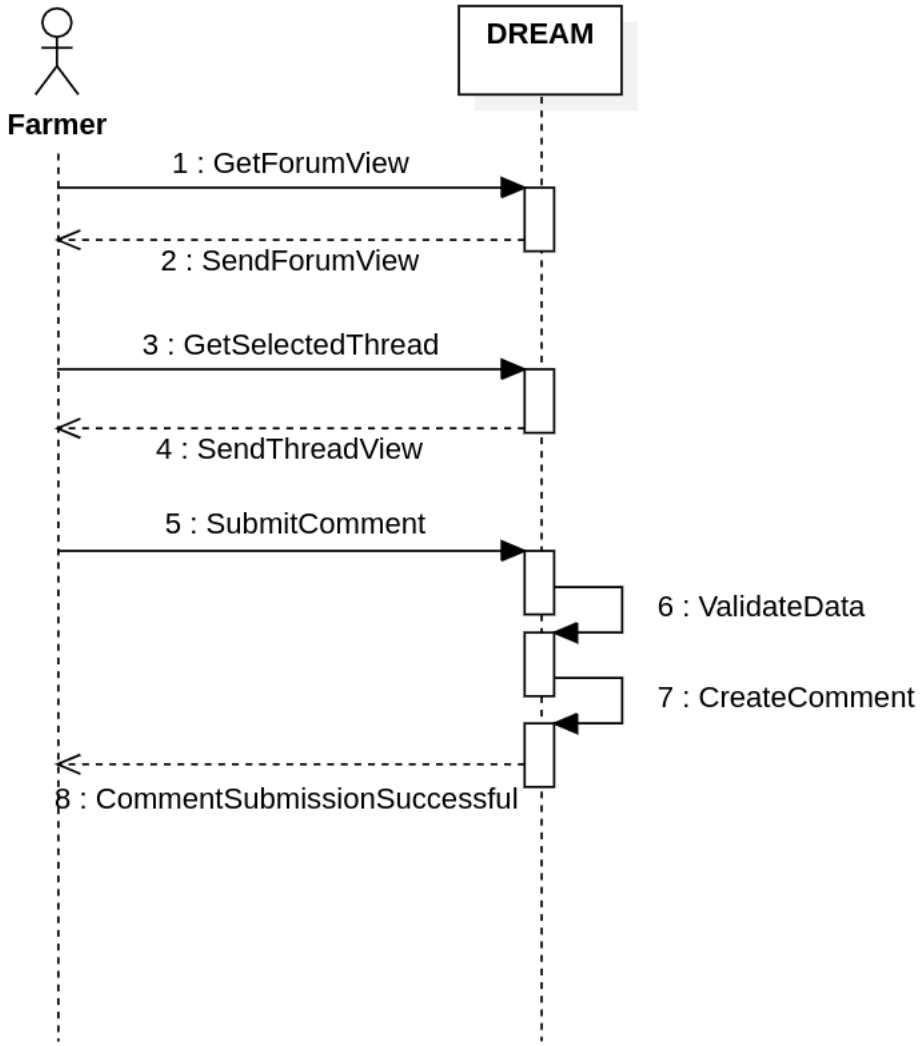

Figure 14: Sequence diagram for commenting on a forum thread.

7. **Farmer insert data about their production**

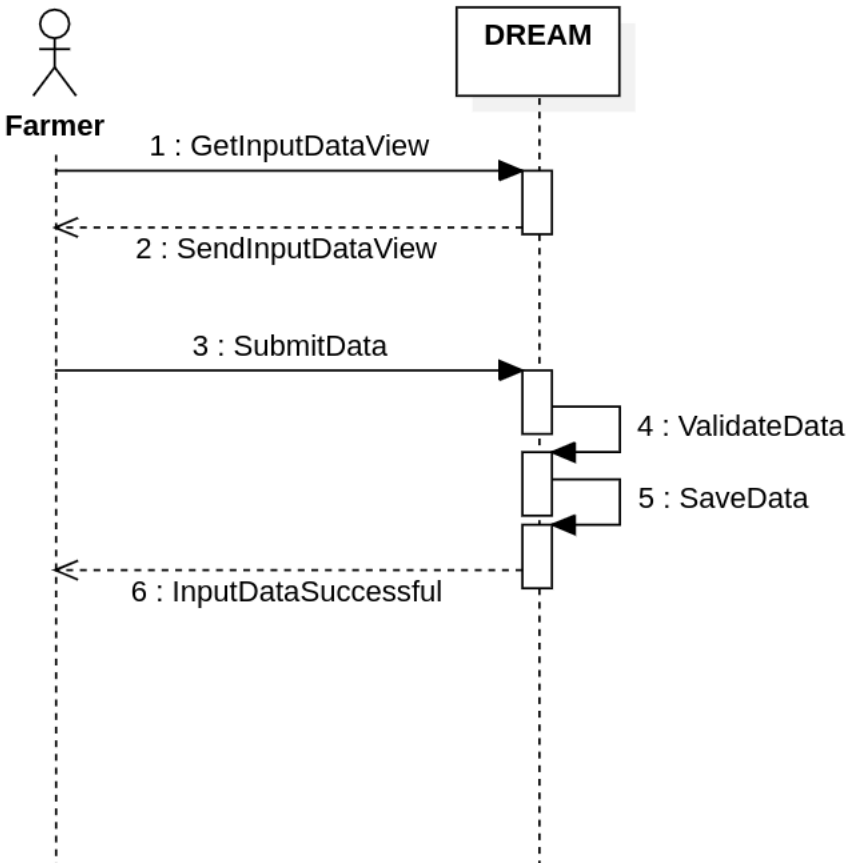

Figure 15: Sequence diagram of the process when a farmer inserts data about their production

8. **Farmer view personalized data**

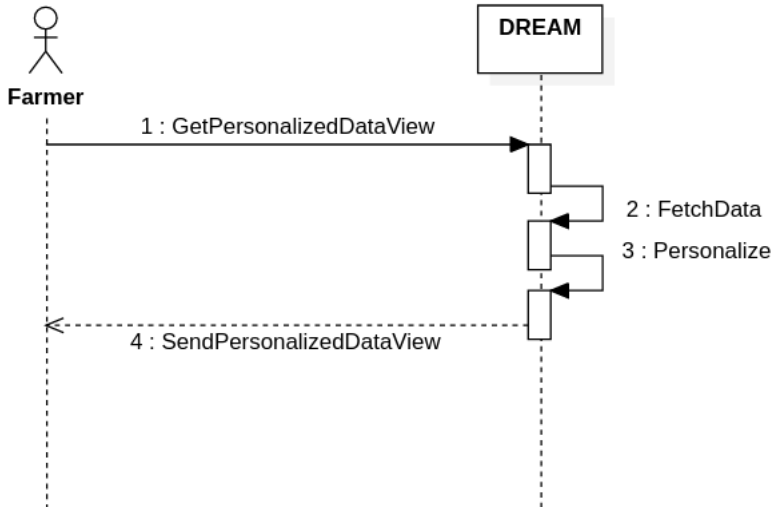

Figure 16: Sequence diagram for farmer viewing personalized data

9. **Policy maker view farmer performance**

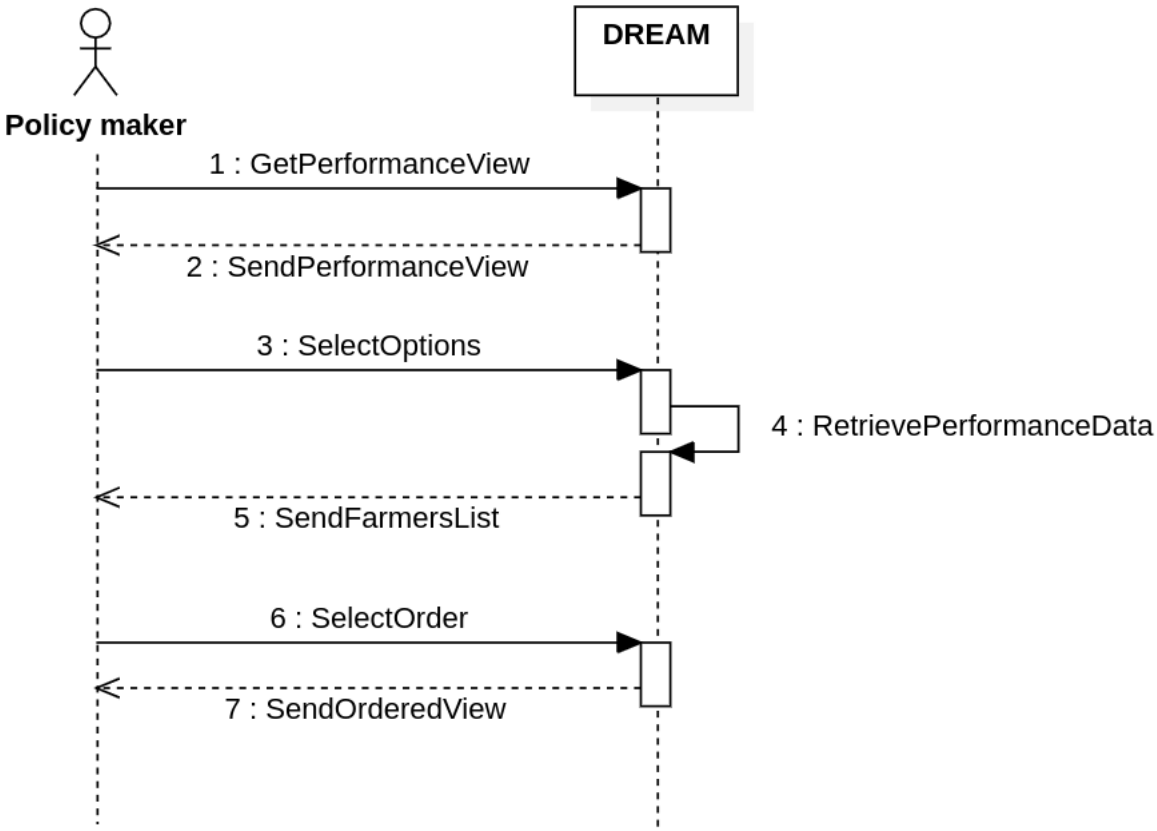

Figure 17: Sequence diagram of the process where a policy maker views the performance of farmers.

10. **Policy maker evaluate steering initiative**

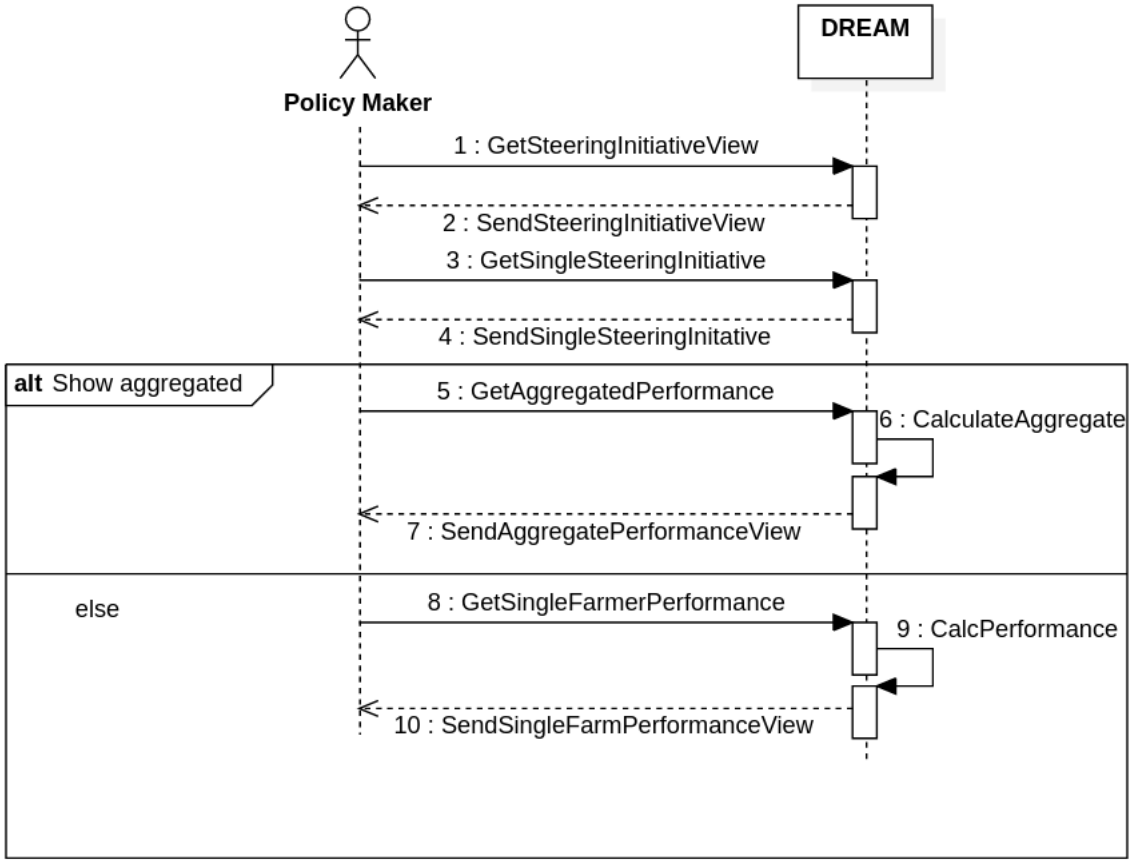

Figure 18: Sequence diagram of the process where a policy maker is evaluating the impact of a steering initiative. The possibility of evaluating a steering initiative aggregated or on a single farmer is modelled.

3.2.5 Mapping on requirements

Use case	Requirements
Farmer registration	R1, R13, R20, R21
User login to DREAM	R20, R21, R23, R24
Request help	R4, R14, R15, R20, R21, R23
Answer help request	R5, R20, R21, R23
Farmer create forum thread	R6, R16, R20, R21, R23
Farmer comment on forum thread	R7, R17, R21, R23
Farmer insert data about their production	R2, R3, R20, R21, R23
Farmer view personalized data	R8, R22, R23
Policy maker view farmer performance	R9, R18, R19, R20, R22, R24
Policy maker evaluate steering initiative on single farmer	R10, R11, R18, R22, R24
Policy maker evaluate steering initiative aggregated	R10, R12, R18, R22, R24

3.3 Performance Requirements

The system should be as light weight as possible in order to provide a better user experience for those with slow internet connections. It should not be assumed that all potential users have access to high speed internet, and since the goal is to facilitate a data driven governance model, the more users the better. Therefore, the system should minimize the amount of broadband needed in order to fully download the site. This could in practice boil down to e.g. removing unused CSS and Javascript, and sending data in smaller chunks (paginated or similar).

Secondly, the system should be able to handle many concurrent users, at least 100 000.

3.4 Design Constraints

3.4.1 Standards compliance

For one, all user data shall be treated in compliance with GDPR, or the equivalent privacy laws with jurisdiction in India.

Secondly, the application should function fully on all widely used web browsers.

Thirdly, the regulation and guidelines of external APIs must be followed.

3.4.2 Hardware limitations

As this system will be made available as a website, the only requirement on the hardware is that it can access the internet through a web browser. This could for example be a smartphone or a PC.

3.5 Software System Attributes

3.5.1 Reliability

The system should have a high reliability and be up and running most of the time. In order to achieve this, preventive regular maintenance should be performed. Additionally a duplication of the server could be run in parallel such that on failure of one the other can be put in place.

3.5.2 Availability

While the goal is to have as high of an availability as possible, for this system in particular it is not devastating with some downtime. For example, if the system is down when a farmer wants to insert data about the production, they could do it the following day instead without much problem.

With that in mind, we deem that an availability of 99% is reasonable for this system. Resulting in about 4 days where the system is unavailable, which could be used for maintenance days etc. To achieve this availability level, we may run a duplication of the server in parallel, and possibly also the database.

It should also be noted that some of the functionality of the system relies on external APIs out of our control, though the system should not completely fail because of failure in one of those.

3.5.3 Security

The system is storing and handling information where some of it is private, so it is of great importance that the system security is of a high standard. Do note that this does not only apply to the obvious one's in password and fiscal. For example, the data inserted by a farmer about their production should be considered private and be protected as it could for example be an indication of income etc.

In practice, this means encryption of passwords, and authorization for every

API endpoint used within the system.

3.5.4 Maintainability

The system should should be easy to maintain, and facilitate future extensions of the software. To achieve this it should adhere to some set of software patterns, and not deviate from them. Moreover, there needs to be clear documentation over the whole system both in a separate document as well as comments directly in the code.

3.5.5 Portability

As previously mentioned, the system should be able to function on all widely used web browser. Moreover, the design and functionality should work well regardless of the screen size used to access the site. I.e. it should not matter if you are using a mobile device or a PC.

4 Formal analysis

4.1 Alloy code

In this section a formal analysis of the system using Alloy is presented. The purpose of this is to highlight some of the constraints that should be imposed. Additionally we want to show that the model of our system is satisfiable.

Below is the code where we have modeled the DREAM system. The system is modeled by the direction of the class diagram, every class is included here except "ProdSuggest".

With this model, we formally show that our approach to constructing the system is sound and consistent. Meaning, that the relations between classes and constraints imposed results in something that is reasonable and useful. This also justifies the modeling activity.

```
// Signatures
sig PersonId{}

sig Type{}
sig Location {}
sig Identification {}
```

```
abstract sig Role {}
one sig FARMER extends Role {}
one sig POLMAK extends Role {}

abstract sig Bool {}
one sig TRUE extends Bool {}
one sig FALSE extends Bool {}

sig Email {}
sig User{
        email: one Email,
        password: one String,
        personId: one PersonId,
        isActive: one Bool,
        role: one Role
        }

sig Farm {
        location: one Location,
        identification: one Identification,
        farmers: set User,
        products: set Product
        }{
                all f: farmers | f.role = FARMER
        }

sig Product{
        type: one Type,
        amount: one Int,
        fromDate: one Date,
        toDate: one Date,
        } {
                amount > 0
                fromDate.date < toDate.date
        }

sig Problem{
        description: one String,
        fromDate: one Date,
        toDate: one Date,
        farm: one Farm
        }{
                fromDate.date < toDate.date
        }

sig SteeringInitiative{
```

```
        startDate: one Date,
        responsible: one User,
        farms: set Farm
        }{
                responsible.role = POLMAK
        }

sig ForumThread{
        title: one String,
        datePosted: one Date,
        content: one String,
        author: one User,
        comments: set Comment
        }{
                author.role = FARMER
        }

sig Comment{
        content: one String,
        datePosted: one Date,
        author: one User,
        }{
                author.role = FARMER
        }

sig HelpRequest{
        title: one String,
        description: one String,
        isAnon: one Bool,
        datePosted: one Date,
        isActive: one Bool,
        author: one User,
        comments: set Comment
        }{
                author.role = FARMER
        }

// We use a simplified model of a date
sig Date {
        date: one Int
} {date >= 0}


// Predicates and facts

fact allUserHasOneRole{
```

```
        all u: User | one r: Role | r in u.role
        }

fact allFarmerHasFarm{
        all u: User |
                (some f: Farm | u.role = FARMER && u in f.farmers) or
                (u.role = POLMAK)
}

fact uniqueEmails{
        no disjoint u1, u2: User | u1.email = u2.email
}

-- All farms in different location
fact uniqueLocFarm {
        no disjoint f1, f2: Farm | f1.location = f2.location
}

fact uniqueFarmId {
        no disjoint f1, f2: Farm | f1.identification = f2.identification
}

fact uniquePersonId {
        no disjoint u1, u2: User | u1.personId = u2.personId
}

-- All comments are either on a thread or a help request (but not both)
fact allCommentsConnected {
        all c: Comment |
                (one ft: ForumThread | c in ft.comments) iff not
                (one hr: HelpRequest | c in hr.comments)
}

fact allCommentsOlder{
        all c: Comment |
                (all hr: HelpRequest | c in hr.comments
                        implies c.datePosted.date > hr.datePosted.date) &&
                (all ft: ForumThread | c in ft.comments
                        implies c.datePosted.date > ft.datePosted.date)
}

fact allDatesUnique {
        no disjoint d1, d2: Date | d1.date = d2.date
}

//-------------------------------------------------------------------------
```

```
// Such that everything is connected in the world
fact allIdentifactionConnected{
        all i: Identification | one f: Farm | i = f.identification
}


fact allLocationConnected{
        all i: Location | one f: Farm | i = f.location
}

fact allPersonIdConnected{
        all p: PersonId | one u: User | p = u.personId
}
fact allEmailConnected{
        all e: Email | one u: User | e = u.email
}

fact allTypeConnected{
        all t: Type | one p: Product | t = p.type
}

fact allProductConnected{
        all p: Product | some f: Farm | p in f.products
}
//-------------------------------------------------------------------------

-- Just to get strings to resolve
fact stringPool{
   none!= "a"+"b"+"c"+"d"+"e"
}

/* Dynamic modeling */

pred createForumComment [ft, ft': ForumThread, c: Comment] {
        ft'.comments = ft.comments + c
}

pred createHRequestComment [hr, hr': HelpRequest, c: Comment] {
        hr'.comments = hr.comments + c
}

pred addProduct [f, f': Farm, p: Product] {
        f'.products = f.products + p
}

run createForumComment
```

```
/* ---------------- */

pred world1 {
        #User>=2
        some u: User | u.role = FARMER
        #Farm>1
        #SteeringInitiative=2
        #Product=2
        #Problem=1
}

pred world2 {
        #ForumThread = 2
        #HelpRequest >= 1
        #User = 3
        #Comment = 4
}

run world1 for 5

run world2 for 5
```

4.1.1 First model

In the first model we focus on the relations between farmer, farm, products and steering initiatives. With this, we want to show, among other things, the following:

1. Every farmer must be associated to a farm
2. Only farmers can be associated to a farm
3. A steering initiative must be associated to a policy maker
4. Every farm identifier is unique
5. Every farm has a unique location

The first model is the predicate world1 in the alloy code.

4.1.2 Second model

This model shows how the forum and help requests works in the system. Some of the constraints imposed are:

1. Only farmers can create forum threads
2. Only farmers can comment
3. Only farmers can create help requests
4. A comment belongs to either a forum thread or a help request
5. Comments are older than the forum thread or the help request it was posted in

4.1.3 Dynamic model

We also modeled some dynamics of the system. However no additional constraints are shown with that.

4.1.4 Resulting worlds

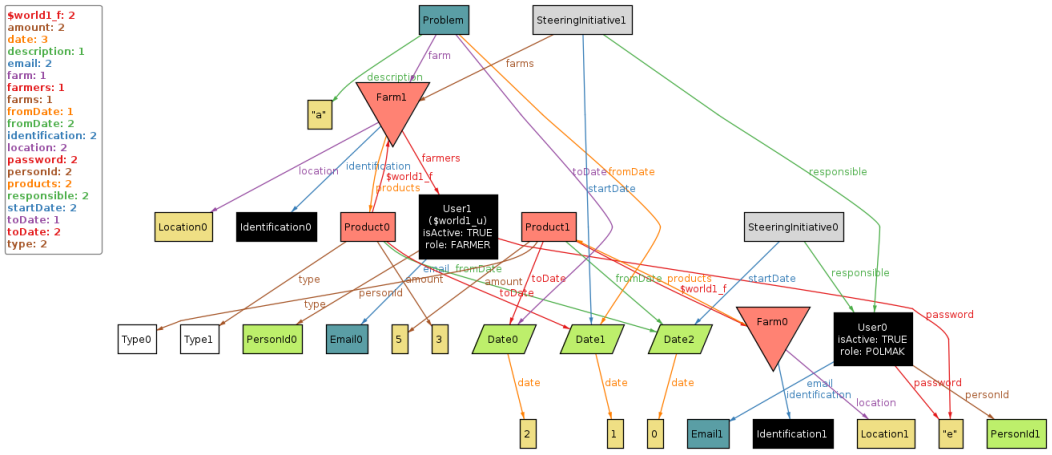

Figure 19: A world obtained for the first model

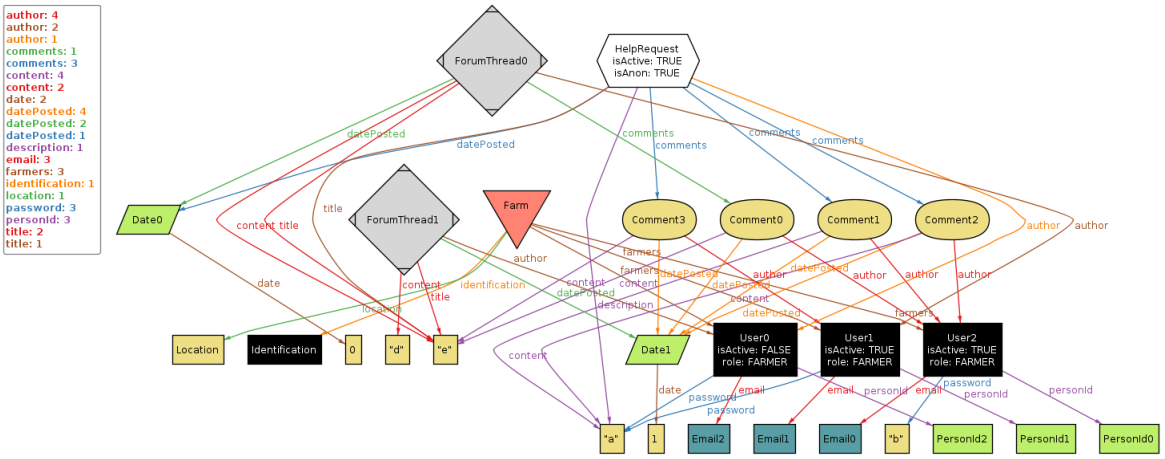

Figure 20: A world obtained for the second model

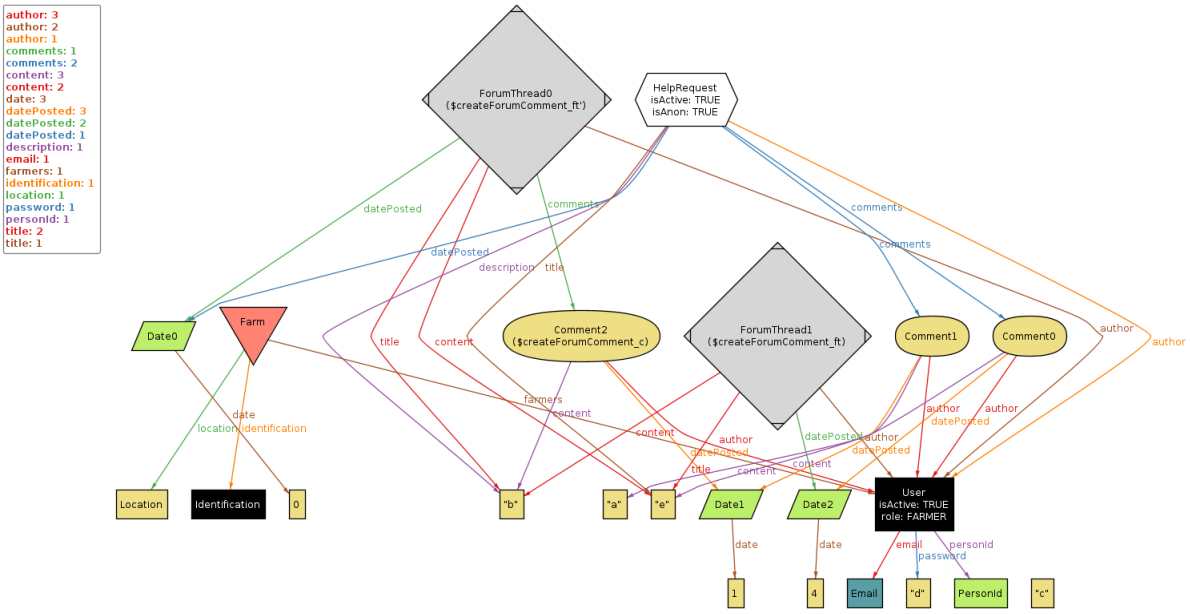

Figure 21: A world obtained for the dynamic model

5 Effort spent

5.0.1

Task	Time spent
Introduction	8 h
Overall description	9 h
Specific requirements	13 h
Formal analysis	6 h
Reasoning	10 h
Total	46 h

5.0.2

Task	Time spent
Introduction	5 h
Overall description	8.5 h
Specific requirements	12 h
Formal analysis	6 h
Reasoning	10 h
Total	41.5 h

6 References