POLITECNICO
MILANO 1863

Requirement Analysis & Specification Document

Deliverable:	RASD
Title:	Requirement Analysis and Specification Document
Authors:	Leonardo Gori, Marco Romanini, Yui Watanabe
Version:	1.0
Date:	December 23, 2021
Download page:	`https://github.com/MarcoRomanini/GoriRomaniniWatanabe`
Copyright:	Copyright © 2021, Leonardo Gori, Marco Romanini, Yui Watanabe – All rights reserved

Contents

List of Figures

List of Tables

1 Introduction

DREAM is an easy-to-use application which intent is [cite assignment] to design dynamic anticipatory governance models for food systems using digital public goods and community-centric approaches to strengthen data-driven policy making in the state of Telangana, India.

The state's main means of livelihood indeed heavily rely on agriculture, widely represented by smallholders farmers' activities. These ones are easily affected by complex problems such as climate change and the incoming raise of food demand due to the continuous population increase. In addition to that, the occurrence of `Covid-19` pandemic is recently causing further obstacles (such as food supply chains disruption) to the achievement of a resilient food system.

The final aim of Telengana's government is collecting and analyzing agriculture related real-time conditions in order to monitor and support smallholders' activity resilience capacity against the above mentioned problems.

1.1 Purpose

The main purpose of the S2B is acting as a bridge between the main actors of the food systems: **policy makers**, **farmers** and **agronomists**. This can be achieved through the development of a multi-user unified platform that allows actors to share data. In particular, the system should guarantee the following needs and functionalities based on the user identity; we present them in the following list, extracted from the original assignment.

Policy makers needs

1. Identify those farmers who are performing well, especially when they demonstrate to be resilient to meteorological adverse events, as these farmers will receive special incentives and will be asked to provide useful best practices to the others
2. Identify those farmers who need to be helped as they are performing particularly badly
3. Understand whether the steering initiatives carried out by agronomists with the help of good farmers produce significant results

Farmers needs

4. Visualize data relevant to them —- for instance, weather forecasts, personalized suggestions concerning specific crops to plant or specific fertilizers to use — based on their location and type of production
5. Insert in the system data about their production and any problem they face
6. Request for help and suggestions by agronomists and other farmers
7. Create discussion forums with the other farmers

Agronomist needs

8. Insert the area they are responsible of;
9. Receive information about requests for help and answer to these requests
10. Visualize data concerning weather forecasts in the area and the best performing farmers in the area
11. Visualize and update a daily plan to visit farms in the area, assuming that all farms must be visited at least twice a year, but those that are under-performing should be visited more often, depending on the type of problem they are facing
12. Confirm the execution of the daily plan at the end of each day or specify the deviations from the plan

Referring to point 1, we assume that is policy maker responsibility, given information about well performing farmers by the system, to send incentives through the platform itself.

Referring to points 1 and 2, we assume that the system is able to provide a well desired evaluation of farmers performance status. The evaluation could be made based on their production rates, climate change resiliency and the location of their activity.

Referring to point 4, we provide an informal definition of the information that is relevant for farmers in section 1.4.1. We also assume that weather forecasts and personalized suggestion are internal behaviors of the S2B.

Referring to point 11, we assume that the system automatically prepares for the agronomists some sketches of the daily plans. Agronomists are then allowed to edit them as they feel more comfortable. The suggestion could be made based on their current assigned farmers information (such as the performance status and the number of previous visits) and the areas they chosen to be responsible of.

Furthermore the system should make use of the already collected data such as:

- 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.

1.2 Scope

The environment in which the platform will be involved will be the whole state of Telangana. Policy makers, farmers and agronomist will have the possibility to access the system through their personal devices. Furthermore, the devices responsible for gathering information, such as the ones of the water irrigation system and the humidity sensors, are supposed to be shared along the territory in a distributed system fashion.

1.2.1 World Phenomena

In this section the environment related phenomena that the S2B is supposed to face are introduced. As described in [4], for each phenomenon we specify if it is shared and who controls it.

Phenomenon	Who controls it?	Is it shared?
User registration	M	Y
User login	W	Y
Check usernane and password	M	N
Visualize the data	M	Y
Gather and send the information about humidity	M	Y
Send the data automatically	M	Y
Setup the systems and locate them	W	Y
Fetch info from the located systems	M	Y
Gather and send the information about each crops condition	M	Y
Put the flag according to the given behaviour of each farmers	W	Y
Visualize the sorted tendency of farmers characteristics	W	Y
Adjust the threshold(distinguish good/problematic farmer)	W	Y
User insert data about their production	W	Y
User insert problems faced information	W	Y
User asks for suggestion to agronomists and/or farmers	W	Y
System notifies addressed users about a farmer request	M	Y
Farmers problems take place	W	N
User create discussion forums	W	Y
Users insert the area they are responsible for	W	Y
User answers requests for help	M	Y
Visualize data concerning weather forecast in the area	M	Y
Visualize best/under performing farmers in the area	M	Y
Visualize daily plan	M	Y
Update daily plan	W	Y
Confirm the execution of the daily plan	W	Y
Specify deviation from the plan	W	Y
Check farmers that are under-performing	M	N
Check farmers that are performing well	M	N
Check farms that need to be visited	M	N

Table 1: Phenomena table

1.3 Goals

According to [5] (revised), it is stakeholders' prerogative to determine the goals. Thus, in this section, in order to introduce some sort of atomicity of the goals, we derive a formal description of them from the requested functionalities presented in section 1.2. Furthermore, in order to justify the definition of each goal, we also map each of them to the related assignment extract.

GX	Definition	Cfr.
G.1	Policy maker should be able to see best performing and under-performing farmers in all the areas	1,2
G.2	Policy maker should be able to handle incentives for high performing farmer	1
G.3	Policy maker should be able to ask high performing farmers to write good practices	2,3
G.4	Policy maker should be able to compare the performance difference of farmer between the current data and the past data(before agronomist visits)	1
G.5	Farmers should be able to visualize data relevant to them, based on their location and type of product	4
G.6	Farmers should be able to manipulate data about their production and any problem faced	5
G.7	Farmers should be able to request for help and suggestions by agronomists and other farmers	6
G.8	Farmers should be able to create discussion forums with other farmers	7
G.9	Agronomists should be able to insert the area they are responsible for	8
G.10	Agronomists should be able to receive information about requests for help and answer to these requests	9
G.11	Agronomists should be able to see best performing and under-performing farmers in the area	10
G.12	Agronomists should be able to see data concerning weather forecasts in the area	10
G.13	Agronomists should be able to visualize and update a daily plan to visit farms in the area	11
G.14	Agronomists should be able to confirm the execution of the daily plan at the end of each day or specify the deviations from the plan	12

Table 2: Goals table

For each of them we provide the **traceability matrix** in section 3.2.5.

1.4 Definitions, Acronyms, Abbreviations

In order to introduce some sort of coherence in an environment— the physical world— that is informal, we present in this section the sets of **definitions**, **acronyms** and **abbreviations** used in the following sections. These are supposed to be as more generic as possible, in order to provide more flexibility for the following phases of the Waterfall software lifecycle.

It is important to agree in advance on the specific keywords and terms that signal the presence of specific entity. Following the convention defined in [3], we therefore adopt the following terms to identify and describe requirements, goals and domain assumptions.

- **Shall**: used to indicate a requirement that is contractually binding, meaning it must be implemented, and its implementation verified;
- **Will**: used to indicate a domain assumption. Will statements are not subject to verification;
- **Should**: used to indicate a goal which must be addressed by the design team but is not formally verified.

1.4.1 Definitions

Concept	Definition
PRODUCTION DATA	All general kind of information that describes what the farmer produces with his activity. We allow the farmer to describe it with a set of mandatory properties that are analogous to each product (e.g. type of product, amount of produced items, unit of measurement etc.). In addition, the user can fill some optional fields like a qualitative description of the items produced or notes relevant for other users of the system
PROBLEM INFORMATION	Every kind of information that describes in a functional way the general set of events that could get in the way of farmers' ordinary production rate
HELP REQUEST	The farmer's tool to contact agronomists. The farmer should be able to ask questions in a private way to other users of the systems (farmers and agronomists) in order to increase the quality of his activity
GOOD PRACTICES DOCUMENT	Document in which the practices that good farmers follow are described. This document is made to be submitted into the system and can be useful for those farmers who want to improve their production
PERFORMANCE	With the term **performance** we refer to the evaluation of farmers, based on their productivity and resilience. Depending on their deviance (positive or negative), farmers can be classified as high-performing, normal-performing or under-performing
INCENTIVE	With the term **incentive** we refer to some sort of voucher which can be given to farmers if they satisfy certain conditions
FARMER'S RELEVANT INFORMATION	The information that is considered to concern farmers: weather forecasts, personalized suggestion, soil humidity data, water irrigation system details, visits calendar of assigned agronomists, history of past faced problems and uploaded product information

Table 3: Table of definitions

1.4.2 Acronyms

Acronym	Expansion
DREAM	Data-dRiven prEdictive fArMing
S2B	Software to be
GDPR	General Data Protection Regulation
ECOWAS	Economic Community of West African States
DPGS	Digital Public Goods Standard
STQC	Standardization Testing and Quality Certification
UML	Unified Modeling Language
BPMN	Business Process Model and Notation
SDG	Sustainable Development Goals
MTTR	Mean Time To Repair
RBAC	Role Based Access Control
DBMS	Data base Management System
API	Application programming interface
GUI	Graphical User Interface

Table 4: Table of acronyms

1.5 Revision history

Date	Description
29/10	World and Machine phenomena
31/10	UML class diagram
10/11	Goals, Domain assumptions, Requirements
17/11	UML class diagram
19/11	Functional requirements
22/11	Actors
	Perspective (interfaces)
	Interface requirements
24/11	Perspective (interfaces)
	Software system attributes
	Interface requirements
	Functional requirements
26/11	Performance requirements
	Design constraints
	Functional requirements
	Product functions
	UI description
29/11	Functional requirements
	Farmer BPMN
1/12	Alloy code
	farmer sequence diagram
	Users section introduction
3/12	Goals, Domain assumptions, Requirements
	Alloy code

	Farmers goals mapping
06/12	Alloy code
	UML
	UI description
	Goals, Domain assumptions, Requirements
	Introduction
07/12	Functional Requirements (sequence diagrams, scenarios)
	UI design
	Sequence diagram
	bibliography
	latex edits, acronyms table
10/12	UML diagram
	Functional Requirements
	Product functions
	Farmers scenarios
	Use case diagram
	Policy maker scenarios
15/12	BPMN
	Alloy code
	UML diagram
	Goals, Domain assumptions, Requirements
	Use case
	Transcribed goals and requirements
	Transcribed domain assumptions and traceability matrix
17/12	UML description
	Added Alloy code with assertion
20/12	Functional requirements
	Alloy code
	UML diagram
	Goals, Domain assumptions, Requirements
	Sequence diagram
	Use cases
	Alloy code transcription
	Introduction revise
21/12	Effort table
	Whole document review

Table 5: History table

1.6 Reference Documents

We present in this section the references we used to gather information about the good practices for the development of this document.

[1] Angtrim. alloy-latex-highlighting. https://github.com/Angtrim/alloy-latex-highlighting, 2019. Last accessed December 2021.

[2] Digital Public Goods community. Digital public goods standard. https://digitalpublicgoods.net/standard/, 2021. Last accessed December 2021.

[3] ISO/IEC/IEEE. Iso/iec/ieee international standard - systems and software engineering – life cycle processes – requirements engineering. *ISO/IEC/IEEE 29148:2018(E)*, pages 1–104, 2018.

[4] Michael Jackson. The world and the machine. In *Proceedings of the 17th International Conference on Software Engineering*, ICSE '95, page 283–292, New York, NY, USA, 1995. Association for Computing Machinery.

[5] Michael Jackson and Pamela Zave. Deriving specifications from requirements: an example. In *1995 17th International Conference on Software Engineering*, pages 15–15, 1995.

[6] LeoGori-MarcoRomanini-y1220. Requirement engineering and design project: goal, schedule, and rules. https://github.com/MarcoRomanini/GoriRomaniniWatanabe/blob/main/01.%20Assignment%20RDD%20AY%202021-2022.pdf, 2021. Last accessed December 2021.

[7] Parvathy Krishnank Swetha Kolluri. Data4policy. https://github.com/UNDP-India/Data4Policy, 2021. Last accessed December 2021.

[8] Standardisation Testing, Quality Certification (STQC) Directorate of the Ministry of Electronics, and Government of India Information Technology. Standards for e-governance applications. http://egovstandards.gov.in/notified-standards-0, 2021. Last accessed December 2021.

[9] Hans van Vliet. *Software Engineering: Principles and Practice*. Wiley, 2007.

1.7 Document structure

The overall document is organized in 5 main sections. For each of them we provide a brief explanation of the contents of their subsections, except for the current one.

Introduction

Here in section 1.1 we offer a brief description of the problem and required functionalities, with some of our assumptions about the terms used in the original assignment. In section 1.2 we briefly describe the environment in which the S2B will be involved and present the phenomena analysis table. In section 1.3 a table of the required goals is presented and defined, while section 1.4 contains the list of definitions of terms and acronyms used along the document. Section 1.5 contains the history of reviewed section of the document during time passing. Finally, section 1.6 provides the list of useful sources that have been exploited to build the document.

Overall description

This section offers a summary description about the overall organization of the system, the Hardware and Software constraints and the interfaces needed to get it work. In particular section 2.1 proposes the UML high level class diagram structure of the application and a brief discussion about the entities contained in it. In section 2.2 we provide a description of the main features offered by the application and the relatives BPMN diagrams for deeper understanding. The main actors that are supposed to interact with the application are defined in section 2.3, while in section 2.4 we define the list of domain assumptions.

Specific Requirements

In section 3.1 some mock-ups of the web and application GUIs are showed and described. Section 3.2 represents the main portion of the document and contains the list of functional requirements described by means of actor's scenarios, use case diagrams, tables and sequence diagrams. Performance requirements are defined in section 3.3, while in section 3.4 we present the design constraints. Finally in section 3.5 we provide the main expected qualitative properties the platform is required to hold.

Formal Analysis using Alloy

Finally, in section 4.1 we provide the Alloy code used for the analysis of the system, together with some diagrams to better understand the behaviour of the system.

Effort spent

Here we provide the table of the overall efforts building the documents since last update for each team member.

2 Overall Description

2.1 Product perspective

DREAM is a functional multi-user software platform whose purpose is to provide functionalities described in section 2.2. The system will be composed by a series of software and hardware interfaces that interact in such a way to let users manipulate shared data. It also will exploit some graphical interface packages in order to be user friendly and easy to use. This product is designed to run on a wide variety of machines, including operating systems Mac OS, Windows, Linux, Android and iOS.

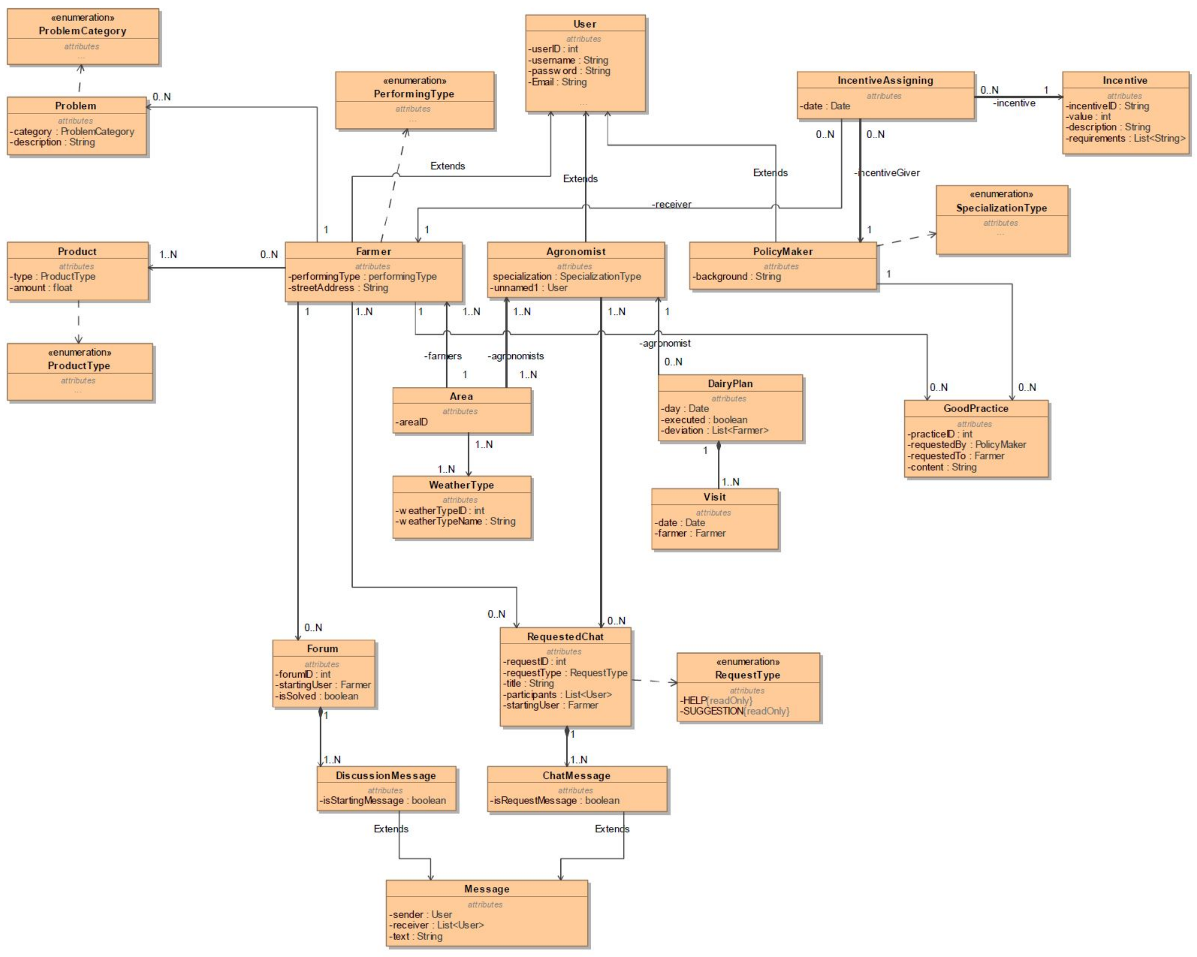


Figure 1: High level UML diagram

UML description

Class	Description
USER	This class represents the people registered to the system, with their credentials
FARMER	This class represents the farmers, with their performing type and their street address (information useful to agronomists)
AGRONOMIST	This class represents the agronomists, with their specialization type
POLICYMAKER	This class represents the policy makers, with their background (e.g., India's government, Telangana's government, United Nations, etc)

AREA	This class represents the areas in which Telangana has been divided for the management of this system. Areas can be the 33 districts in which Telangana is formally divided, but a different subdivision criteria can be used
WEATHERTYPE	This class represents some characteristics of the area regarding weather aspects (e.g., humidity, rainfall frequency, average temperature, etc)
FORUM	This class represents the discussion forums that farmers can use to communicate with each other, to get information and to exchange ideas. Only farmers can write in forums
REQUESTCHAT	This class represents the requests that farmers can send to agronomists and to other farmers. Requests can be for help or for suggestions. A request is modelled as a chat where the participants are selected by the farmer that makes the request. A request always has at least one agronomist as a participant
MESSAGE	This class represents the messages exchanged in forums and chats across the platform, with their sender, receivers and text
DISCUSSIONMESSAGE	This class represents the messages belonging to forums. For every forum there is only one starting message, while all the other messages are considered as replies to that message
CHATMESSAGE	This class represents the messages belonging to request chats. For every request there is only one message of type "request "(the first message), while all the other messages are considered as "reply". Every chat message is delivered to all the participants of that chat (excluding the sender, of course)
DAILYPLAN	This class represents the daily plans of the agronomists, with the date, the list of visits for that day and the list of unvisited farmers (once the plan has been executed)
VISIT	This class represents the visits that agronomists arrange for checking the farmers' activity
PRODUCT	This class represents the products that are inserted in the system by the farmers, with their type and their amount
PROBLEM	This class represents the problems that farmers may encounter, with their category and description
INCENTIVE	This class represents the incentives that are available for farmers, with their description, their value and their set of requirements. Incentives are modelled as some sort of vouchers
INCENTIVEASSIGNING	This class represents the assignments of incentives, expressed as a mapping between "when", "to who" and "from who" a certain incentive has been given
PERFORMINGTYPE	This enumeration represents the performance type of farmers, giving information about how good a farmer is doing (for example: well-performing, normal-performing, under-performing)
SPECIALIZATIONTYPE	This enumeration represents the type of agronomists' specializations
REQUESTTYPE	This enumeration represents the type of requests, which can be for help or for suggestions
PROBLEMCATEGORY	This enumeration represents the type of problems that farmers can face and insert into the system

Table 6: UML description table

2.1.1 User interfaces

According to the assignment document, the system will interact with 3 different user classes: policy makers, farmers and agronomists. In order to be more accessible and to fulfill the user needs, the application will be supported by different devices. These users should interface to the service through electronic devices with an internet connection. Users that need to access the service will have the possibility to connect through:

- an internet browser, addressing a specific web domain (such as *www.dream.com*) that permits users to sign up/in a dedicated web application;
- a mobile application that can be installed on smartphones or tablets (both iOS and Android).

2.1.2 Software interfaces

In order to improve software flexibility and quality, DREAM will use a set of external software interfaces. Rather than providing names of real specific services, we consider reasonable referring to them as functionalities to be later defined in the design phase:

UNIVERSAL LOGINS
Login APIs that also provide access by using their Facebook, Twitter, or Google profile login details are good candidates in order to quickly authenticate the user while guaranteeing security.

BIG DATA MANIPULATION
Since a wide quantity of information needs to be recorded and accessed in a distributed system fashion, DBMS APIs are necessary for data extraction performances optimization.

THIRD PARTY DATA SETS ACCESS
The system will use open data sets to obtain information about weather forecasts, soil moisture, water irrigation, humidity and so on.

FARMERS EVALUATION
To evaluate the performance of farmers, the system relies on external APIs that use specific algorithms to understand how good a farmer is doing (positive or negative deviance).

INCENTIVES MANAGEMENT
The system will rely on an external service for the definition of incentives and for their money collection by the farmers.

WEATHER TYPES CATEGORIZATION
To define the weather type categories of the areas, the system will use some sort of API to analyse areas characteristics and tendencies in order to define shared phenomena.

2.1.3 Hardware interfaces & constraints

DREAM system will be composed by multiple different hardware components which can be described from two points of view:

USER PERSPECTIVE
Since DREAM platform is accessed by users in a fully virtual fashion, the minimum required hardware interfaces are the ones that provides internet connection, input components, a screen to visualize GUI and a web browser or an application store (like smartphones, personal computers, tablets and smart TVs).

SYSTEM PERSPECTIVE
According to the assignment, the system should be composed by hardware devices designed to gather Telangana's environment information such as soil humidity sensor and the ones responsible for the predefined water irrigation system.

The user hardware interfaces also represent constrains that are required in order to permit the users to interact with the systems and manipulate shared data.

2.2 Product functions

In this section the main functionalities of the S2B are presented, described and enriched with BPMN diagrams in order to guarantee an higher level of understanding.

2.2.1 Sign up

This functionality allows the user to create an account to access the platform. Firstly he opens the sign up page and fills the information required such as email, address etc. Then, if the inserted data is accepted, an e-mail is sent to the User asking for their verification. Lastly, if all the steps above are done, the user is redirected to the login page.

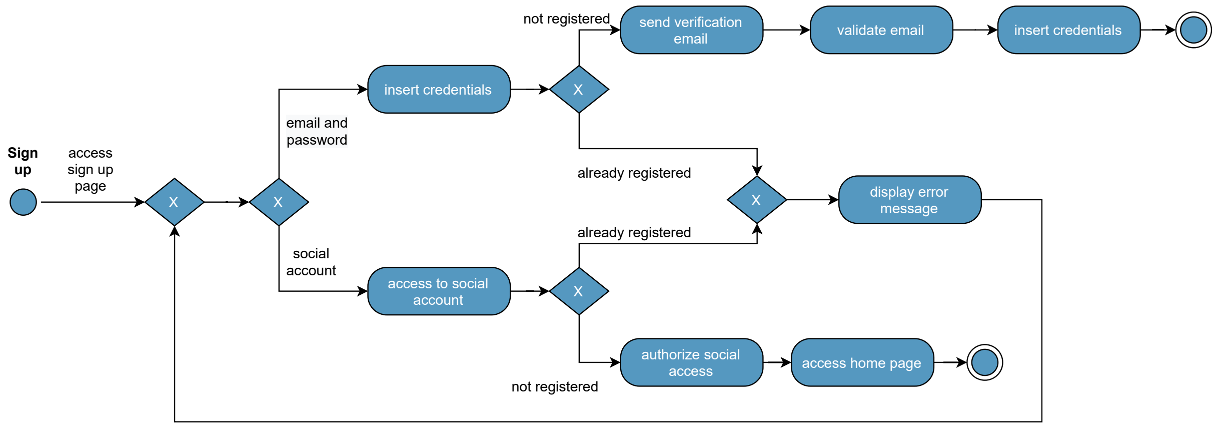


Figure 2: BPMN diagram of sign Up

2.2.2 Sharing issues to get help and suggestions

This functionality is available for farmers and agronomists. The farmer selects the request section and the system displays the send request button; if it is clicked, all the saved contacts are shown. After selecting to whom to ask, insert the question in the text form, presses send button, then the request is sent successfully.

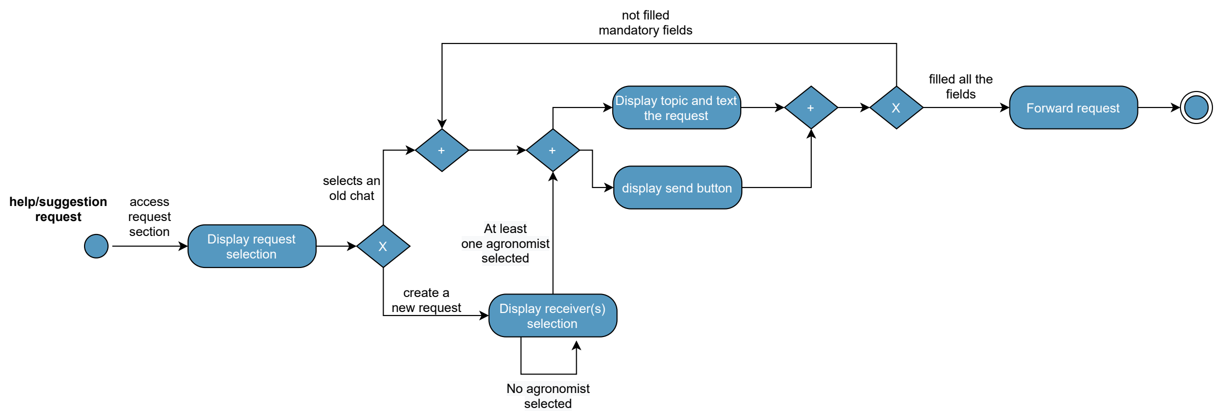


Figure 3: BPMN diagram of help/suggestion request

2.2.3 Communication (among farmers) on forums

This functionality is required for allowing farmers to exchange their opinions. The farmer accesses the forum section which presents an eventual list that contains both previous submitted forums and farmer's forum replies. By selecting forum upload button, the system displays insertion form containing topic/context of the thread, title and question content. If the farmer inserts all the required data correctly and presses the submission button, confirmation page is shown. Lastly, by clicking confirm button the forum is generated.

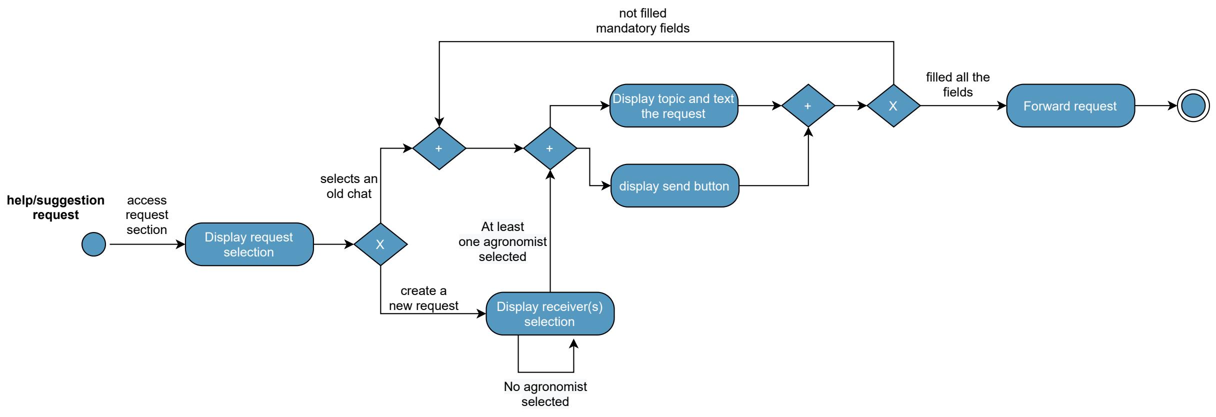
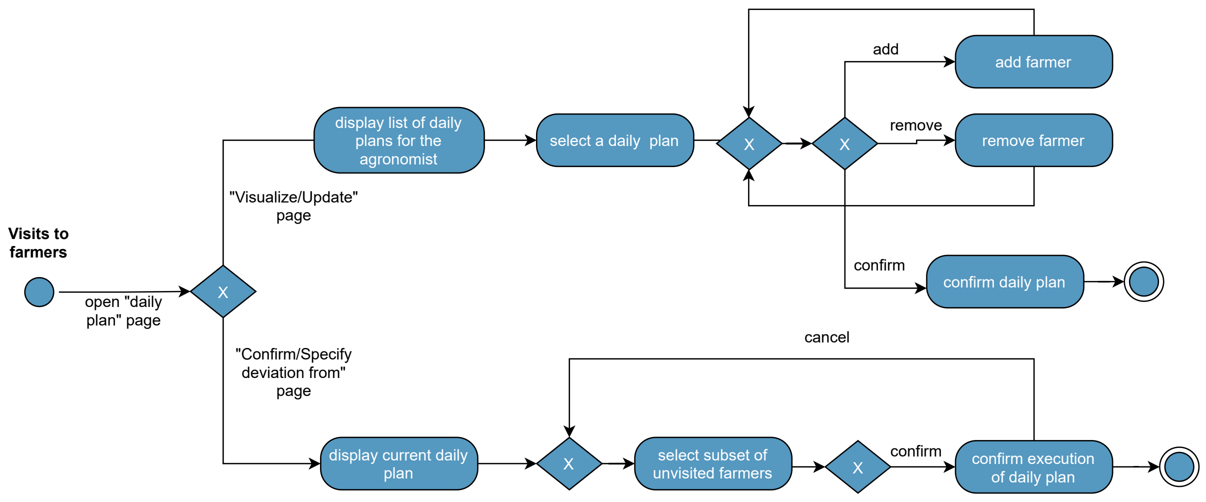


Figure 4: BPMN diagram of forum generation

2.2.4 Visits to low performing farmers

This functionality is necessary to organize the visits in order to make the schedule well-shared between an agronomist and a farmer. The agronomist goes to the daily plan section and the application displays a visualize or update button. If it is clicked, the system extracts their schedule data which is shortly expressed by a form with day-month-year and the farmers to visit. If there is a wish to modify the plan, it is possible to modify it in a form guided by selecting the update button.



add
add farmer
remove
remove farmer
display list of daily plans for the agronomist
select a daily plan
"Visualize/Update" page
Visits to farmers
confirm
confirm daily plan
open "daily plan" page
cancel
"Confirm/Specify deviation from" page
display current daily plan
select subset of unvisited farmers
confirm
confirm execution of daily plan

Figure 5: BPMN diagram of visit farmers

2.2.5 Visualize data about performance

This functionality is used by policy makers. The policy maker clicks the farmer's performance button and the system visualize the list of farmers grouped by their production performance. If there is any interesting farmer, by selecting his/her name, the application shows the information of his/her production history.

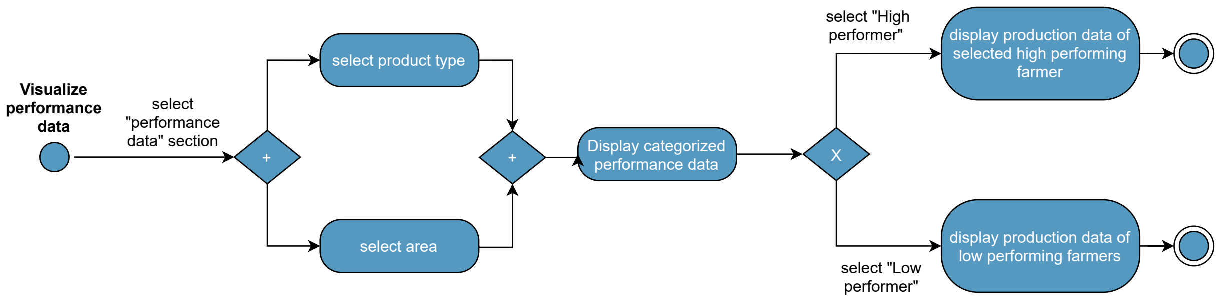

Visualize performance data
select "performance data" section
+
select product type
select area
+
Display categorized performance data
X
select "High performer"
display production data of selected high performing farmer
select "Low performer"
display production data of low performing farmers

Figure 6: BPMN diagram of farmer performance

2.3 Actors

In this section are defined the professional figures which, according to the assignment, the system will interact with: **policy makers**, **farmers** and **agronomists**. In order to avoid redundancy, we also introduce the concept of **user** as an abstract entity that collects their common properties. As shown in figure 1, they inherit user's properties.

User

It is a person who wants to use the service. In order to access the system, it has to register to the platform (the first time) and be logged in (the following times). It also requires an Internet connection to properly use the system.

Policy maker

It is someone interested in the overall performing situation of Telangana's farmers (e.g., Telangana's government). It is able to surf on DREAM's website. It uses the service to visualise information about well and under-performing farmers and to understand if the steering initiatives are producing significant results. We assume that those who register to the platform as policy makers are some sort of "authorized" people (for example, people working for public institutions or government) since they are allowed to actively assign incentives.

Farmer

It is a farmer of Telangana. It is able to surf on DREAM's website (or to use the smartphone application). It uses the service in order to discuss with other farmers, to send requests for help to an agronomist, to insert product information, to know when it will receive a visit from an agronomist. It takes advantage in using the system because it can receive incentives if it is well-performing or can receive help if it is under-performing.

Agronomist

It is an agronomist of Telangana. It is able to surf on DREAM's website (or to use the smartphone application). It uses the service to answer farmers' requests for help, to visualise data concerning their areas (weather forecasts, well and under-performing farmers, problems encountered by farmers), to visualise and update a daily plan to visit farms in the area, to confirm the execution or specify deviation from the daily plan.

2.4 Assumptions, dependencies and constraints

In this section we present the assumptions that are expected to hold in the world, the part of the environment that the machine cannot perceive nor control. The satisfaction of these so called **domain assumptions** are deeply related to the reliability of the system to behave in the expected way. If at least one of these assumptions is discovered not to be guaranteed, then the expected behaviour of the machine would allow the occurrence of an unstable state of the machine with unpredictable results.

DX	Definition
D1	Data concerning meteorological short-term and long-term forecasts is exact and reliable
D2	Information obtained by the water irrigation system is reliable
D3	Information on the humidity of soil obtained by sensors deployed on the territory is reliable
D4	Each user who wants to use the online service (web page, app) must have a device connected to Internet
D5	The user is supposed to be at least 18 years old
D6	The third party analysis about how good a farmer is doing is reliable
D7	The categorized weather types provided by a third party analysis are reliable
D8	The weather information provided by a third party is reliable
D9	The third party analysis about the effectiveness of the steering initiatives is reliable
D10	Observation of farmers' performance is done frequently by policy makers
D11	Policy maker has enough information and knowledge to understand the proper moment to ask for good practices
D12	Outstanding high performing farmers are asked by policy makers to write their good practices periodically
D13	Incentive assignment done by policy makers is reliable
D14	Information stored in the system by farmers is reliable (e.g. farmers do not insert false data about their production or their problems to look better/worse performing)
D15	When a problem occurs, farmers insert that information in the system
D16	Farmers usually interact with the system (periodic product information upload, checking for agronomist visits)
D17	When a farmer starts a forum thread, a reply will be given as soon as possible
D18	Farmer agrees on allowing the system to store information about them (location of the activity, information on production, ...)
D19	Information inserted in the system by agronomists is reliable
D20	When an agronomist plans the visits of a daily plan, most farmers will be present
D21	Agronomists will always confirm the execution (specifying deviations, if needed) of the daily plan at the end of the same day
D22	An agronomist is responsible for several areas

Table 7: Table of domain assumptions

link to traceability matrix

3 Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

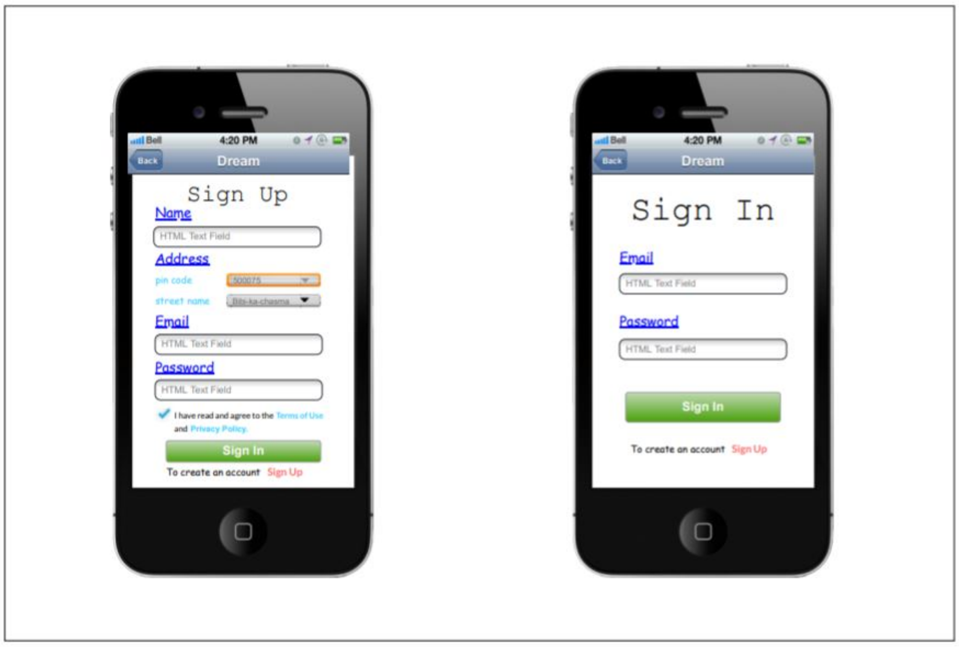
The system allows farmers to access the personalized best practice data based on their location and type of production. To analyze farmers' performance, they are required to insert production information periodically. The system provides pages where they can exchange their opinion among farmers and ask suggestions to agronomists by sending messages.

On the other hand, the system serves agronomists to let them answer the requests from farmers. Agronomists need to specify which areas are under their responsibility; then, according to the registered areas, the system organizes the visits plan and notify them. The provided plans can be modified by the agronomist if necessary, and the meetings status will be also tracked.

Lastly, for policy maker, the system provides the classification of farmer according to their performance, places with critical natural disaster. Assignment of incentives and request of writing good practices on selected farmers will be also managed. To track the effectiveness of the steering initiatives carried out by farmers and agronomists, commitments from them will be accessible.

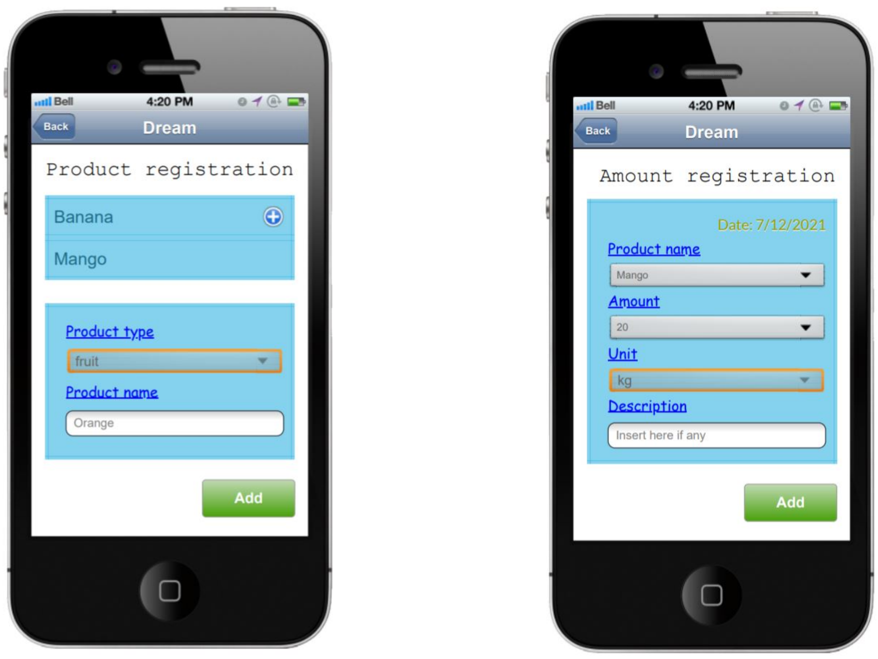
Considering their jobs, since farmers need to be outside most of the time, using smartphone application would be suitable for them by the aspect of portability. Instead agronomists and policy makers are supposed to work mainly inside; therefore, the desktop based web application will be provided.

Here we would like to introduce a few images to show how the application looks like on main functions. Figure 7 shows "sign up" and "sign in" pages, which are common for all types of user. Figure 8 serves for farmers to let them insert the production results. Lastly, figures 9 and 10 visualize the chat page between a farmer and an agronomist. We will present other mock-ups and more detailed ones in our Design Document.



4:20 PM
Back
Dream
Sign Up
Name
HTML Text Field
Address
pin code
street name
Email
Password
I have read and agree to the Terms of Use and Privacy Policy
Sign In
To create an account Sign Up
Sign In
Email
Password
Sign In
To create an account Sign Up

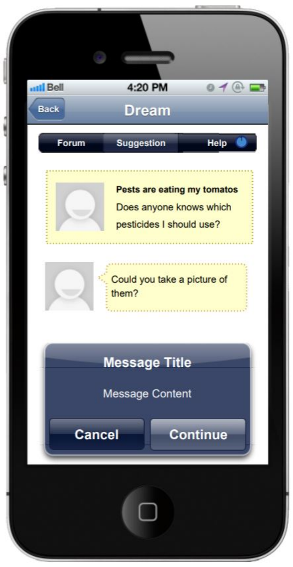
Figure 7: Sign Up, Sign In



Product type: fruit
Product name: Orange

Date: 7/12/2021
Product name: Mango
Amount: 20
Unit: kg
Description: Insert here if any

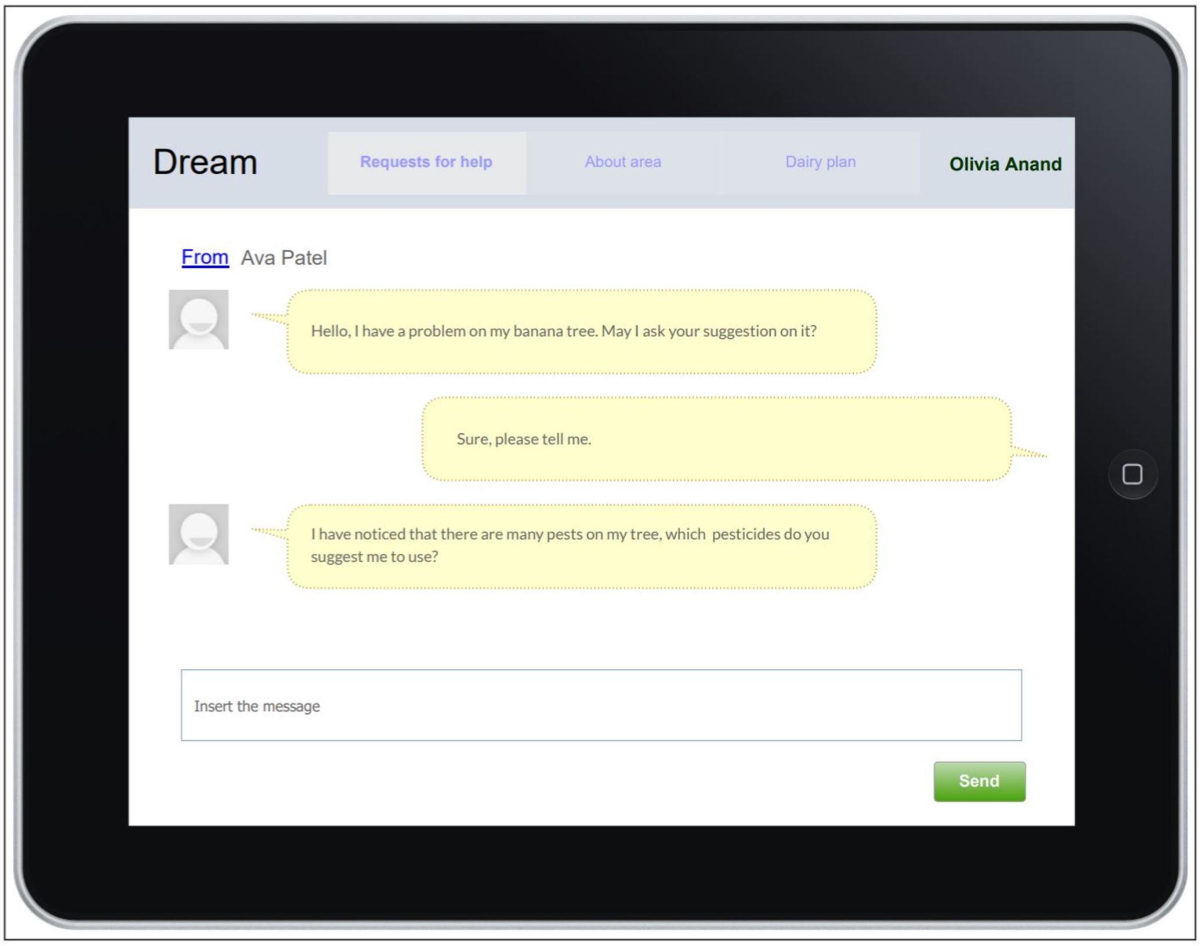
Figure 8: Product registration, Amount registration



Pests are eating my tomatos
Does anyone knows which pesticides I should use?

Could you take a picture of them?

Figure 9: Message insertion



Hello, I have a problem on my banana tree. May I ask your suggestion on it?

Sure, please tell me.

I have noticed that there are many pests on my tree, which pesticides do you suggest me to use?

Figure 10: Desktop chat

3.2 Functional Requirements

In this section we define the main **functional requirements** that the application shall offer. As well as for domain assumptions, these functionalities are fundamental to hold for the correct expected behavior of the system. However, requirements are directly perceived by the machine, that is therefore the one responsible to guarantee their accomplishment. In the same way as discussed in 2.4, if at least one of the functional requirements is not satisfied, then an unstable state of the machine can occur and lead to unexpected behaviors and results.

RX	Definition
R1	The system shall allow the user to register to the platform using a username and a password
R2	The system shall allow the policy maker to see the performance data of each farmer
R3	The system shall allow the policy maker to select the filtering
R4	The system shall show the weather data history
R5	The system shall allow the policy maker to see the information of incentives
R6	The system shall allow the policy maker to assign incentives
R7	The system shall allow the policy maker to see the production history

R8	The system shall allow the policy maker to see the commitments done by agronomists
R9	The system shall allow the policy maker to ask requests of writing good practices
R10	The system shall notify requests for writing good practices to the requested farmers
R11	When a farmer inserts information, the system shall store it
R12	The system shall allow farmer to add production information
R13	The system shall allow farmer to remove production information
R14	The system shall allow farmer to visualize production information
R15	The system shall allow farmer to insert faced problems information
R16	The system shall allow farmer to remove faced problems information
R17	The system shall allow farmer to visualize faced problems information
R18	The system shall allow farmer to edit his street address
R19	The system shall allow farmer to receive help/suggestions messages
R20	The system shall allow farmer to send help/suggestions messages
R21	The system shall allow only to farmers user to insert data about their production
R22	The system shall prevent farmer users to access other farmers data
R23	The system shall allow farmers to visualize agronomists visit information
R24	The system shall allow farmer open forum thread
R25	The system shall allow farmer reply to a forum thread
R26	The system shall allow a farmer to receive other farmer's reply on his/her forum thread
R27	The system shall allow the agronomist to receive farmers' requests
R28	The system shall allow the agronomist to answer to farmers' requests
R29	The system shall allow the agronomist to visualize data concerning weather forecast in the areas
R30	The system shall allow the agronomist to visualize the well performing and the under-performing farmers in the areas
R31	The system shall allow the agronomist to visualize the problems faced by the farmers
R32	The system shall allow the agronomist to visualize his/her daily plans
R33	The system shall allow the agronomist to update his/her daily plans
R34	The system shall allow the agronomist to confirm the execution of the current daily plan
R35	The system shall allow the agronomist to specify a deviation from the current daily plan
R36	The system shall allow the agronomist to add an area he is responsible of
R37	The system shall allow the agronomist to remove an area he is responsible of
R38	The system shall automatically prepare future daily plans for the agronomist, based on the number of visits received by each farmer and the farmers' performing status

Table 8: Table of requirements

Each requirement is used in the **traceability matrix** in section 3.2.5. In the following subsections, for each actor, we introduce a set of possible scenarios of an ideal future where DREAM has already been developed and deployed in the state of Telangana and some of the most relevant use cases of the S2B. The use cases sections are composed firstly by the relative use case diagram, then the main use case tables and sequence diagrams are alternately presented.

3.2.1 Users

Use case tables

ID	U.1
NAME	Sign up User with email
ACTOR	User
ENTRY CONDITION	User has opened the Web page OR User has downloaded and opened the application on his smartphone
INPUT	Email to use for the registration
EVENTS FLOW	• The system displays the "Sign in" page • User clicks on "Sign up" • The system displays two fields: email and password • User inserts the data and accepts the "Terms of services" • User clicks on the "Confirm" button • The system displays the acceptance of the registration and invites User to go to his inbox in order to confirm the registration • User opens his inbox, checks the email and clicks on the confirmation link
EXIT CONDITION	User registration has been successful: user data are stored in the system's database. User can now login with his credentials
OUTPUT	• User's email is stored in the system's database • User receives the confirmation email
EXCEPTIONS	• User inserts an email which is already stored in the database. So, after User clicks on "Confirm", the system displays an error page which tells that User is already registered to the service and invites him to login with that email • User inserts an invalid email. So, after User clicks on "Confirm", the system displays the same sign up page with an error message, which suggests User to check the inserted email or to change it

Table 9: Sign up User

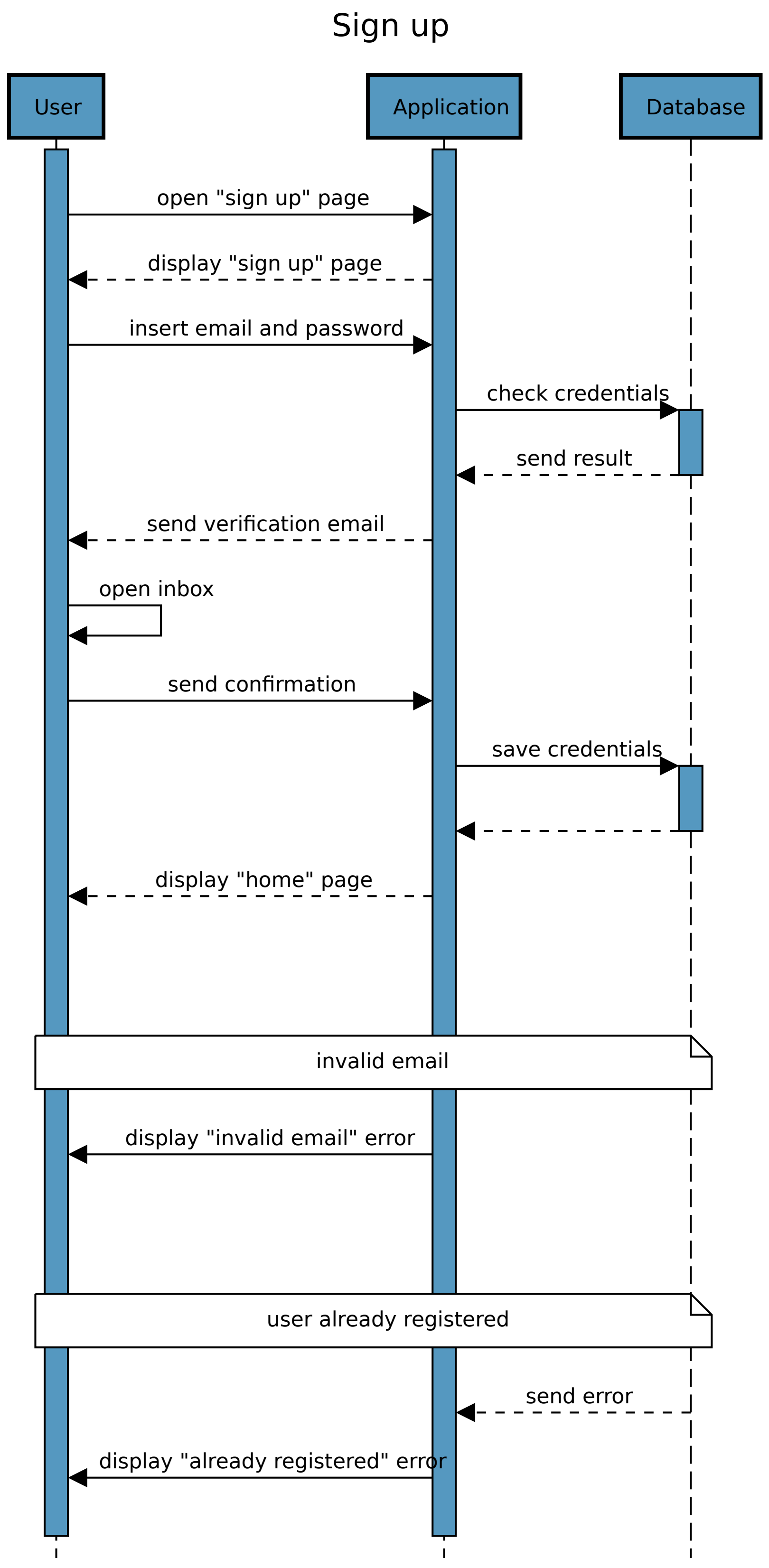


Figure 11: Sign up User - sequence diagram

ID	U.2
NAME	Login User
ACTOR	User
ENTRY CONDITION	User has opened the Web page OR User has downloaded and opened the application on his smartphone
INPUT	User's valid email and password
EVENTS FLOW	• The system displays the "Login" page • User inserts his credentials (email, password) and clicks the "Login" button • The system checks the correctness of the inserted credentials
EXIT CONDITION	The system displays the home page
OUTPUT	• User is logged in
EXCEPTIONS	• User inserts a wrong combination of email and password. The system displays the same page with an error message • User inserts an email which is already stored in the database. So, after User clicks on "Confirm", the system displays an error page which tells that User is already registered to the service and invites him to login with that email • User inserts an invalid email. So, after User clicks on "Confirm", the system displays the same sign up page with an error message, which suggests User to check the inserted email or to change it

Table 10: Login User

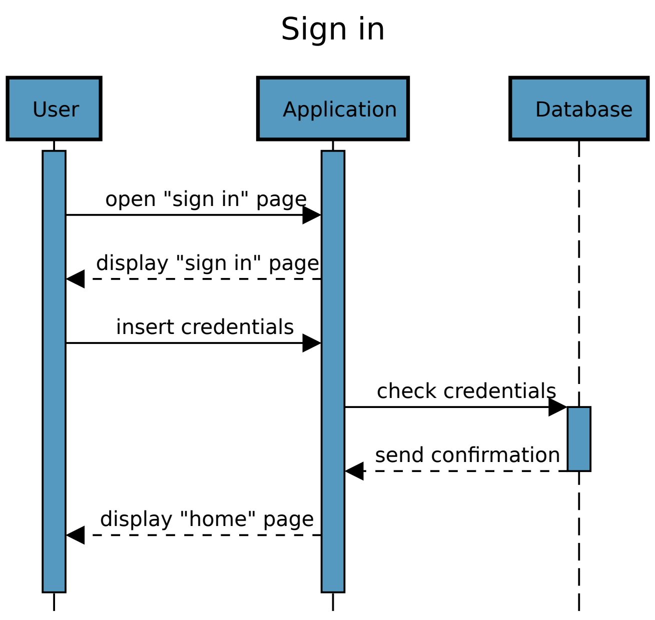


Figure 12: Login User - sequence diagram

3.2.2 Policy makers

Policy maker scenarios

Scenario 1

Kiara Kumar is a policy maker who works in the Indian government. She has a task to make sure that all the positively evaluated farmers will be gain the incentives. She opens her working PC and accesses to the DREAM app to check the farmers' performance. By looking at the fetched high-performing farmers, she finds the ones who are missing incentives. According to their performance behaviour, she select which e-voucher to assign to each farmer.

Scenario 2

Sai Devi is a policy maker who works in the Indian United Nations. He was told by his manager that he needs to make an assumption about why the production outcomes in Nalgonda are having issues in recent months. To have ideas of the core issues, he accesses to DREAM app to compare the behavior of low performing farmers against high performing farmers. Firstly, he browses several plots which highlight the period when the temperature was particularly higher for consecutive days. Secondly, he observed the history of watering done by irrigation systems. He discovered that low performing farmers didn't change the amount of watering for their crops in that period, therefore he concluded that the issue is caused by the missing adjustment of irrigation systems in critical situations.

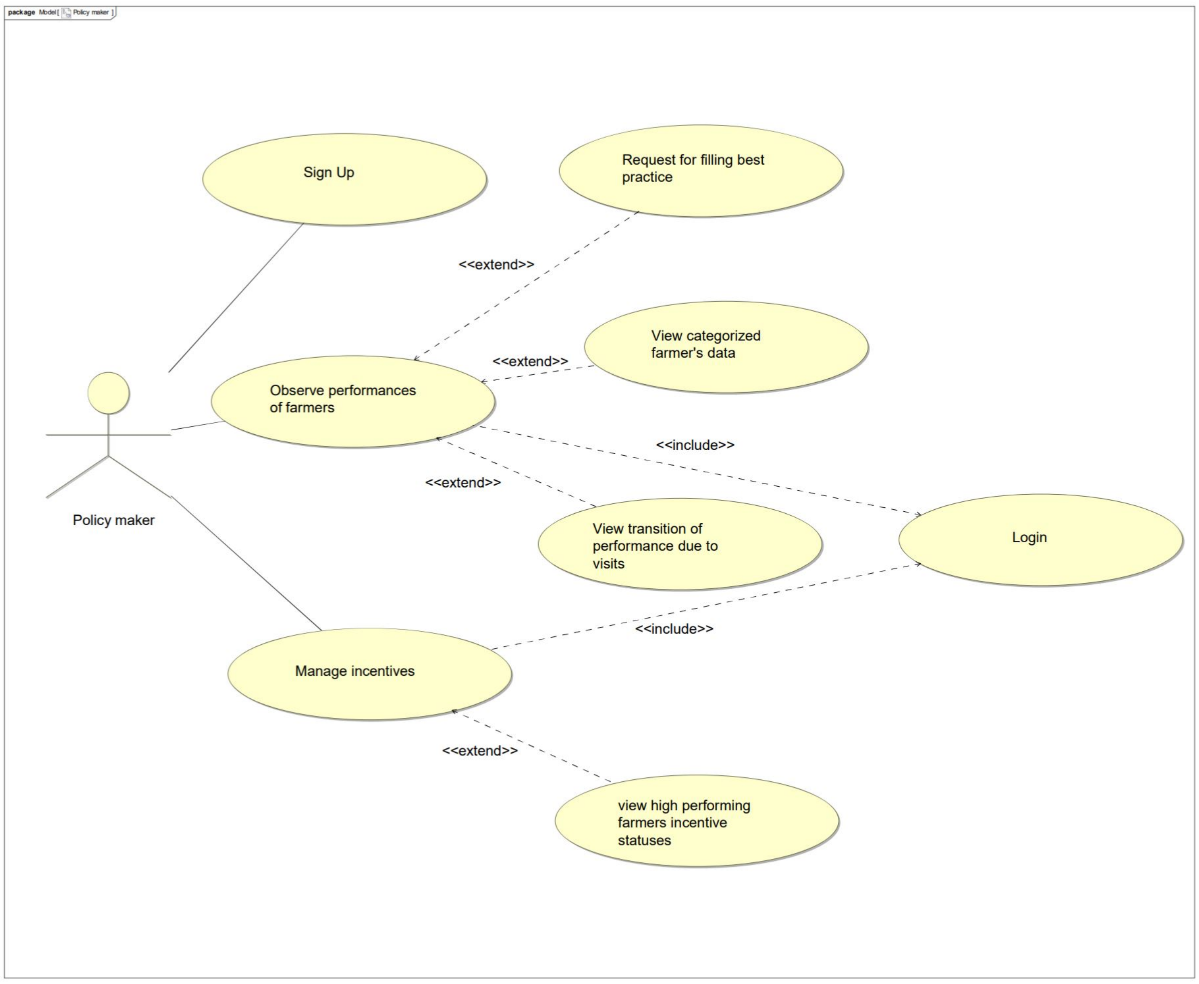
Scenario 3

Rudra Singh is a policy maker who works in the Indian government. She is the manager of her department and she needs to pick up exceptionally high performing farmers to ask them to write their best practices to standardize good behaviours with the rest of the farmers. As a first step, she checks low performing farmers' average behaviour on the aspects they are toughly struggling with and defines it as a domain. Then she navigates to visualize the high performing farmers list, which contains details about the production. She fetches a farmer who has high performance in that critical domain. Lastly, she operates the registration of the request form to ask them to provide good practices by filling the needed information. It will shortly notify the farmers.

Scenario 4

Saanvi Das is a policy maker who works in the Indian United Nations. Today's task is to make a report of the agronomists' visits outcome. For completing this assignment, she wants to gather the information of the performance variation due to the visits. She navigates the page, which contains several domains such as name of agronomist, type of product and area. She selects the filters and fetches the interested data. The system displays the results as plots. By looking at them, she writes down the insights for sharing her discovery to her colleagues in the next meeting.

Policy maker use cases



package Model[Policy maker]
Sign Up
Request for filling best practice
<<extend>>
View categorized farmer's data
<<extend>>
Observe performances of farmers
<<include>>
<<extend>>
Policy maker
View transition of performance due to visits
Login
<<include>>
Manage incentives
<<extend>>
view high performing farmers incentive statuses

Figure 13: Policy maker use case diagram

Use case tables

ID	PM.1
NAME	Visualize the performance data of each farmer
ACTOR	Policy maker
ENTRY CONDITION	Policy maker has logged in
EVENTS FLOW	• Policy maker goes to the “Farmer’s performance” section • The system displays options to let user select filters about weather type, product type • Policy maker selects desired filters • The system displays the list of farmers divided by performance • Policy maker selects interested farmer’s name • The system shows detailed information about that farmer’s production
EXIT CONDITION	The system returns to the main page of policy maker
OUTPUT	• Policy maker has obtained the farmer’s production data they were looking for
EXCEPTION	Policy maker could not find the name of farmer who should exist. The system displays an error message

Table 11: Visualize the performance data of farmers

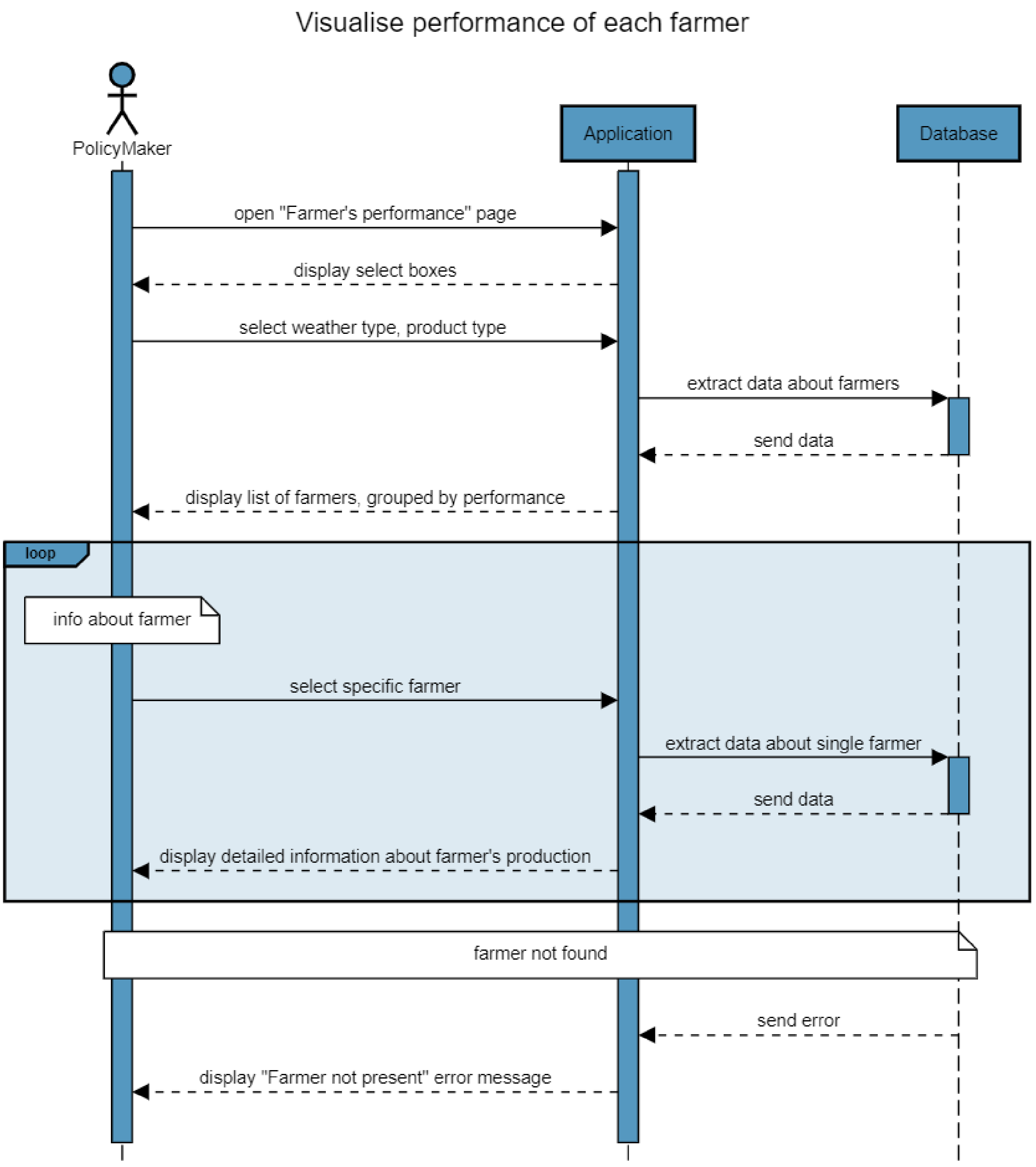

Visualise performance of each farmer
PolicyMaker
Application
Database
open "Farmer's performance" page
display select boxes
select weather type, product type
extract data about farmers
send data
display list of farmers, grouped by performance
loop
info about farmer
select specific farmer
extract data about single farmer
send data
display detailed information about farmer's production
farmer not found
send error
display "Farmer not present" error message

Figure 14: Visualize the performance data of farmers - sequence diagram

ID	PM.2
NAME	Manage the incentives of high performing farmers
ACTOR	Policy maker
ENTRY CONDITION	Policy maker has logged in
EVENTS FLOW	• Policy maker goes to the “Farmer’s performance” section • The system displays a page with the list of farmers (grouped by performance) and a column which clarifies the status of their incentives • Policy maker selects the interested farmer’s incentive column • The system shows detailed information about farmer’s incentives status • Policy maker clicks "Select incentive" button • The system shows the possible choices • Policy maker selects the incentive to give • The system shows the popup to ask a confirmation to proceed • Policy maker clicks "Confirm" button • The system shows the updated incentive column
EXIT CONDITION	The system returns to the main page of policy maker
OUTPUT	• Policy maker has managed the incentive to give • The farmer correctly received the inventive
EXCEPTION	Policy maker wrongly selects an incentive data. The system shows a popup to ask a confirmation to proceed. The policy maker can redo the operation by clicking on “Cancel” button

Table 12: Manage the incentives of farmers

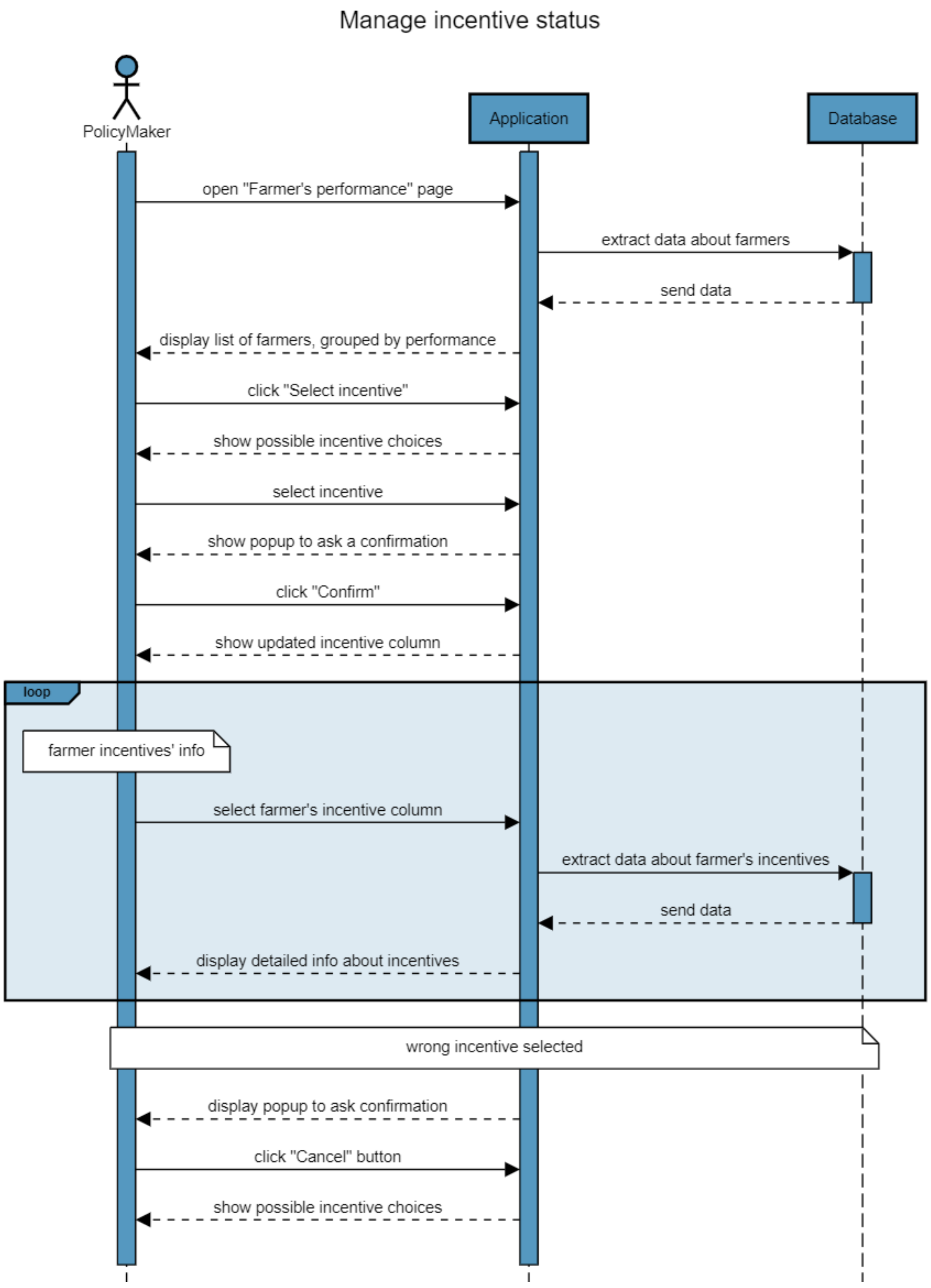

Manage incentive status
PolicyMaker
Application
Database
open "Farmer's performance" page
extract data about farmers
send data
display list of farmers, grouped by performance
click "Select incentive"
show possible incentive choices
select incentive
show popup to ask a confirmation
click "Confirm"
show updated incentive column
loop
farmer incentives' info
select farmer's incentive column
extract data about farmer's incentives
send data
display detailed info about incentives
wrong incentive selected
display popup to ask confirmation
click "Cancel" button
show possible incentive choices

Figure 15: Manage the incentives of farmers - sequence diagram

ID	PM.3
NAME	Visualize the effectiveness of the steering initiatives
ACTOR	Policy maker
ENTRY CONDITION	Policy maker has logged in
EVENTS FLOW	• Policy maker goes to the "Effectiveness of initiatives" section • The system displays options to let user select filters about weather type, product type • Policy maker selects the desired filters • The system displays a page with categorized farmers, highlighting those who have improved their performance significantly within a year • Policy maker selects interested farmer's name • The system shows detailed information about the farmer's performance trend, the problems encountered, the history of requests and visits
EXIT CONDITIONS	The system returns to the main page of policy maker
OUTPUT	• Policy maker has obtained the history of performance data and interaction between farmers and agronomists they were looking for
EXCEPTION	Policy maker couldn't find the name of farmer who should exist. The system displays an error message

Table 13: Visualize the effectiveness of the steering initiatives

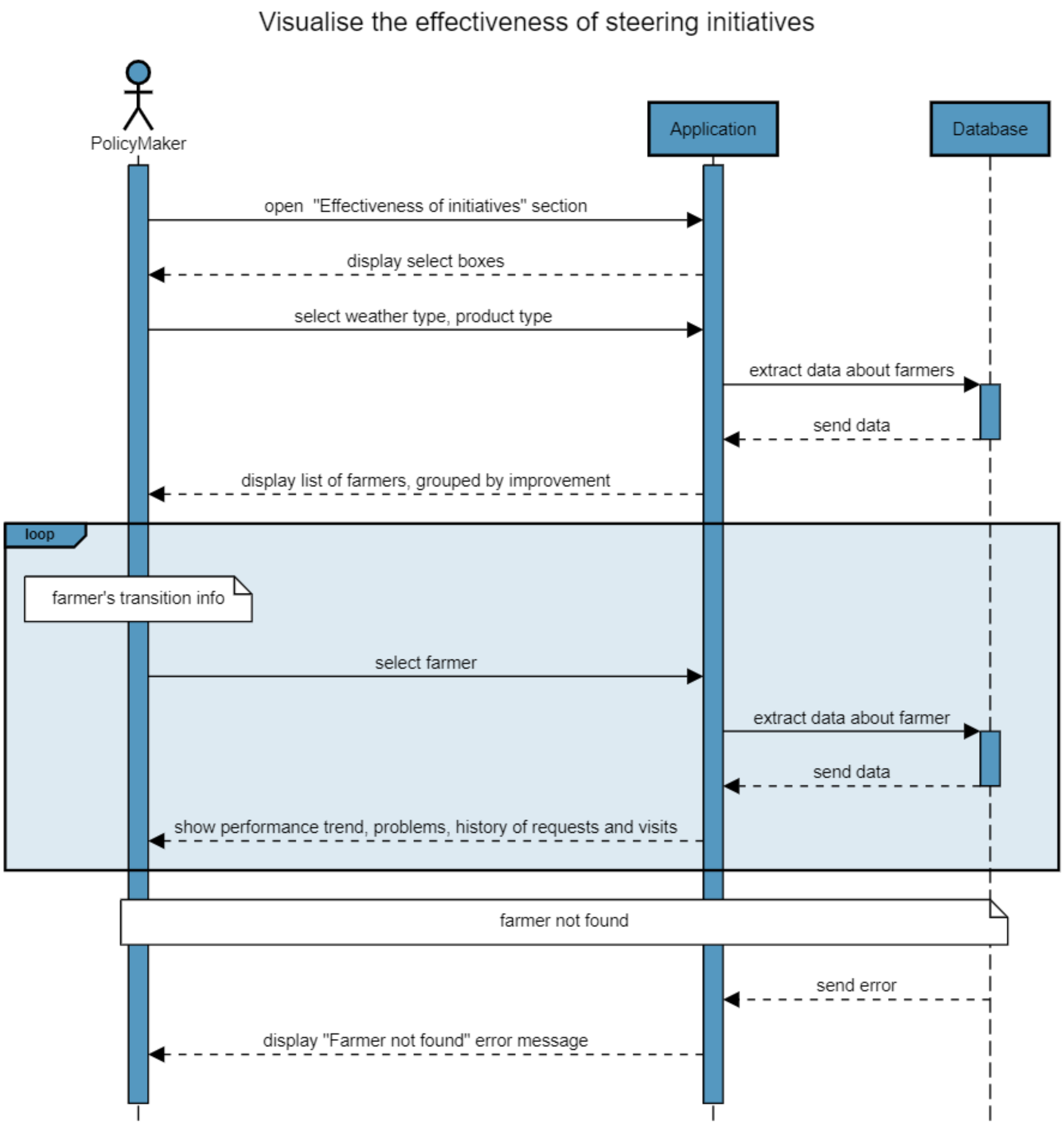

Visualise the effectiveness of steering initiatives
PolicyMaker
Application
Database
open "Effectiveness of initiatives" section
display select boxes
select weather type, product type
extract data about farmers
send data
display list of farmers, grouped by improvement
loop
farmer's transition info
select farmer
extract data about farmer
send data
show performance trend, problems, history of requests and visits
farmer not found
send error
display "Farmer not found" error message

Figure 16: Visualize the effectiveness of the steering initiatives - sequence diagram

ID	PM.4
NAME	Ask high performing farmers to write good practices
ACTOR	Policy maker
ENTRY CONDITION	Policy maker has logged in
EVENTS FLOW	• Policy maker goes to the “Farmer’s performance” section • The system displays a page with the list of farmers grouped by performance • Policy maker selects interested high performing farmer’s name • The system shows the detailed information about farmer’s production and "request writing" button • Policy maker clicks "Request writing" button • The system shows a popup to ask the confirmation to proceed • Policy maker clicks "Confirm" button • The system sends the request to the selected farmer and shows a message to notify the success of the operation
EXIT CONDITION	The system returns to the Farmer’s performance page
OUTPUT	• Policy maker has requested the high performing farmer to write their good practice • Farmer has received the good practice request
EXCEPTION	Policy maker wrongly selects a farmer. The system shows a popup to ask a confirmation to proceed. The policy maker can redo the operation by clicking on “Cancel” button

Table 14: Ask a farmer to write good practices

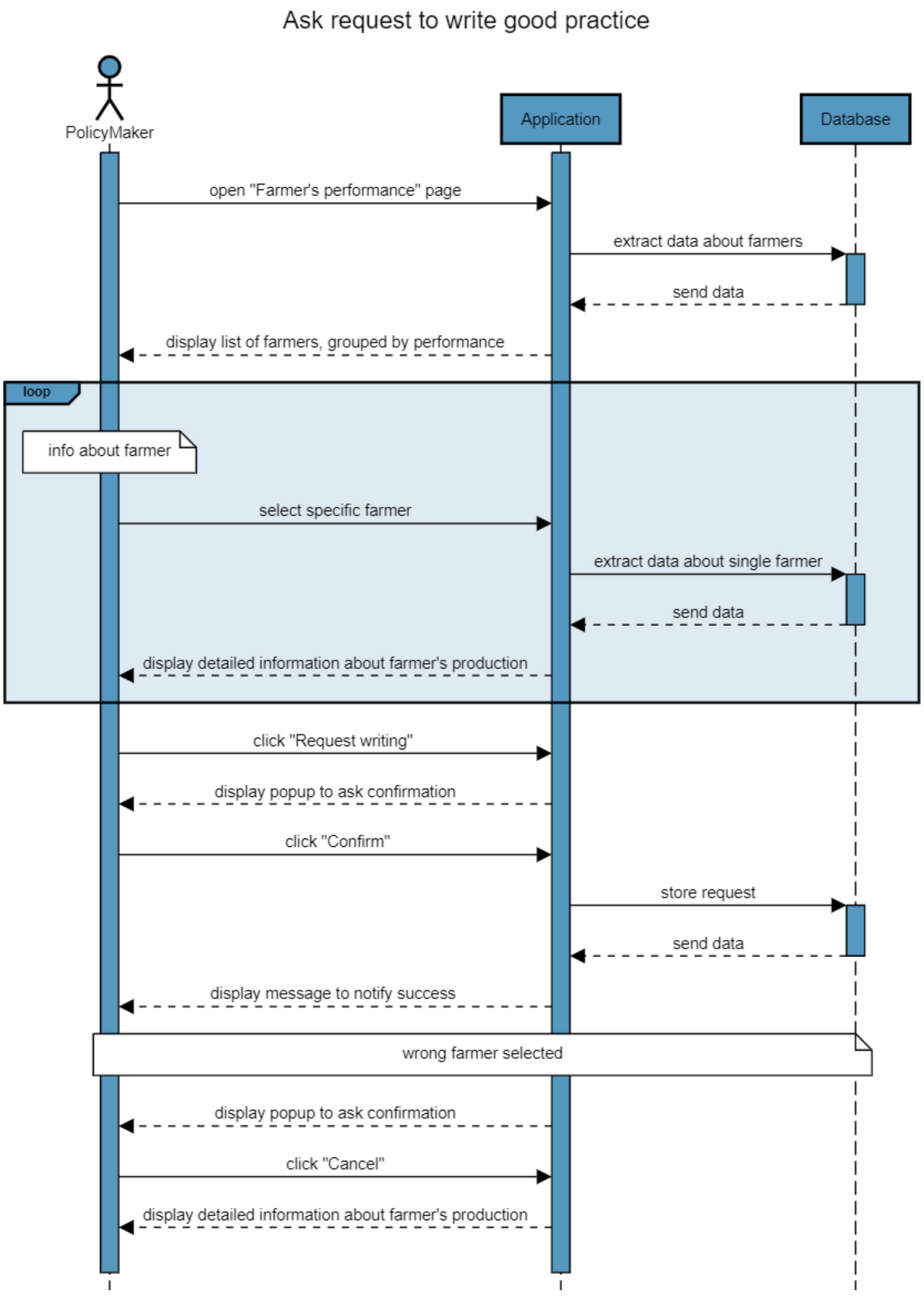

Ask request to write good practice
PolicyMaker
Application
Database
open "Farmer's performance" page
extract data about farmers
send data
display list of farmers, grouped by performance
loop
info about farmer
select specific farmer
extract data about single farmer
send data
display detailed information about farmer's production
click "Request writing"
display popup to ask confirmation
click "Confirm"
store request
send data
display message to notify success
wrong farmer selected
display popup to ask confirmation
click "Cancel"
display detailed information about farmer's production

Figure 17: Ask request to write good practice - sequence diagram

3.2.3 Farmers

Farmer scenarios

Scenario 1

A farmer named Mishka wakes up early for working and, while she's having breakfast, she remembers about an app one of his farmer friends suggested her. She decides to take a try, but she finished her internal smartphone memory storage so she can't install the application. However, she knows she can also access the platform via Internet. She therefore opens her mobile browser and looks for DREAM website and signs up. She takes a look to the recent forum threads and reads about a new agricultural technique. She decides to apply the technical suggestion: she closes her smartphone and starts working. At the end of the day, while having dinner she logs in again to the web application through her PC. She decides to insert her last week's production information and sees an on-app notification: the system informs X that an agronomist, specialized in Y, scheduled a visit next month. She is now aware about the event. Furthermore, she notices that the appointment has also been saved in the calendar section.

Scenario 2

A farmer named Dhakshan starts his working day. He is already a loyal user of DREAM application and all the mornings he takes a look to his relevant information section through his smartphone. He looks at the daily suggestion of the system: from the next days until the end of the month temperatures are reaching $39°C$, it's suggested to .... He follows the suggestion as usual and submits his usual production information, maybe that's the reason he is considered a good farmer. At the end of the day he receives a request for submitting a good practice document. He decides to accept: he fills the fields and writes a short description of his usual *modus operandi*.

Scenario 3

Pranit is a farmer and also a new user of the DREAM platform, in fact he recently installed it on his smartphone. He has already been assigned by the application to the agronomists and he has previously received by them their visit schedule. One of them is going to visit him today, but Pranit forgot it. Luckily the application notifies him about the visit, preventing him to miss the meeting.

Scenario 4

A farmer named Benjamin is going great with his activity. One day somehow a part of his plantation shows signs of disease, introducing the risk of infection in other parts of the same plantation. He immediately signs in to the DREAM platform and inserts in the system the problem he's facing. Meanwhile, the agronomist specialized in plant diseases planned his visit to Benjamin's property in 5 months from today. After the farmer submitted information about the problem he's facing, the agronomist chose to change his plan anticipating his visit to Benjamin's activity: the meeting will now occur within few days and the update is forwarded to the related farmers.

Scenario 5

A farmer named Siya is determined to improve her activity production rate and is curious to know other farmers' modus operandi, chasing for something new to discover about agriculture literature. She therefore logs in DREAM web application through her tablet and opens a Forum thread entitled "What is the main good practice to follow as first step in order to increase the production rate?". At the same time Siya also texts a suggestion request addressed to both few agronomists and two of her farmer friends. After few hours the farmer receives some replies by both the Forum thread and the private chat she created, making her able to learn new stuff and potentially making some new friend.

Farmer use cases

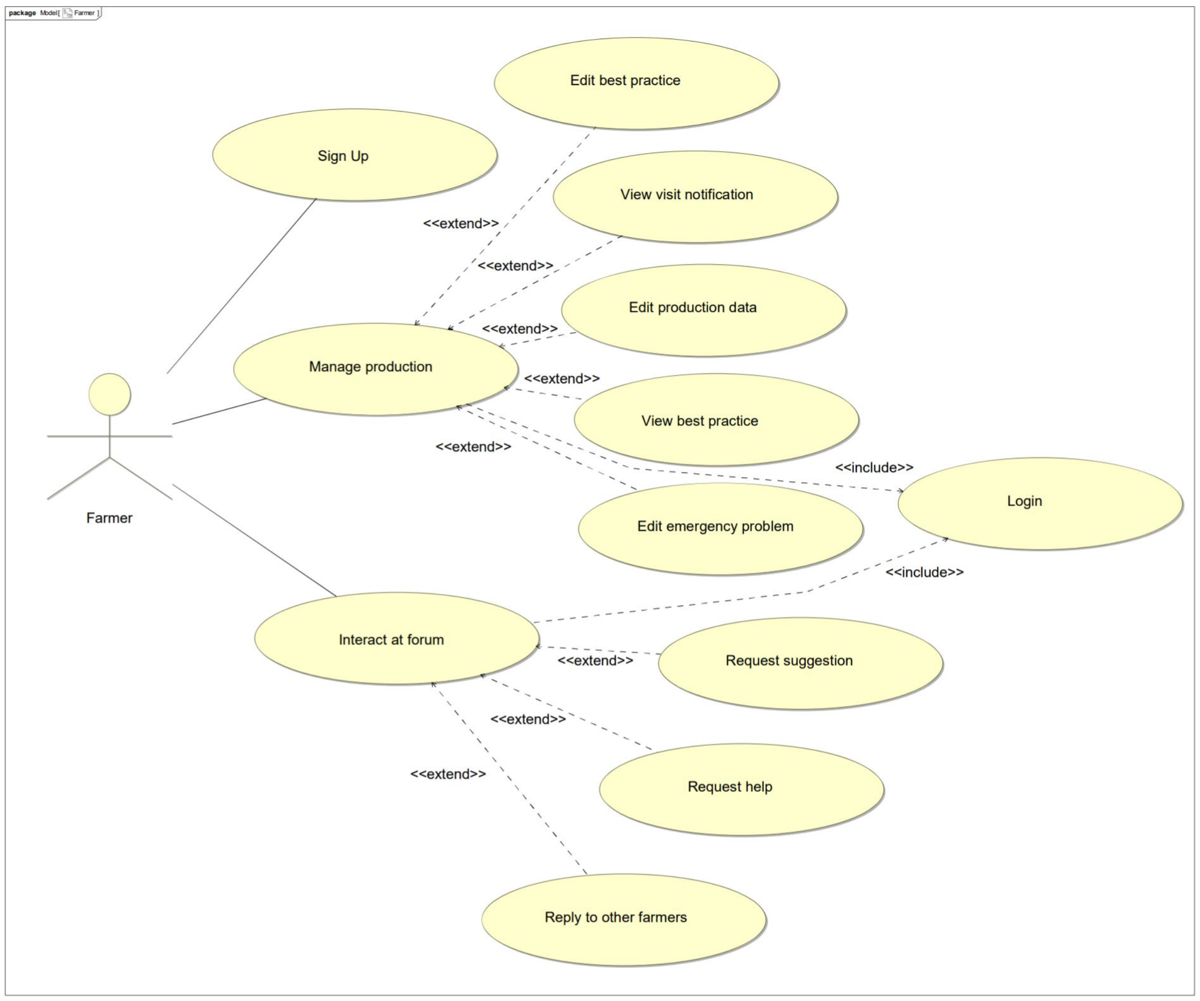
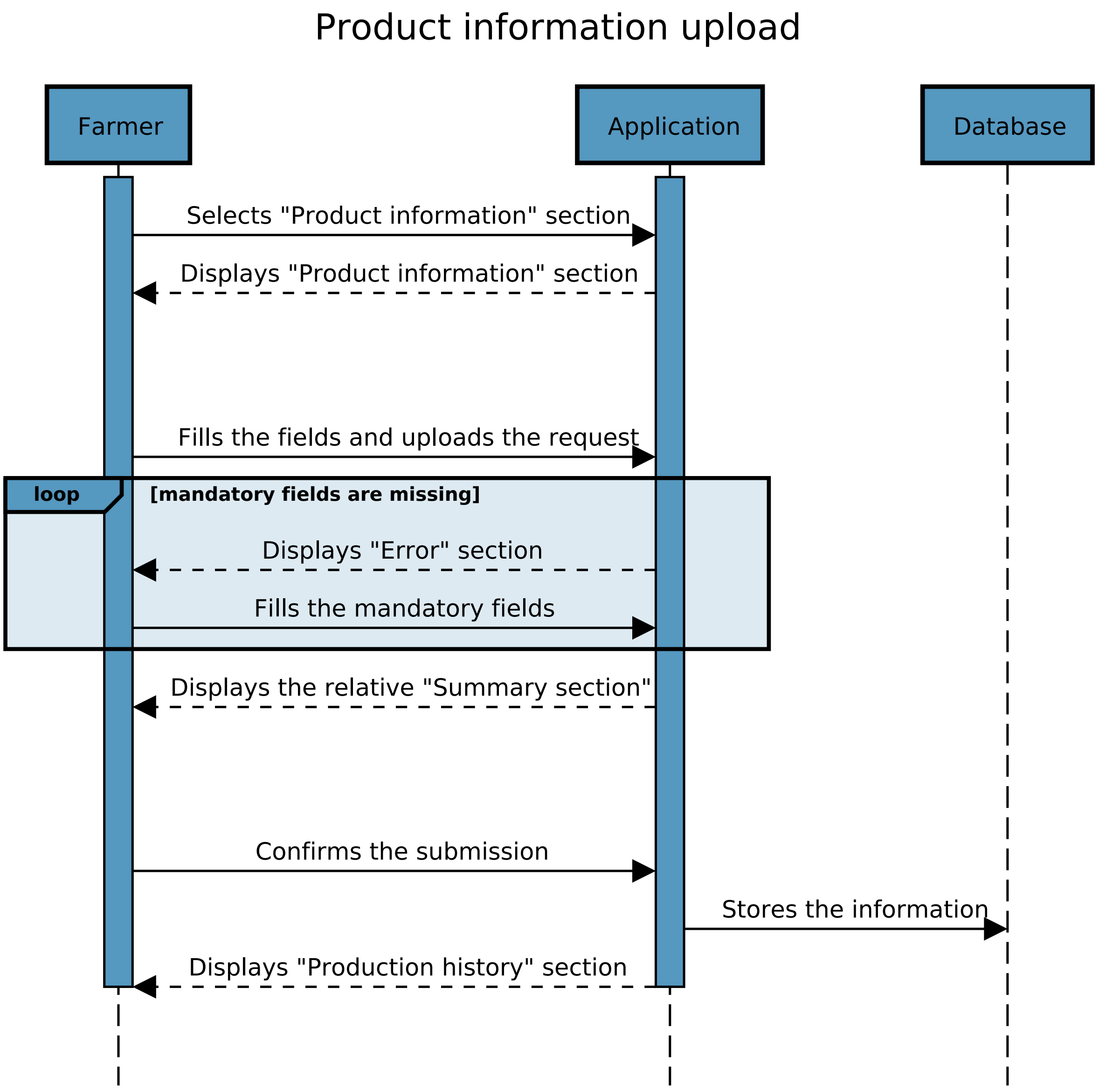


Figure 18: Farmer use case diagram

Use case tables

ID	F.1
NAME	Production data upload
ACTOR	Farmer
ENTRY CONDITION	Farmer has logged in
EVENTS FLOW	• Farmer goes to the *Record Production Data* section • The application displays a section that asks for product information and an *Upload Button* • Farmer fills the mandatory fields of the current section and eventually the optional ones. Then press the *Upload Button*. • The application displays a confirm popup revealing the summary of the information is going to be recorded, asking for Farmer confirmation through a *Confirm Button* • The farmer confirms the submission by selecting the Confirm Button
EXIT CONDITION	The application displays the summary page of both already uploaded product information and the previous submitted ones
OUTPUT	• The system collects the new production data • The farmer can visualize the list of both current production information and the previous ones
EXCEPTION	Farmer submits production data without filling the mandatory fields. In such case, the system displays an error message informing the Farmer about the missing field(s) required in order to achieve the goal

Table 15: Production data upload



Product information upload
Farmer
Application
Database
Selects "Product information" section
Displays "Product information" section
Fills the fields and uploads the request
loop
[mandatory fields are missing]
Displays "Error" section
Fills the mandatory fields
Displays the relative "Summary section"
Confirms the submission
Stores the information
Displays "Production history" section

Figure 19: Production data upload - sequence diagram

ID	F.2
NAME	Help/suggestion request
ACTOR	Farmer
ENTRY CONDITION	Farmer has logged in
EVENTS FLOW	• Farmer selects the *requests section* • The application displays a section that presents an eventual list of previous request chats and a *New request* button • Farmer clicks on the *New request* button • The application displays a new section asking for the selection of the receivers • The farmer selects the contact he wants to send the request • The system displays a section with an editable text form, asks for the selection of the topic of the request and displays a *Send request* button • The farmer writes the request, fills the topic form and clicks on the *Send request* button
EXIT CONDITION	The application displays the summary page of both already sent request chat and the previous submitted ones
OUTPUT	• The system collects the new request chat • The farmer can visualize the list of both current request chat and the previous ones
EXCEPTION	Farmer clicks on the request button without editing the text form. In such case, the system displays an error message informing the Farmer about the missing field required in order to achieve the goal

Table 16: Help/suggestion request

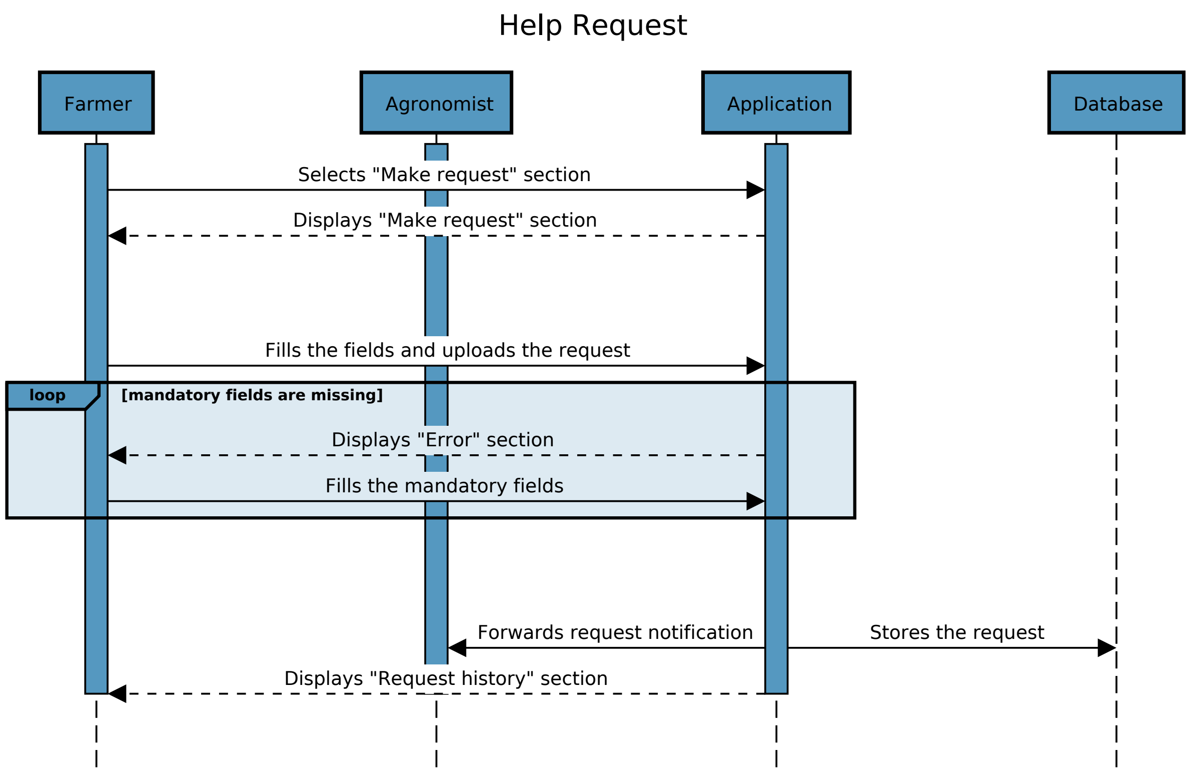

Help Request
Farmer
Agronomist
Application
Database
Selects "Make request" section
Displays "Make request" section
Fills the fields and uploads the request
loop
[mandatory fields are missing]
Displays "Error" section
Fills the mandatory fields
Forwards request notification
Stores the request
Displays "Request history" section

Figure 20: Help request - sequence diagram

ID	F.3
NAME	Forum generation
ACTOR	Farmer
ENTRY CONDITION	Farmer has logged in
EVENTS FLOW	• Farmer selects the *Forum section* • The application displays a section that presents an eventual list that contains both previous submitted forums and the ones farmer replied, and an *Add forum* button • Farmer selects the *Add forum* button • The application displays a new section that present 3 mandatory fields (the topic/context of the thread, the title and the question content) and a *Submit Forum* button • The farmer fills all the mandatory fields and presses the *Submit Forum* button • The application displays a confirm popup revealing the summary of the information is going to be uploaded, asking for Farmer confirmation through a *Confirm* button • The farmer confirms the submission by pressing the *Confirm* Button
EXIT CONDITION	The application displays the summary page containing both already submitted forum thread, the previous submitted ones and the ones to whom the farmer replied
OUTPUT	• The system collects the new forum thread • The farmer can visualize the list containing both current forum thread, the previous submitted ones and the ones to whom the farmer replied
EXCEPTION	Farmer submits Forum thread without filling all the mandatory fields. In such case, the system displays an error message informing the Farmer about the missing field(s) required in order to achieve the goal

Table 17: Forum generation

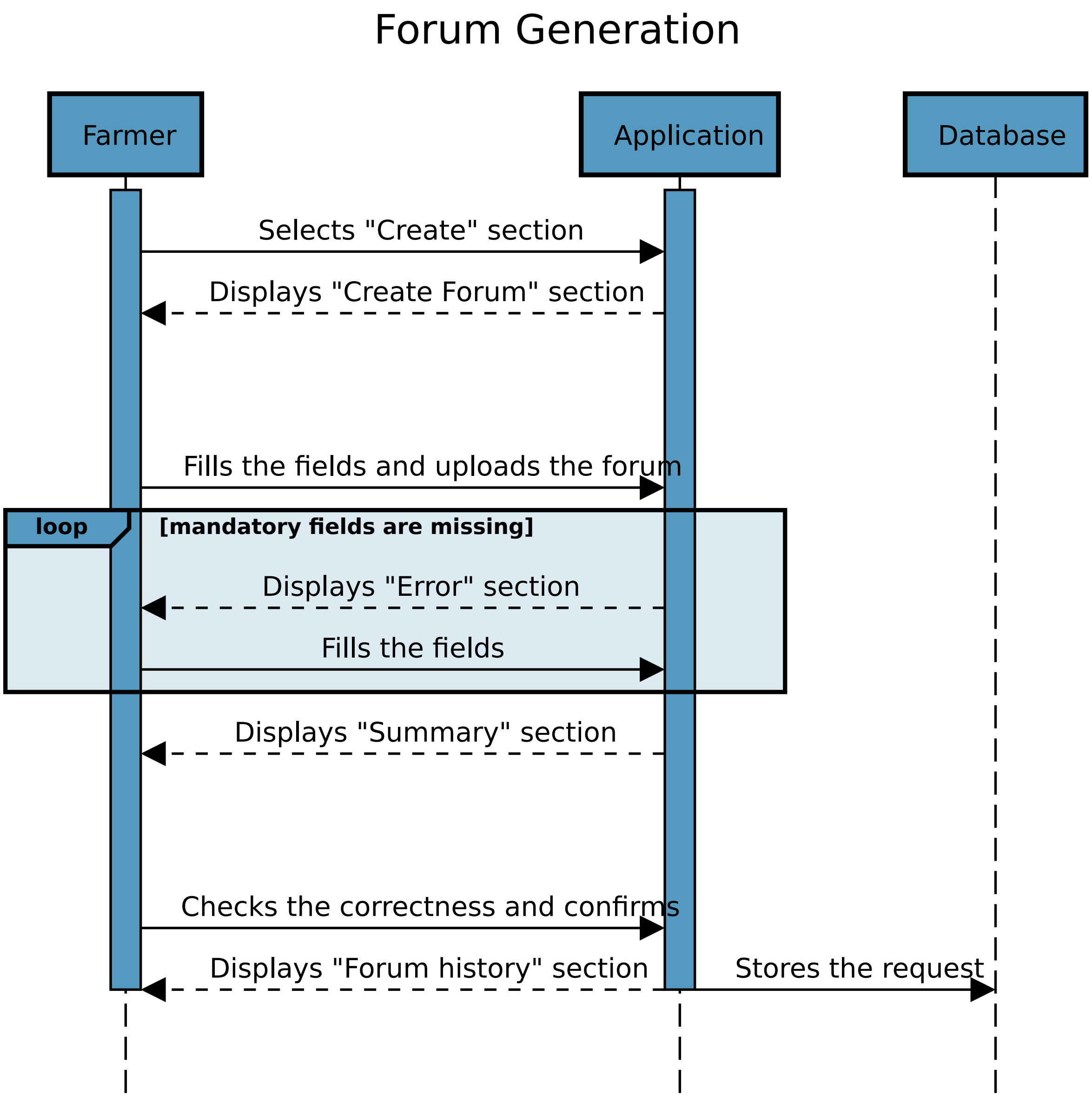

Forum Generation
Farmer
Application
Database
Selects "Create" section
Displays "Create Forum" section
Fills the fields and uploads the forum
loop
[mandatory fields are missing]
Displays "Error" section
Fills the fields
Displays "Summary" section
Checks the correctness and confirms
Displays "Forum history" section
Stores the request

Figure 21: Forum generation - sequence diagram

ID	F.4
NAME	Problem information upload
ACTOR	Farmer
ENTRY CONDITION	Farmer has logged in
EVENTS FLOW	• Farmer selects the graphical section responsible for the problem information submission, called informally *Problems section* • The application displays a section that requires for information (described previously) and a "submit button" • Farmer fills the mandatory fields and eventually the optional ones, then selects the upload button • The application displays a confirm popup revealing the summary of the information is going to be uploaded, asking for Farmer confirmation through a *Confirm Button* • The farmer confirms the submission by selecting the Confirm Button
EXIT CONDITION	The application displays the summary page containing both already submitted problem information and the previous submitted ones
OUTPUT	• The system collects the new problem information instance • The farmer can visualize the list containing both the current submitted problem information and the previous submitted ones
EXCEPTION	Farmer submits problem information without filling all the mandatory fields. In such case, the system displays an error message informing the Farmer about the missing field(s) required in order to achieve the goal

Table 18: Problem information upload

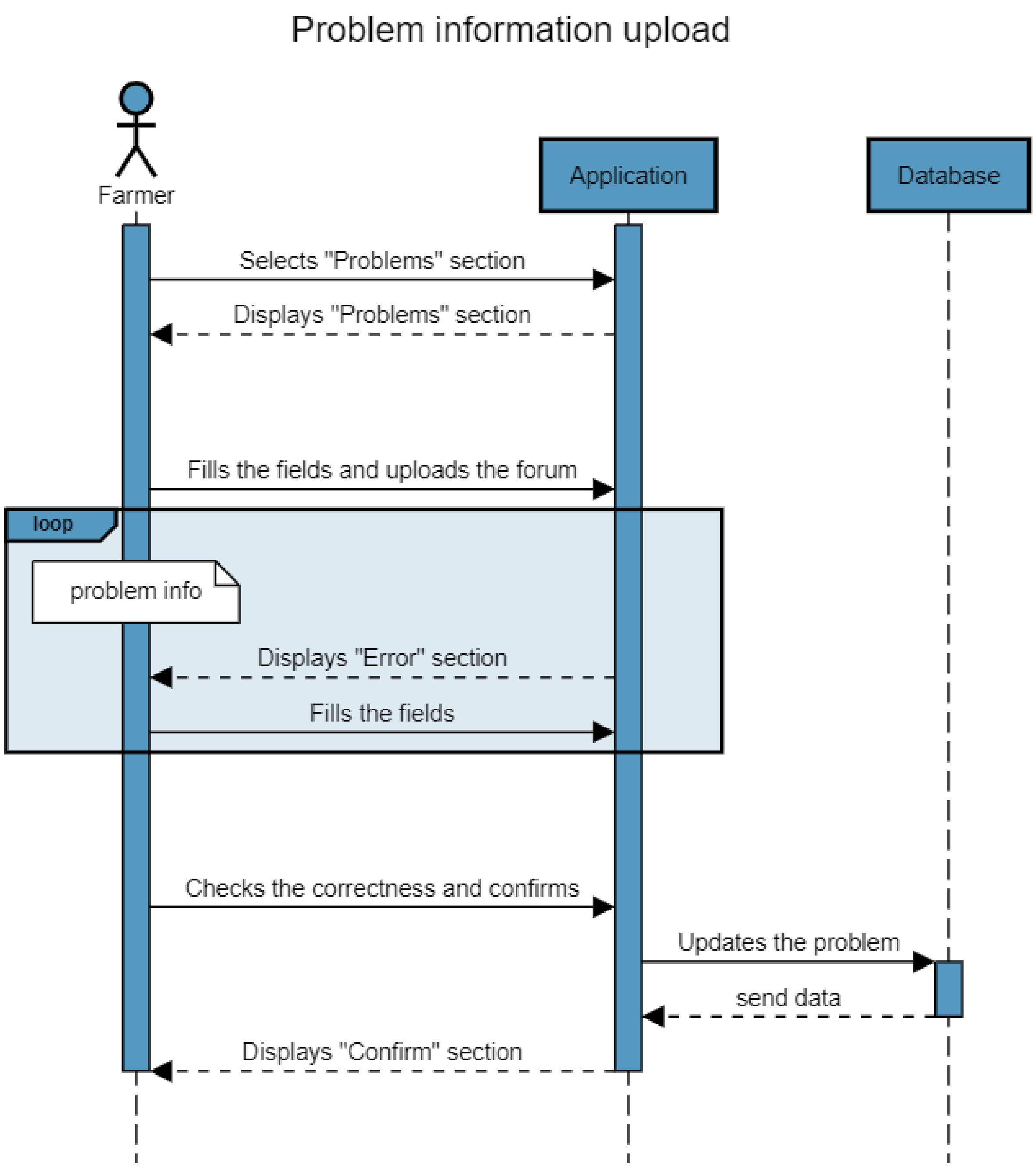
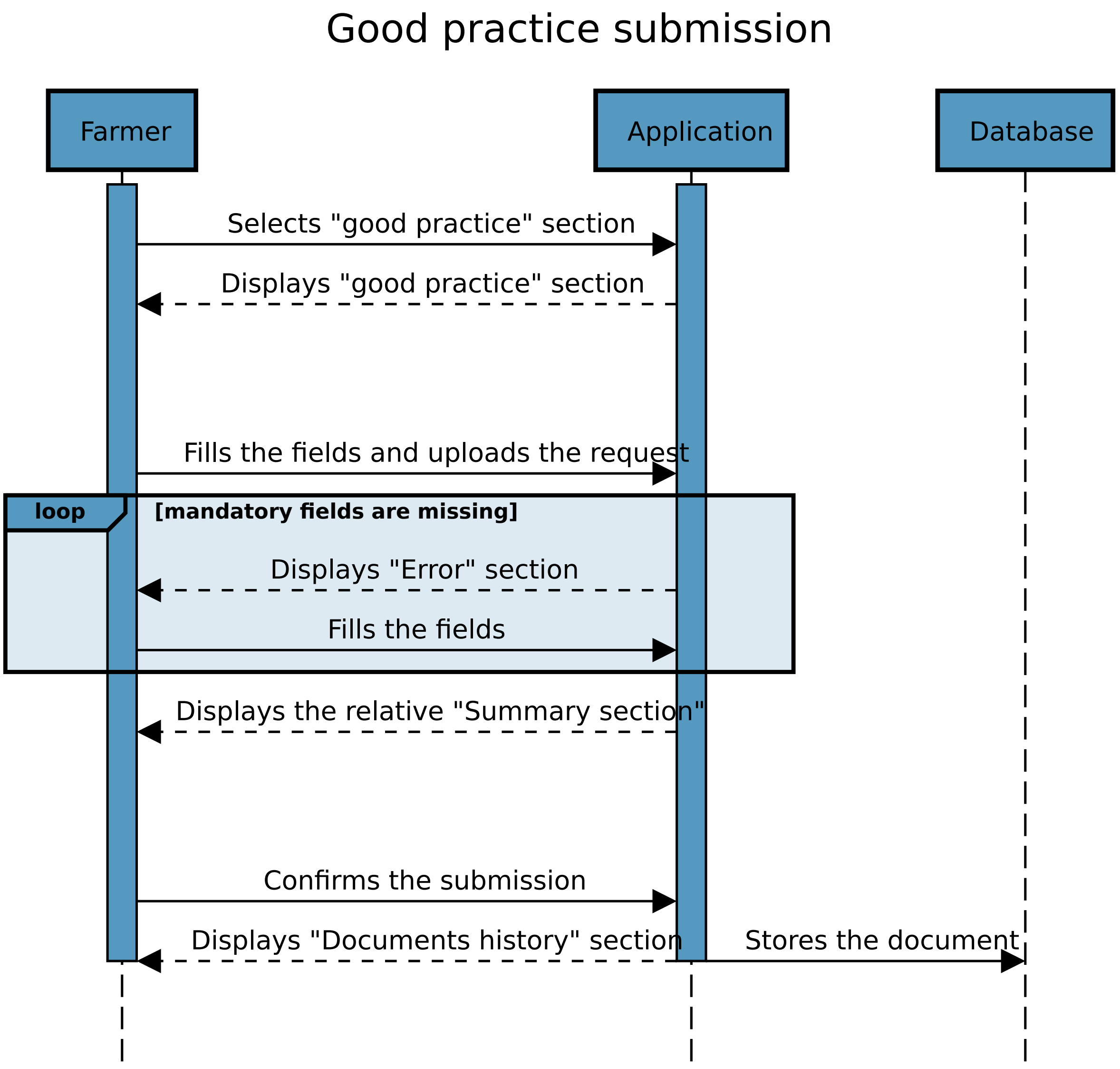

Problem information upload
Farmer
Application
Database
Selects "Problems" section
Displays "Problems" section
Fills the fields and uploads the forum
loop
problem info
Displays "Error" section
Fills the fields
Checks the correctness and confirms
Updates the problem
send data
Displays "Confirm" section

Figure 22: Problem information upload - sequence diagram

ID	F.5
NAME	Good practices upload
ACTOR	Farmer
ENTRY CONDITION	Farmer has logged in
EVENTS FLOW	• Farmer selects the graphical section responsible for the good practices document submission, called informally *document section* • The application displays a section that requires for information (described previously) and a "submit button" • Farmer fills the mandatory fields and eventually the optional ones, then selects the upload button • The farmer confirms the submission by selecting the Confirm Button
EXIT CONDITION	The application displays the summary page containing both already submitted document and the previous submitted ones
OUTPUT	• The system collects the new document • The farmer can visualize the list containing both the current submitted document and the previous submitted ones
EXCEPTION	Farmer submits document without filling all the mandatory fields. In such case, the system displays an error message informing the Farmer about the missing field(s) required in order to achieve the goal

Table 19: Good practices upload



Good practice submission
Farmer
Application
Database
Selects "good practice" section
Displays "good practice" section
Fills the fields and uploads the request
loop
[mandatory fields are missing]
Displays "Error" section
Fills the fields
Displays the relative "Summary section"
Confirms the submission
Displays "Documents history" section
Stores the document

Figure 23: Good practices upload - sequence diagram

ID	F.6
NAME	Visualize relevant data
ACTOR	Farmer
ENTRY CONDITION	Farmer has logged in
EVENTS FLOW	• Farmer selects the graphical section responsible for the relevant data visualization, called informally *Relevant section*
EXIT CONDITION	The application displays a section containing information about weather forecasts, farm related tools suggestion, agronomist visit time, amount of water used that month, soil humidity etc.)
OUTPUT	• The farmer can visualize relevant information • Farmer can interact with information (links to shop websites etc.)
EXCEPTION	If the internet connection is lost, the application displays a section informing the task cannot be achieved

Table 20: Visualize relevant data for farmers

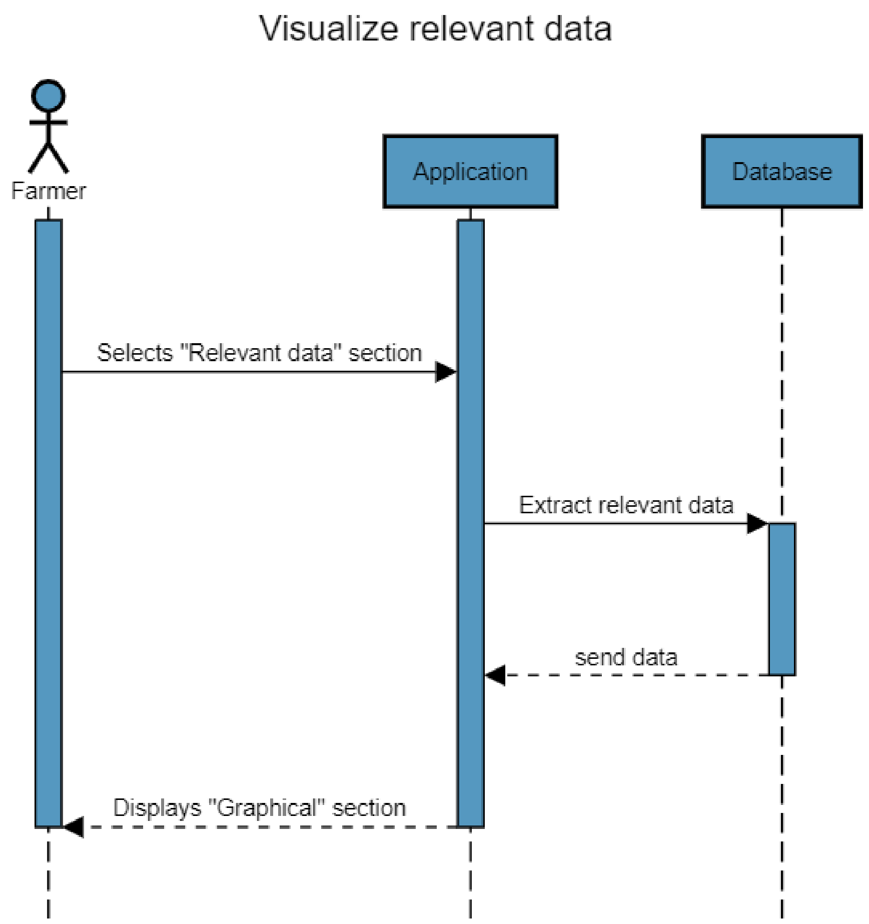


Figure 24: Visualize relevant data for farmers - sequence diagram

3.2.4 Agronomists

Agronomist scenarios

Scenario 1

Yuvaan is an agronomist in the Mahbubnagar district of Telangana. He has the DREAM app installed on his device and uses it to remain in contact with the farmers of his area. When a notification about a new request arrives on his smartphone, he opens the app to check it. X goes to the request section and selects the right conversation to see the new messages. He can now chat directly with the farmer to answer his requests and help him with his problems.

Scenario 2

Swara is an agronomist in the Nizamabad district of Telangana. She has the DREAM app installed on her smartphone to be able to keep track of the visits she did in the past and to organize future visits to the farmers of her area. After chatting with a farmer, she decides to delay a planned visit to that farmer. To do so, she opens the daily plan section in the app and selects the daily plan referred to the previously agreed day. She removes that farmer from the selected daily plan and confirms the change. Afterwards, Swara opens the daily plan of the newly agreed day, adds that farmer and confirms. The system will store the information for the agronomist, so that she can concentrate on her work.

Scenario 3

Raghav is an agronomist in the Jayashankar Bhupalpally district of Telangana. Today he has to visit some farms in his area. Raghav has the DREAM app installed on his smartphone to check the daily plan for the current day, so that he knows the exact list of farmers to visit. Once the visiting day ended, Raghav managed to visit all farmers, except for one who had a last minute emergency and could not be present. Raghav opens the app, goes in the daily plan section and opens the current daily plan. From there, he specifies this deviation by marking the specific farmer as “non-visited” and then confirms the execution of the daily plan. A new visit for that farmer will be scheduled in the future.

Scenario 4

Michelle is an agronomist in the Khammam district of Telangana. She wants to know the situation about the performance of the farmers of her area. For that, Michelle opens the app and goes to the farmers’ performing situation page. From there she can have an overview on how many farmers are performing well and how many of them are under-performing. She sees that there are some farmers with problems who have not received a visit already, so she decides to add them to a daily plan in the near future.

Agronomist use cases

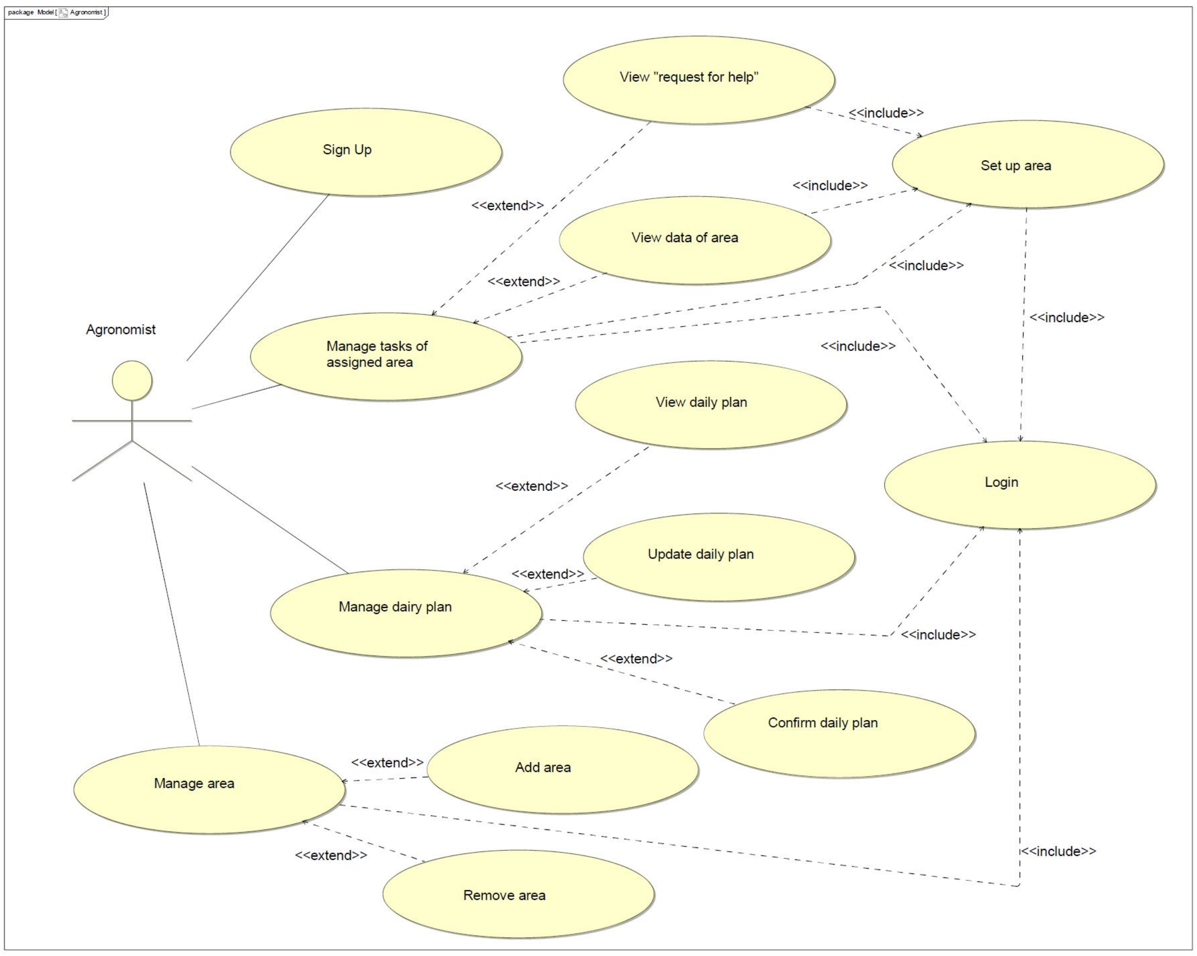


Figure 25: Agronomist use case diagram

Use case tables

ID	A.1
NAME	Add an area under the agronomist's responsibility
ACTOR	Agronomist
ENTRY CONDITION	Agronomist has logged in
EVENTS FLOW	• Agronomist goes to the "Area management" section • The system extracts the data from the database and displays the list of areas under the agronomist's responsibility • Agronomist presses the button "Add an area" • The system displays a page with all the available areas and a search bar • Agronomist makes a research based on the name/location of the area • The system shows the results • Agronomist selects the desired area • The system shows the details of the area selected (number of farmers, number of agronomists, weather information, etc) • Agronomist clicks on "Add this area" • The system updates the database and shows a message of insertion completed
EXIT CONDITION	The system displays the "Area management" section
OUTPUT	• The system has added the agronomist in the database as responsible for that area
EXCEPTION	Agronomist wrongly clicks "Add this area". The system displays a popup notifying that he will be added as responsible for that area and the agronomist clicks on "Cancel" button

Table 21: Add an area for an agronomist

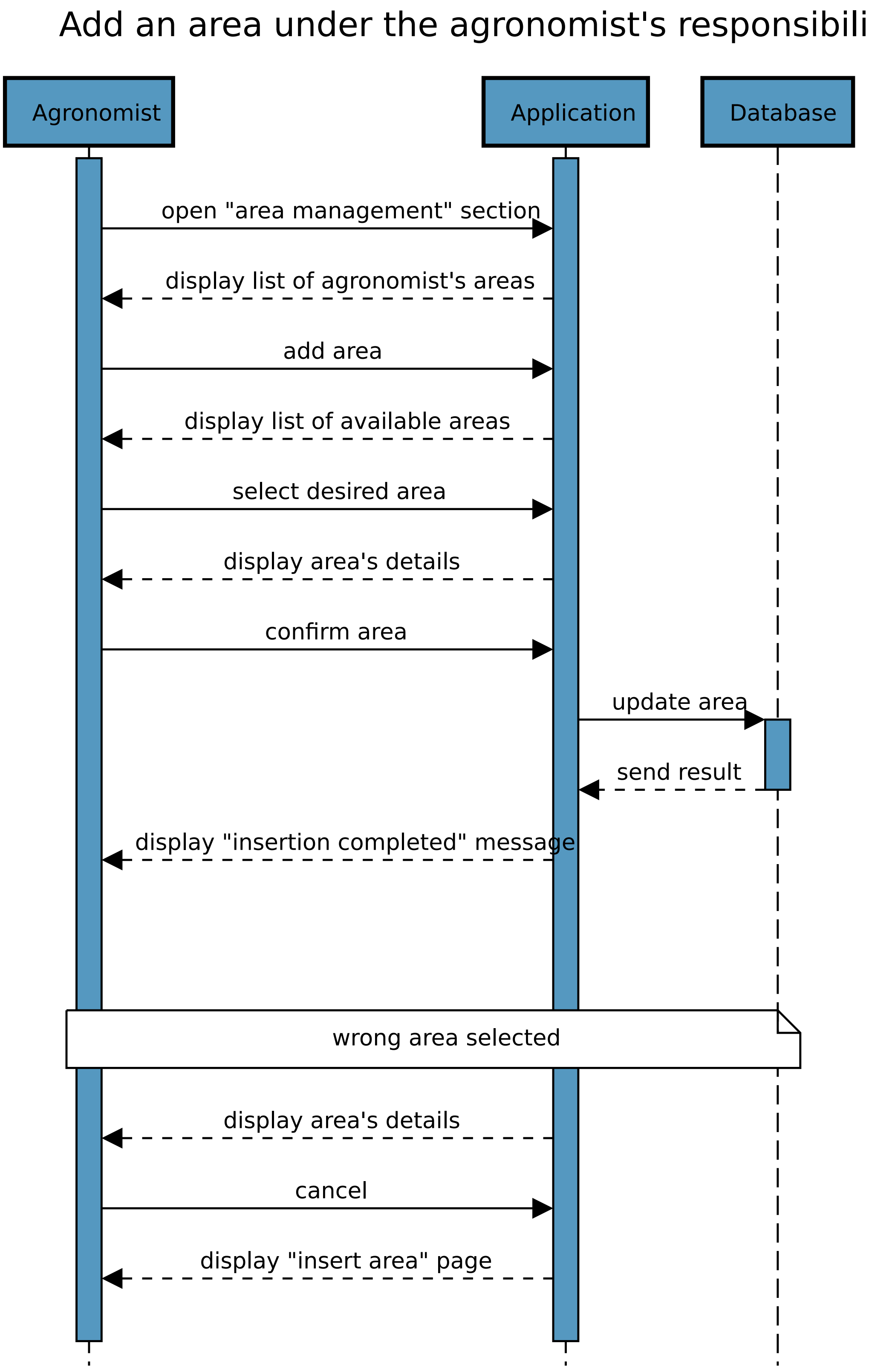

Add an area under the agronomist's responsibili
Agronomist
Application
Database
open "area management" section
display list of agronomist's areas
add area
display list of available areas
select desired area
display area's details
confirm area
update area
send result
display "insertion completed" message
wrong area selected
display area's details
cancel
display "insert area" page

Figure 26: Add an area for an agronomist - sequence diagram

ID	A.2
NAME	Remove an area under the agronomist’s responsibility
ACTOR	Agronomist
ENTRY CONDITIONS	• Agronomist has logged in • Agronomist is responsible for at least one area
EVENTS FLOW	• Agronomist goes to the "Area management" section • The system extracts the data from the database and displays the list of areas under the agronomist’s responsibility • Agronomist presses the button “Remove this area” next to the desired area • The system displays a popup asking for confirmation • Agronomist clicks on “Confirm” • The system updates the database and shows a message of removal completed
EXIT CONDITION	The system displays the "Area management" section
OUTPUT	• The system has removed the agronomist in the database as responsible for that area
EXCEPTION	Agronomist wrongly clicks “Remove this area”. The system displays a popup notifying that he will be removed as responsible for that area and the agronomist clicks on “Cancel” button

Table 22: Remove an area for an agronomist

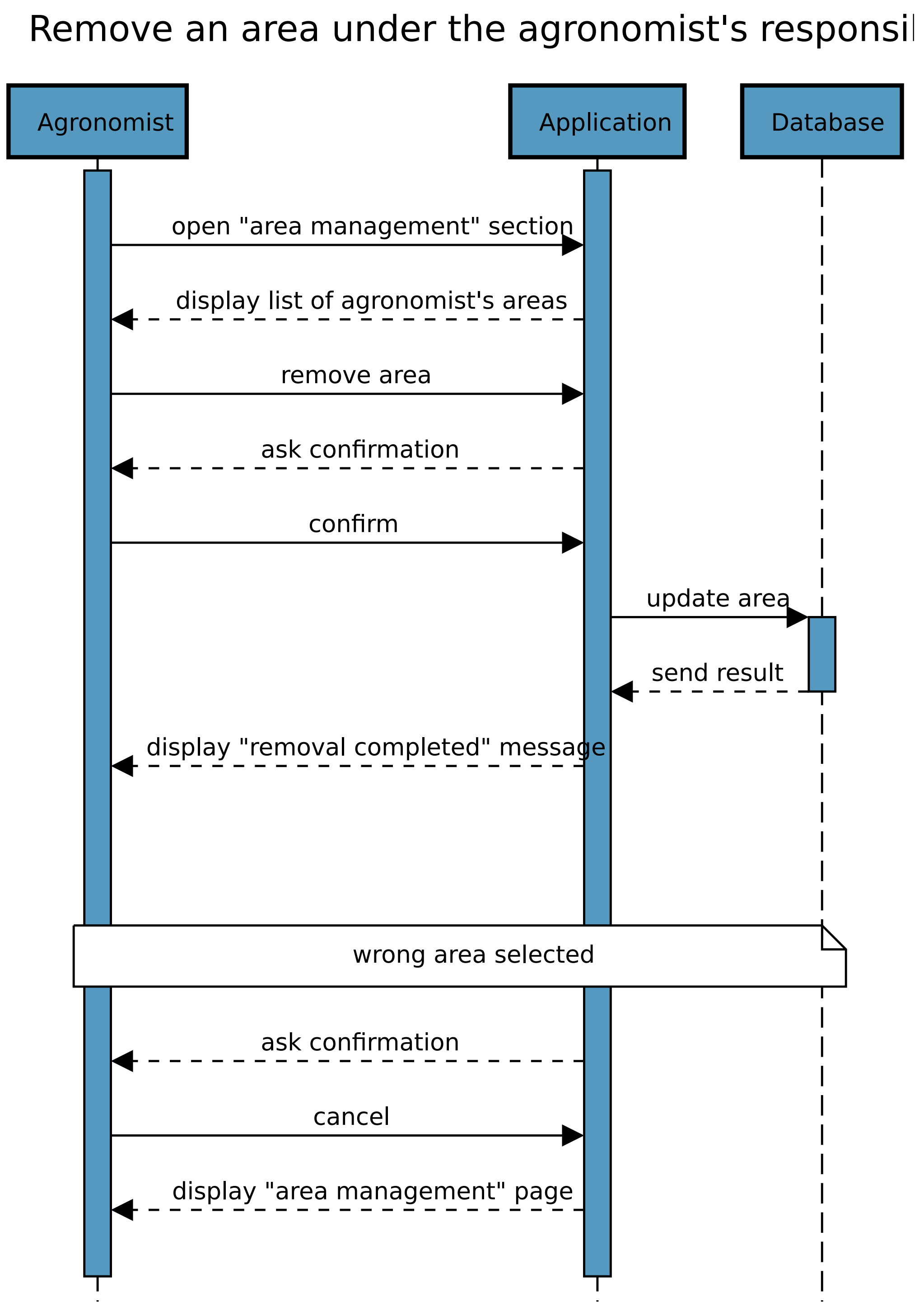

Remove an area under the agronomist's responsil
Agronomist
Application
Database
open "area management" section
display list of agronomist's areas
remove area
ask confirmation
confirm
update area
send result
display "removal completed" message
wrong area selected
ask confirmation
cancel
display "area management" page

Figure 27: Remove an area for an agronomist - sequence diagram

ID	A.3
NAME	Visualize and answer to requests for help
ACTOR	Agronomist
ENTRY CONDITION	Agronomist has logged in
EVENTS FLOW	• Agronomist goes to the “Requests for help” section • The system extracts the data from the database and displays a search bar and the list of conversations (sorted by recent activity), marking the ones which contains new messages • Agronomist selects a conversation • The system displays all the messages contained in that conversation and a text box • Agronomist writes in the text box an answer to the request and clicks “Send” • The system adds the message to the database
EXIT CONDITION	The system remains in the same page, refreshing its content
OUTPUT	• The system adds the message to the database • The system notifies the participants of that conversation about the new message
EXCEPTION	The system cannot connect to the database/server. The system displays an error message.

Table 23: Visualize and answer to requests for help

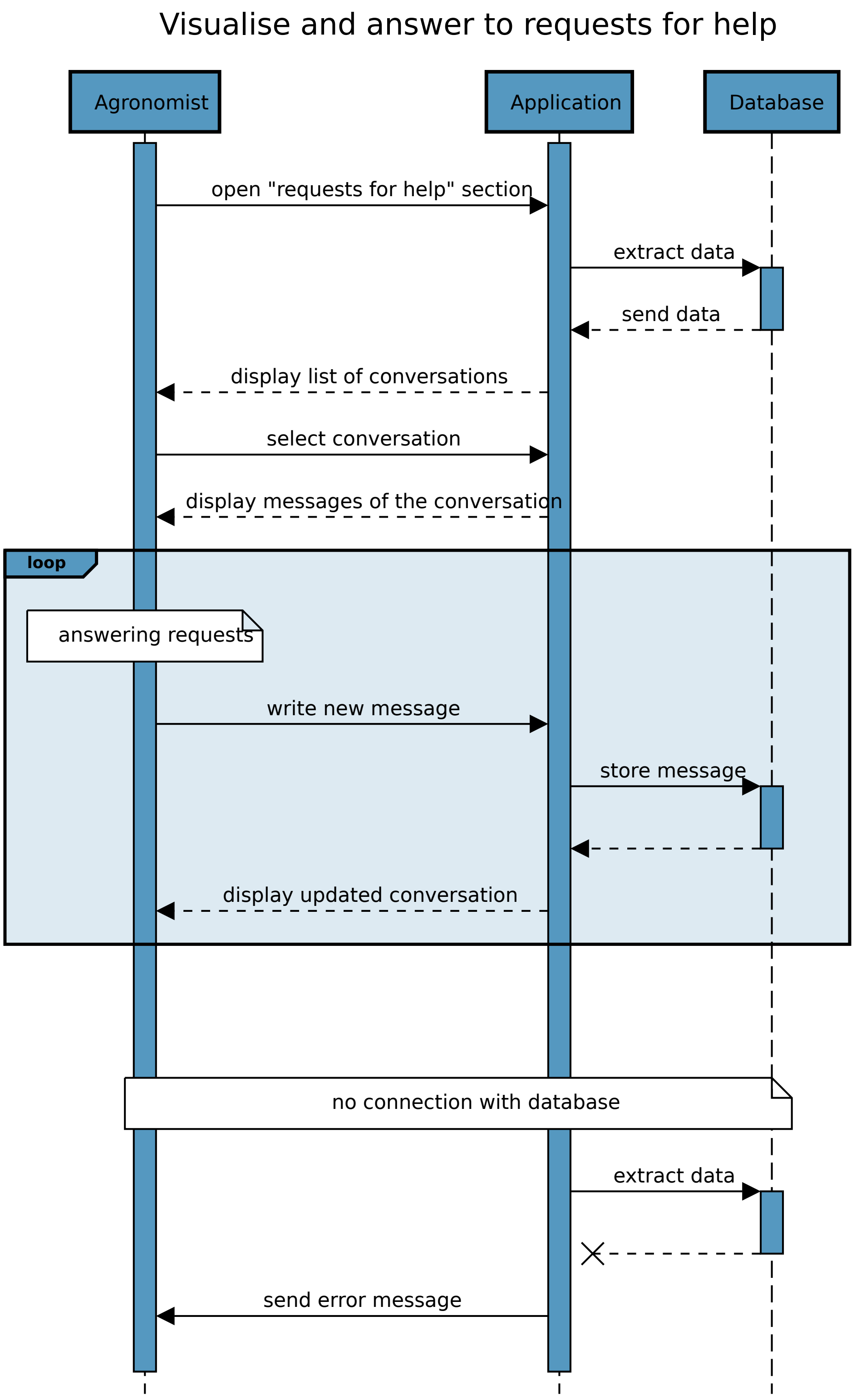

Visualise and answer to requests for help
Agronomist
Application
Database
open "requests for help" section
extract data
send data
display list of conversations
select conversation
display messages of the conversation
loop
answering requests
write new message
store message
display updated conversation
no connection with database
extract data
send error message

Figure 28: Visualize and answer to requests for help - sequence diagram

ID	A.4
NAME	Visualize data of an area
ACTOR	Agronomist
ENTRY CONDITION	Agronomist has logged in
EVENTS FLOW	• Agronomist goes to the “Information about areas” section • The system extract the data from the database and displays the list of areas under the agronomist’s responsibility • Agronomist selects an area • The system displays a page with three main options: “Weather forecasts”, “Farmers’ performing situation” and "Problems encountered" • Agronomist selects the “Weather forecasts” section • The system extracts the data from the database and shows all the information concerning the weather forecasts in that area • Agronomist selects the “Farmers’ performing situation” section • The system extracts the data from the database and displays the list of all the farmers in the area, grouping them in “Best performing”, “Normal performing” and “Under-performing” • Agronomist selects a farmer • The system displays detailed information about that farmer (overall situation, planned visits, past visits, problems encountered) • Agronomist selects the "Problems encountered" section • The system displays the list of problems inserted by the farmers belonging to the Agronomist’s area
EXIT CONDITION	The system returns to the main page
EXCEPTIONS	The system cannot connect to the database/server. The system displays an error message.

Table 24: Visualize data of an area

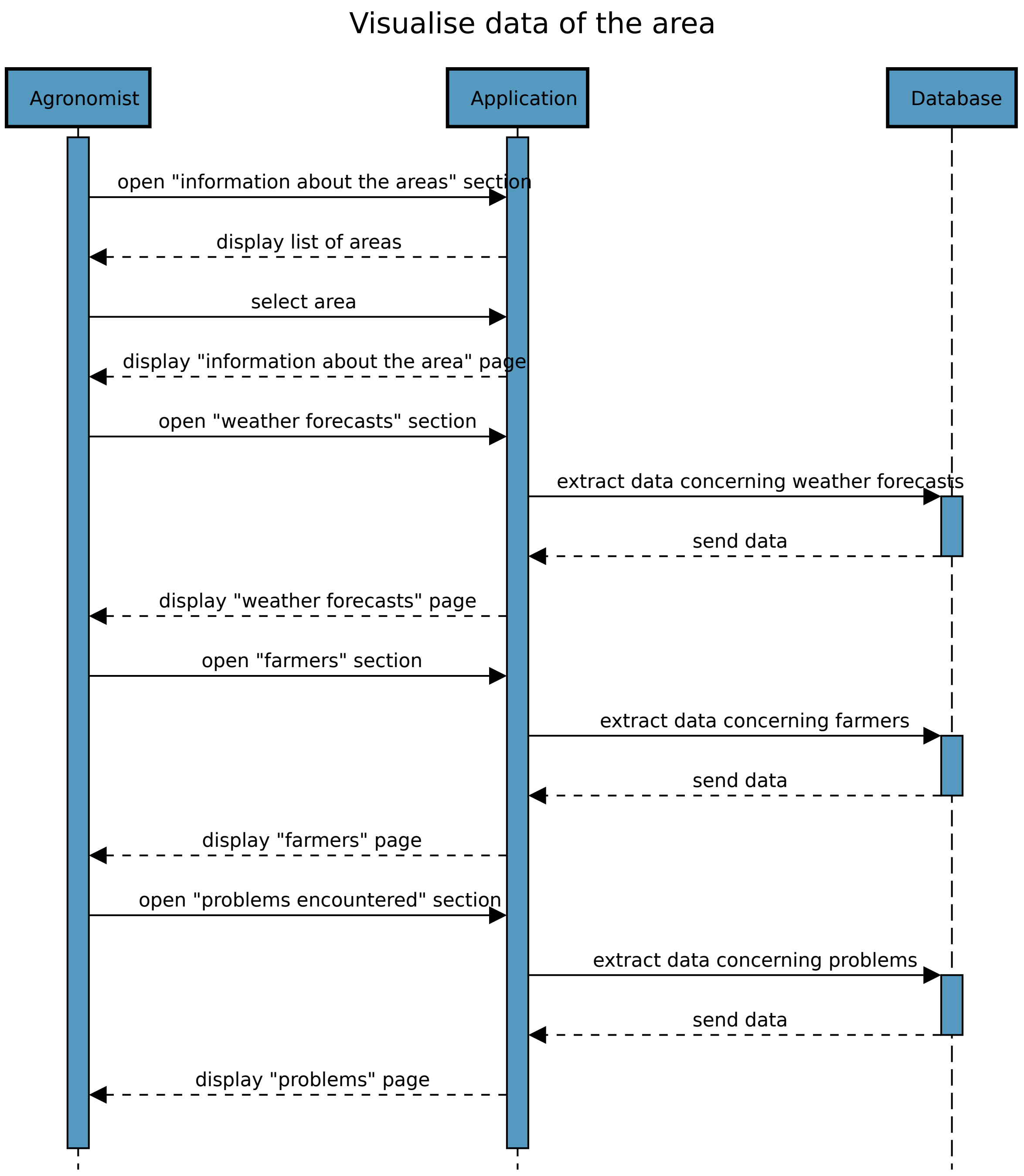

Visualise data of the area
Agronomist
Application
Database
open "information about the areas" section
display list of areas
select area
display "information about the area" page
open "weather forecasts" section
extract data concerning weather forecasts
send data
display "weather forecasts" page
open "farmers" section
extract data concerning farmers
send data
display "farmers" page
open "problems encountered" section
extract data concerning problems
send data
display "problems" page

Figure 29: Visualize data of an area - sequence diagram

ID	A.5
NAME	Visualize and update a daily plan
ACTORS	Agronomist
ENTRY CONDITIONS	• Agronomist has logged in • At least one daily plan is present
EVENTS FLOW	• Agronomist goes to the “Daily Plan” section, which is divided in “Visualize/Update” and “Confirm/Specify deviation from”. Agronomist selects the “Visualise/Update” section • The system extracts the data from the database and displays the list of daily plans for that agronomist • Agronomist selects a daily plan • The system displays all the information about that daily plan (day-month-year, farmers to be visited) • Agronomist clicks on the “Update” button • The system displays the list of farmers contained in the selected daily plan • Agronomist clicks on the “Remove” button near the farmer he wants to remove • The systems displays the updated list of farmers • Agronomist clicks on the “Add farmer” button • The system displays the list of all the farmers for which the agronomist is responsible of and that are not already in the selected daily plan. In addition, for each farmer it is shown the list of problems encountered and the list of past and future visits (planned also by other agronomists) in a sort of compact calendar • Agronomist selects a subset of farmers and clicks on “Add” • The system displays the updated list of farmers • Agronomist checks the info and clicks on “Confirm changes”
EXIT CONDITION	The system shows a popup to notify the agronomist of the success of the operation
OUTPUT	• The updated daily plan is stored in the database
EXCEPTIONS	• The selected daily plan has already been confirmed and cannot be updated anymore. The system shows an error message. • The selected daily plan is referring to the current day and cannot be updated anymore. The system shows an error message. • Agronomist has made the wrong modifications to the daily plan. Instead of clicking “Confirm changes”, he/it clicks on “Discard changes”. The system displays the original daily plan and doesn’t store the new one in the database.

Table 25: Visualize and update a daily plan

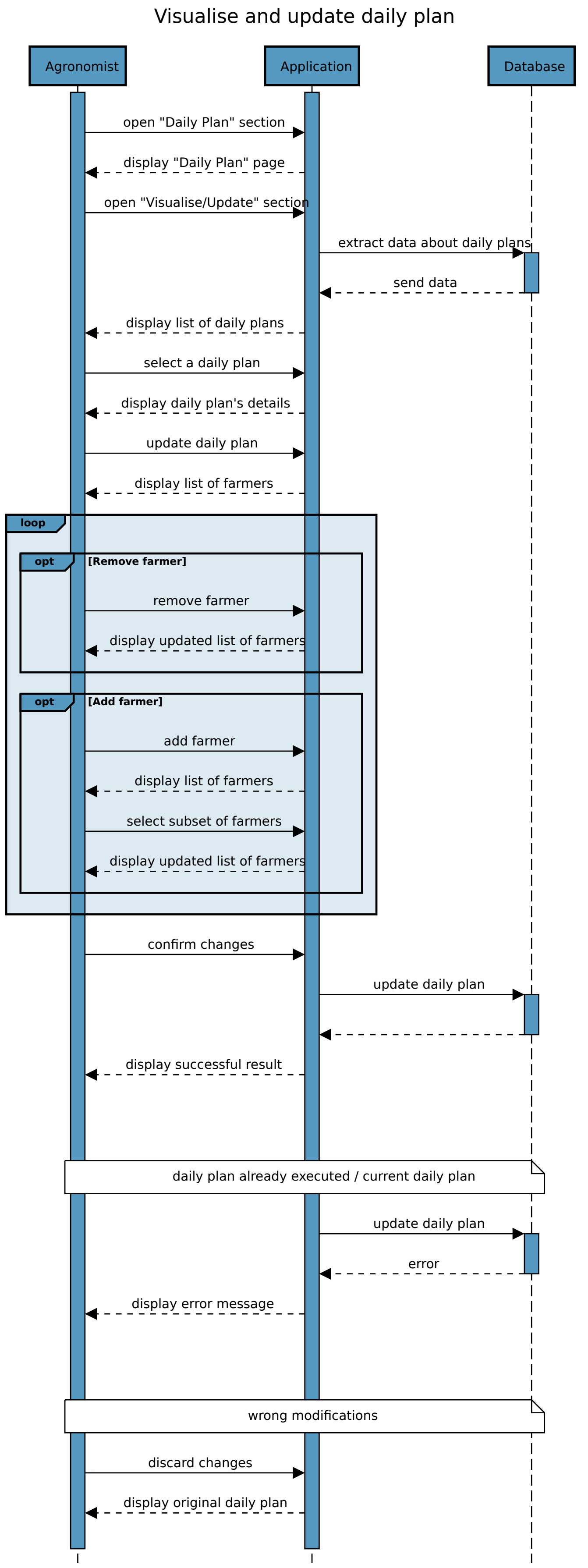

Visualise and update daily plan
Agronomist
Application
Database
open "Daily Plan" section
display "Daily Plan" page
open "Visualise/Update" section
extract data about daily plans
send data
display list of daily plans
select a daily plan
display daily plan's details
update daily plan
display list of farmers
loop
opt
[Remove farmer]
remove farmer
display updated list of farmers
opt
[Add farmer]
add farmer
display list of farmers
select subset of farmers
display updated list of farmers
confirm changes
update daily plan
display successful result
daily plan already executed / current daily plan
update daily plan
error
display error message
wrong modifications
discard changes
display original daily plan

Figure 30: Visualize and update a daily plan - sequence diagram

ID	A.6
NAME	Confirm execution of daily plan
ACTOR	Agronomist
ENTRY CONDITIONS	• Agronomist has logged in • Agronomist has at least one daily plan
EVENTS FLOW	• Agronomist goes to the “Daily Plan” section, which is divided in “Visualize/Update” and “Confirm/Specify deviation from”. Agronomist selects the “Confirm/Specify deviation from” section. • The system extracts the data from the database, displays the daily plan for the current day and enables the agronomist to select the subset of farmers which has not been visited that day • Agronomist checks the info, selects the subset of farmers and clicks on “Continue” • The system displays a compact summary of the info and asks for confirmation • Agronomist checks the info and clicks on “Confirm”
EXIT CONDITION	The system shows a popup to notify the agronomist of the success of the operation
OUTPUT	• The system stores the completed daily plan in the database • The system increments by one the number of visits received by the farmers that actually have been visited • The system rearranges the non-visited farmers in future daily plans
EXCEPTION	Agronomist has selected the wrong subset of farmers. Given the compact summary, it clicks on “Cancel”. The system displays again the daily plan and enables the agronomist to choose another subset of farmers.

Table 26: Confirm execution of daily plan

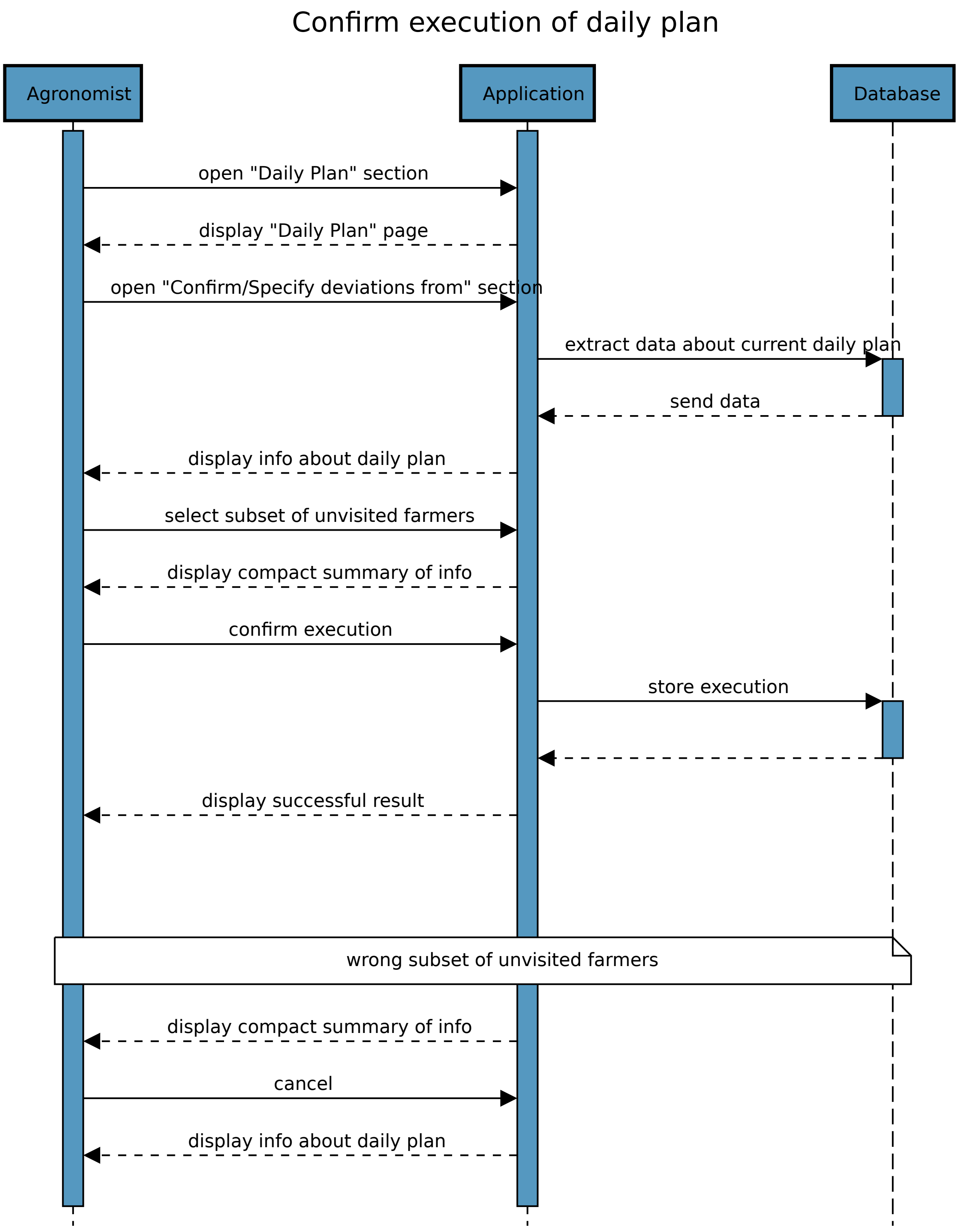

Confirm execution of daily plan
Agronomist
Application
Database
open "Daily Plan" section
display "Daily Plan" page
open "Confirm/Specify deviations from" section
extract data about current daily plan
send data
display info about daily plan
select subset of unvisited farmers
display compact summary of info
confirm execution
store execution
display successful result
wrong subset of unvisited farmers
display compact summary of info
cancel
display info about daily plan

Figure 31: Confirm execution of daily plan - sequence diagram

3.2.5 Traceability Matrix

Goal	Domain assumption	Requirement
G1	D6, D7, D9, D14	R2, R3, R4
G2	D6, D7, D10, D13, D14, D16	R3, R5, R6
G3	D6, D7, D8, D9, D14, D19	R3, R4, R7, R8
G4	D6, D7, D10, D11, D12, D14, D16	R2, R3, R9, R10
G5	D1, D2, D3	R11, R12, R13, R14, R18, R23
G6	D14, D15	R12, R13, R14, R15, R16, R17, R21, R22
G7	D14, D17, D19	R11, R19, R20, R27, R28
G8	D17	R11, R24, R25, R26
G9	D19, D22	R36, R37
G10	D14, D17, D19	R27, R28
G11	D6	R30, R31
G12	D1, D2, D3	R29
G13	D6, D19, D20	R32, R33, R38
G14	D19, D21	R34, R35

Table 27: Traceability matrix

3.3 Performance Requirements

The system should have a good general response time, that may change depending on the specific service offered. Here are some numeric examples the system might follow:

- messages and requests for help may be delivered in 5 seconds, or less;
- daily plans may be accessed in 7 seconds, or less;
- data concerning areas, with information about farmers and weather forecasts, may be given in 10 seconds, or less;
- registration and login operations may be confirmed in 6 seconds, or less.

It is important to notice that these numbers will also depend on the Internet connection of the users, which is assumed to be good.

The average workload of the system is expected to be very high since the user base is quite big (in the order of hundreds of thousands or even millions, considering Telangana's population and farming prevalence) and could grow over time, if the system is extended to other states. The system should guarantee millions of operations every day, including messages, requests, access to data, etc. This could be accomplished with a good distribution of the work among the components of the system, especially during daytime.

Since most of the operations are handled by the servers of the system, the mobile app may be lightweight, in order to be reactive and to occupy little memory on the user's device. Also the web page should be lightweight and responsive. These software elements should take into account communication protocols unreliability.

The system interacts a lot with external services, collecting and providing data and exploiting computational analysis on such data. All the data transmission with these third party entities should be reasonably fast and scalable on the increasing number of users over time.

3.4 Design Constraints

3.4.1 Standards compliance

According to UNDP GitHub repository (ref.[7]) the required platform will be compliant to DPG Standards (ref.[2]) that defines open-source software, open data, open AI models, open standards, and open content that adhere to privacy and other applicable best practices, do no harm by design and are of high relevance for attainment of the United Nations 2030 SDGs.

3.4.2 Privacy constraints

The above mentioned collection of standards (ref.[2]) is responsible to check also privacy related requirements that have to be guaranteed. In particular, it shall ensure adherence with relevant privacy, domestic and international laws such as GDPR or the ECOWAS supplementary act in order to be accepted as a reliable software. When a user registers to the application, the privacy policy must be read and accepted, otherwise, he/she will not be able to use the service. By the fact that the platform is going to cover Telangana's territory, then STQC directorate of the Ministry of Electronics and Information Technology (ref.[2]) data privacy standards shall be the minimum required constraints to be satisfied.

3.5 Software System Attributes

3.5.1 Usability

The system should be very easy to use, since the user base is very large and various and also comprehends many farmers. The graphical interface of both the web application and the smartphone application should help the users to identify the proper choice on the screen.

3.5.2 Reliability

The system should be fault tolerant in order to prevent downtime. The highest number of accesses is expected to be in the early morning (e.g., agronomists accessing the daily plan, policy makers monitoring the results) and in the late afternoon (e.g., agronomists confirming the execution of the daily plan, farmers inserting info about their production and problems).
Redundancy may be considered for a storage implementation, in order to recover from eventual data losses and to guarantee the lowest MTTR possible.

3.5.3 Availability

The system should offer its functionalities as long as possible, with an availability of 99% or more, so that 3.65 days of downtime per year are allowed. Some critical parts, for example the daily plan handler or the request for help platform, may have an higher availability value of 99.9%, so that only 8.76 hours of downtime per year are allowed.
Possible maintenance procedures in the database or in the servers may be performed using replicas or at night, in order to ensure continuity to the service.

3.5.4 Security

The communication between parties should be encrypted and sent along secure channels (e.g., using SSL and HTTPS protocols), in order to guarantee the protection of user's sensitive data.
The system should guarantee that all the operations on the database are always authorized, for example performing RBAC, an authorization scheme that grants access rights based on the role of the user.

3.5.5 Portability

Since it is a Digital Public Good, the system will be compatible with the principal operating systems, either computers and mobile, and the principal web browsers. The downloadable application will be available for both Android and iOS, using for example cross platform development tools.

3.5.6 Maintainability

The system may be composed of scalable and reusable modules, which are easier to maintain and replace in case of failure. The source code may be commented as well as possible and the correlated documentation may be kept updated during the whole lifecycle of the system. Ordinary maintenance, for bug fixes and improvements, may be scheduled for night time, when the user traffic is minimal.

3.5.7 Scalability

The system should guarantee high scalability for future upgrades and expansions. For this purpose, modularity and low coupling are key aspects of the designing and developing phases.

4 Formal Analysis Using Alloy

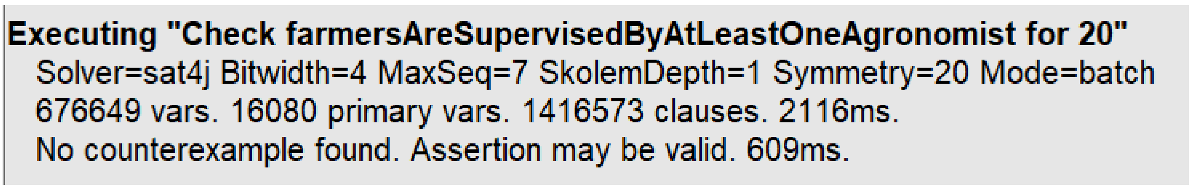
In this section we provide the whole Alloy code used for the analysis of the S2B, the results of the assertions and the diagrams representing the different worlds generated by the predicates.

4.1 Alloy Code

```
// User (Farmer, Agronomist, PolicyMaker)
abstract sig User {
       userID: one ID,
       username: one Username,
       password: one Password,
       email: one Email
}

sig Username {}
sig Password {}
sig Email {}
sig ID {}

sig Farmer extends User {
       performingType: one PerformingType
}

sig Agronomist extends User {
       specialization: one SpecializationType
}

sig PolicyMaker extends User {}


// PerformingType - enum
abstract sig PerformingType {}
one sig GOOD_PERFORMING extends PerformingType {}
one sig NORMAL_PERFORMING extends PerformingType {}
one sig UNDER_PERFORMING extends PerformingType {}

// SpecializationType - enum
abstract sig SpecializationType {}
one sig SPECIALIZATION_A extends SpecializationType {}
one sig SPECIALIZATION_B extends SpecializationType {}
one sig SPECIALIZATION_C extends SpecializationType {}


// Daily Plan
sig DailyPlan {
       date: one Date,
       agronomist: one Agronomist,
       executed: one Bool,
       deviationList: set Farmer,
       visitList: set Visit
} {
       #visitList >0
}

sig Date {}

sig Visit {
       date: one Date,
       farmer: one Farmer
}

// Bool - enum
abstract sig Bool {}
one sig True extends Bool {}
one sig False extends Bool {}


// Message
```

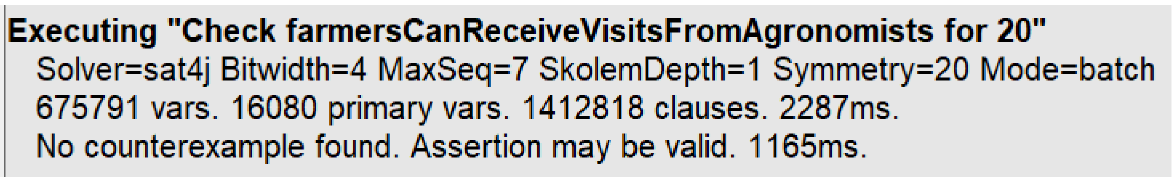
```
abstract sig Message {
        sender: one User,
        receiver: set User,
        text: one Text
} {
        // message must have at least one receiver
        #receiver >0
}

sig Text {}

sig DiscussionMessage extends Message {
        isStartingMessage: one Bool
}

sig ChatMessage extends Message {
        isRequestMessage: one Bool
}


// RequestType - enum
abstract sig RequestType {}
one sig HELP extends RequestType {}
one sig SUGGESTION extends RequestType {}


// Forum e Request
sig Forum {
        forumID: one ID,
        discussionMessageList: set DiscussionMessage,
        startingUser: one Farmer,
        isSolved: one Bool
}

sig RequestChat{
        requestID: one ID,
        title: one ChatTitle,
        chatMessageList: set ChatMessage,
        requestType: one RequestType,
        participants: set User,
        startingUser: one Farmer
}

sig ChatTitle {}


// Area
sig Area {
        areaID: one ID,
        agronomists: set Agronomist,
        farmers: set Farmer
} {
        // non-empty area
        #agronomists >0
        #farmers >0
}


//
sig Problem {}

// ProductType - enum
abstract sig ProductType {}
one sig PRODUCT_A extends ProductType {}
one sig PRODUCT_B extends ProductType {}
one sig PRODUCT_C extends ProductType {}

sig Amount {}
sig UnitOfMeasurement {}


sig Product {
        type: one ProductType,
        amount: one Amount,
        unitOfMeasurement: one UnitOfMeasurement,
        --description: one String
}


// Incentive
sig Incentive {
        incentiveID: one ID,
        --description: one String,
        value: one Amount
}

sig IncentiveAssigning{
        incentive: one Incentive,
        incentiveGiver: one PolicyMaker,
        receiver: one Farmer,
        date: one Date
}


// Good Practice
sig GoodPractice {
        practiceID: one ID,
        requestedBy: one PolicyMaker,
        requestedTo: one Farmer,
        content: one Text
}



// FACTS

// credentials constraints
fact {
        // different Users have different userIDs
        no disj u1, u2: User |u1.userID = u2.userID

        // different Users have different usernames
        no disj u1, u2: User |u1.username = u2.username

        // different Users have different emails
        no disj u1, u2: User |u1.email = u2.email

        // a password must belong to a User
        all p: Password |(some u: User |u.password = p)

        // an email must belong to a User
        all e: Email |(one u: User |u.email = e)
}

// Daily Plan constraints
fact {
        // all daily plans must contain visits with the same date
        all d: DailyPlan |
                all v: d.visitList |d.date = v.date

        // unvisitedList must contain only farmers specified in the visitList
        all d: DailyPlan |
                all f: d.deviationList |(one v: d.visitList |v.farmer.userID = f.userID)

        // unvisitedList can be non-empty only if daily plan has been executed
        all d: DailyPlan |
                #d.deviationList >0 implies d.executed = True

        // different daily plans must have different dates
        no disj d1, d2: DailyPlan |d1.date = d2.date

        // a daily plan cannot contain multiple visits to the same farmer
        all d: DailyPlan |
                no disj v1, v2: d.visitList |v1.farmer.userID = v2.farmer.userID
        // a visit must belong to a daily plan
        all v: Visit |
                one d: DailyPlan |v in d.visitList
}

// Agronomists are assigned to at least one area
fact {
        all ag: Agronomist |
                some ar: Area |ag in ar.agronomists
}

// Farmers are assigned to only one area
fact {
        all f: Farmer |
                one a: Area |f in a.farmers
}

// Messages constraints
fact {
        // User cannot send messages to himself
        all m: Message |m.sender not in m.receiver

        // PolicyMakers cannot send messages
        no m: Message |m. sender in PolicyMaker

        // PolicyMakers cannot receive messages
        no m: Message |(some r: m.receiver |r in PolicyMaker)

        // PolicyMakers cannot participate to requests
        no r: RequestChat |(some p: r.participants |p in PolicyMaker)

        // Agronomist cannot send REQUEST messages or DISCUSSION messages
        no m: ChatMessage |(m.sender in Agronomist and m.isRequestMessage = True)

        // all m: RequestReplyMessage | (m.sender.userType = AGRONOMIST implies m.requestReplyType = REPLY)
        no m: DiscussionMessage |m.sender in Agronomist
        no m: DiscussionMessage |(some r: m.receiver |r in Agronomist)

        // discussion messages must belong to a Forum
        all m: DiscussionMessage |
                one f: Forum |m in f.discussionMessageList

        // request_reply message must belong to a Request
        all m: ChatMessage |
                one r: RequestChat |m in r.chatMessageList
}

// Requests constraints
fact {
        /* requests from a farmer must have as participant at least one Agronomist
        responsible of the farmer's area*/
        all r: RequestChat |one a: Area |
                (r.startingUser in a.farmers and
                (some ag: a.agronomists |ag in r.participants))

        // requests must have as participant the farmer that started it
        all r: RequestChat |r.startingUser in r.participants

        // request messages must be delivered to all the participants, but not to the sender
        all r: RequestChat |
                all m: r.chatMessageList |
                        all p: r.participants |(p in m.receiver or p = m.sender)

        // request messages must be sent by and delivered to participants only
        all r: RequestChat |
                all m: r.chatMessageList |
                        (all u: m.receiver |u in r.participants) and m.sender in r.participants

        // a request message must be sent by the farmer who started the conversation
        all r: RequestChat |
                all m: r.chatMessageList |(m.isRequestMessage = True
                    implies (m.sender = r.startingUser and m.sender in Farmer))


        // a request discussion must contain only one request message
        all r: RequestChat |
                one m: r.chatMessageList |m.isRequestMessage = True
}

// Forum constraints
fact {
        // forum messages must be delivered to all farmers, but not to the sender
        all m: DiscussionMessage |
                all f: Farmer |(f in m.receiver or f = m.sender)

        // forums can have only one starting message
        all f: Forum |
                one m: f.discussionMessageList |m.isStartingMessage = True

        // starting message must belong to starting user
        all f: Forum |
                all m: f.discussionMessageList |
                        m.isStartingMessage = True implies m.sender = f.startingUser
}

// Incentive constraints
fact {
        // no incentive assigned multiple times on the same date to the same farmer
        no disj ia1, ia2: IncentiveAssigning |
                ia1.incentive = ia2.incentive and ia1.receiver = ia2.receiver and ia1.date = ia2.date

        // different incentives have different IDs
        no disj i1, i2: Incentive |i1.incentiveID = i2.incentiveID
}

// Good Practice constraints
fact {
        // different good practices have different IDs
        no disj gp1, gp2: GoodPractice |gp1.practiceID = gp2.practiceID
}

// an ID is assigned to only one entity
fact {
        no f: Forum, rc: RequestChat |f.forumID = rc.requestID
        no f: Forum, u: User |f.forumID = u.userID
        no rc: RequestChat, u: User |rc.requestID = u.userID
        no i: Incentive, rc: RequestChat |i.incentiveID = rc.requestID
        no i: Incentive, f: Forum |i.incentiveID = f.forumID
        no i: Incentive, u: User |i.incentiveID = u.userID
        no gp: GoodPractice, f: Forum |gp.practiceID = f.forumID
        no gp: GoodPractice, rc: RequestChat |gp.practiceID = rc.requestID
        no gp: GoodPractice, u: User |gp.practiceID = u.userID
        no gp: GoodPractice, i: Incentive |gp.practiceID = i.incentiveID
        no a: Area, f: Forum |a.areaID = f.forumID
        no a: Area, rc: RequestChat| a.areaID = rc.requestID
        no a: Area, u: User |a.areaID = u.userID
        no a: Area, i: Incentive |a.areaID = i.incentiveID
        no a: Area, gp: GoodPractice |a.areaID = gp.practiceID
}



// ASSERTIONS

// farmers are supervised by at least one agronomist
assert farmersAreSupervisedByAtLeastOneAgronomist {
        all f: Farmer |one a: Area |
                some ag: Agronomist |f in a.farmers and ag in a.agronomists
}
check farmersAreSupervisedByAtLeastOneAgronomist for 20

// farmers can send requests for help to agronomists
assert farmersCanSendRequestsForHelpToAgronomists {
        all r: RequestChat|
                some f: Farmer |
                        some ag: Agronomist |f in r.participants and ag in r.participants
}
check farmersCanSendRequestsForHelpToAgronomists for 20

// farmers can receive visits from agronomists
assert farmersCanReceiveVisitsFromAgronomists {
        all d: DailyPlan |
                some v: Visit |
                        v in d.visitList
}
check farmersCanReceiveVisitsFromAgronomists for 20

// policy makers can assign incentives to farmers
assert policyMakerCanAssignIncentivesToFarmers {
        all i: IncentiveAssigning |i.incentive in Incentive and i.incentiveGiver in PolicyMaker and i.receiver in Farmer
}
check policyMakerCanAssignIncentivesToFarmers for 20



// PREDICATES

// simulation 1 - Daily Plan and Visit
pred world1 {
        #Farmer = 3
        #Agronomist = 1
        #PolicyMaker = 0
        #DailyPlan = 2
        #Message = 0
        #Visit = 4
}
run world1 for 20

// simulation 2 - Request
pred world2 {
        #Farmer = 3
        #Agronomist = 2
        #PolicyMaker = 0
        #DailyPlan = 0
        #DiscussionMessage = 0
        #RequestChat = 1
        #RequestChat.participants = 4
        #ChatMessage = 3
}
run world2 for 20

// simulation 3 - Forum
pred world3 {
        #Farmer = 4
        #Agronomist = 1
        #PolicyMaker = 0
        #DailyPlan = 0
        #DiscussionMessage = 4
        #RequestChat = 0
        #Product = 0
        #Incentive = 0
}
run world3 for 20

// simulation 4 - Good Practice
pred world4 {
        #Farmer = 2
        #Agronomist = 1
        #PolicyMaker = 2
        #GoodPractice = 3
}
run world4 for 10

// simulation 5 - Incentive
pred world5 {
        #Farmer = 2
        #PolicyMaker = 4
        #IncentiveAssigning = 2
        # Incentive = 3
        # Product = 0
        #DailyPlan = 0
        #Message = 0
}
run world5 for 10
```

4.2 Assertions



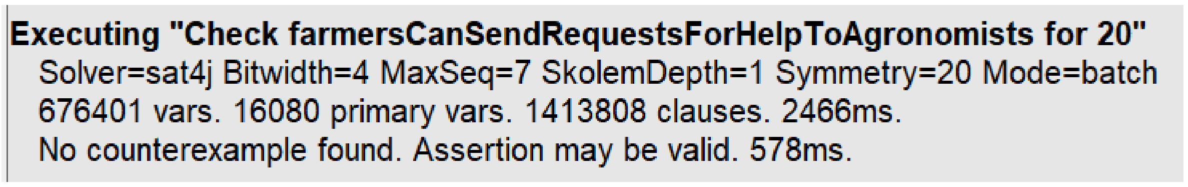
```
Executing "Check farmersAreSupervisedByAtLeastOneAgronomist for 20"
   Solver=sat4j Bitwidth=4 MaxSeq=7 SkolemDepth=1 Symmetry=20 Mode=batch
   676649 vars. 16080 primary vars. 1416573 clauses. 2116ms.
   No counterexample found. Assertion may be valid. 609ms.
```

Figure 32: Assertion 1



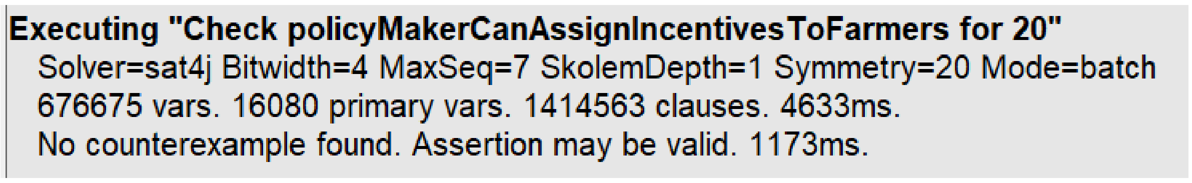
```
Executing "Check farmersCanReceiveVisitsFromAgronomists for 20"
   Solver=sat4j Bitwidth=4 MaxSeq=7 SkolemDepth=1 Symmetry=20 Mode=batch
   675791 vars. 16080 primary vars. 1412818 clauses. 2287ms.
   No counterexample found. Assertion may be valid. 1165ms.
```

Figure 33: Assertion 2



```
Executing "Check farmersCanSendRequestsForHelpToAgronomists for 20"
   Solver=sat4j Bitwidth=4 MaxSeq=7 SkolemDepth=1 Symmetry=20 Mode=batch
   676401 vars. 16080 primary vars. 1413808 clauses. 2466ms.
   No counterexample found. Assertion may be valid. 578ms.
```

Figure 34: Assertion 3



```
Executing "Check policyMakerCanAssignIncentivesToFarmers for 20"
   Solver=sat4j Bitwidth=4 MaxSeq=7 SkolemDepth=1 Symmetry=20 Mode=batch
   676675 vars. 16080 primary vars. 1414563 clauses. 4633ms.
   No counterexample found. Assertion may be valid. 1173ms.
```

Figure 35: Assertion 4

4.3 Worlds

In this section we present the world diagrams generated by the Alloy predicates. This worlds are intended to show separate parts of the system, each one focusing on a different aspect.

World 1

Figure 36 represents a world focused on Daily Plans and Visits. It is possible to see that visits belonging to a specific daily plan have the same date of that daily plan. Also, a farmer cannot be visited twice in the same daily plan, while it is possible to receive visits in separate days. Moreover, it is possible to see that the farmers and the agronomist belong to the same area. In the end, each user has its own credentials.

World 2

Figure 37 represents a world focused on Requests, in this case a request for help. It is possible to see that each message sent in the "chat" is received by all the participants (except the sender, of course), while those users who do not belong to the "chat" (in this case Farmer1) cannot send or receive messages of that conversation. Also, a request has at least one agronomist as a participant (in this case even two). Moreover, the first message of the "chat" (ChatMessage2, isRequestMessage = True) is correctly sent by the farmer that made the request (Farmer0, startingUser), while the other messages are replies. In the end, each user has its own credentials.

World 3

Figure 38 represents a world focused on Forums. It is possible to see that different farmers can write (send a message) in the forum and each message is received by all farmers, while the agronomist does not participate to the forum. As for Requests (world 2), also here there is the correct mapping between the startingMessage (DiscussionMessage3) and the startingUser (Farmer0). In the end, each user has its own credentials.

World 4

Figure 39 represents a world focused on Good Practices. It is possible to see that each good practice has been requested by a policy maker and requested to a good-performing farmer. Also, the content of each practice is out of the scope of the analysis and is collapsed to a single entity (Text). Moreover, the agronomist does not act directly in this process. In the end, each user has its own credentials.

World 5

Figure 40 represents a world focused on Incentives. It is possible to see that a policy maker can assign incentives to a farmer (in this case different incentives to the same farmer) and there can be incentives not assigned yet (Incentive2). In this situation there are also two areas: each area has at least one agronomist (in this case the same one) and each farmer belongs to only one area. In the end, each user has its own credentials.

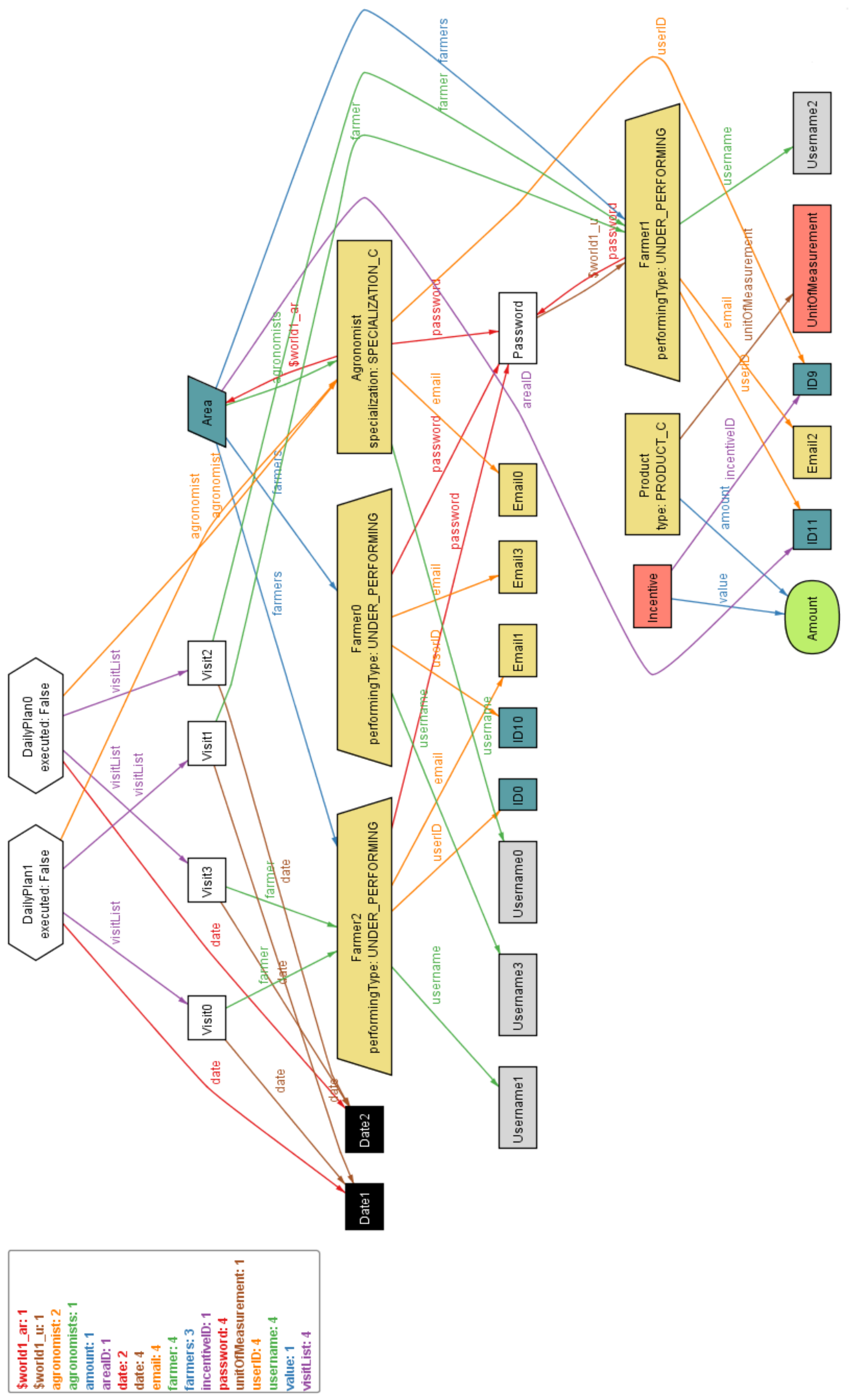


Figure 36: World focused on Daily Plans and Visits

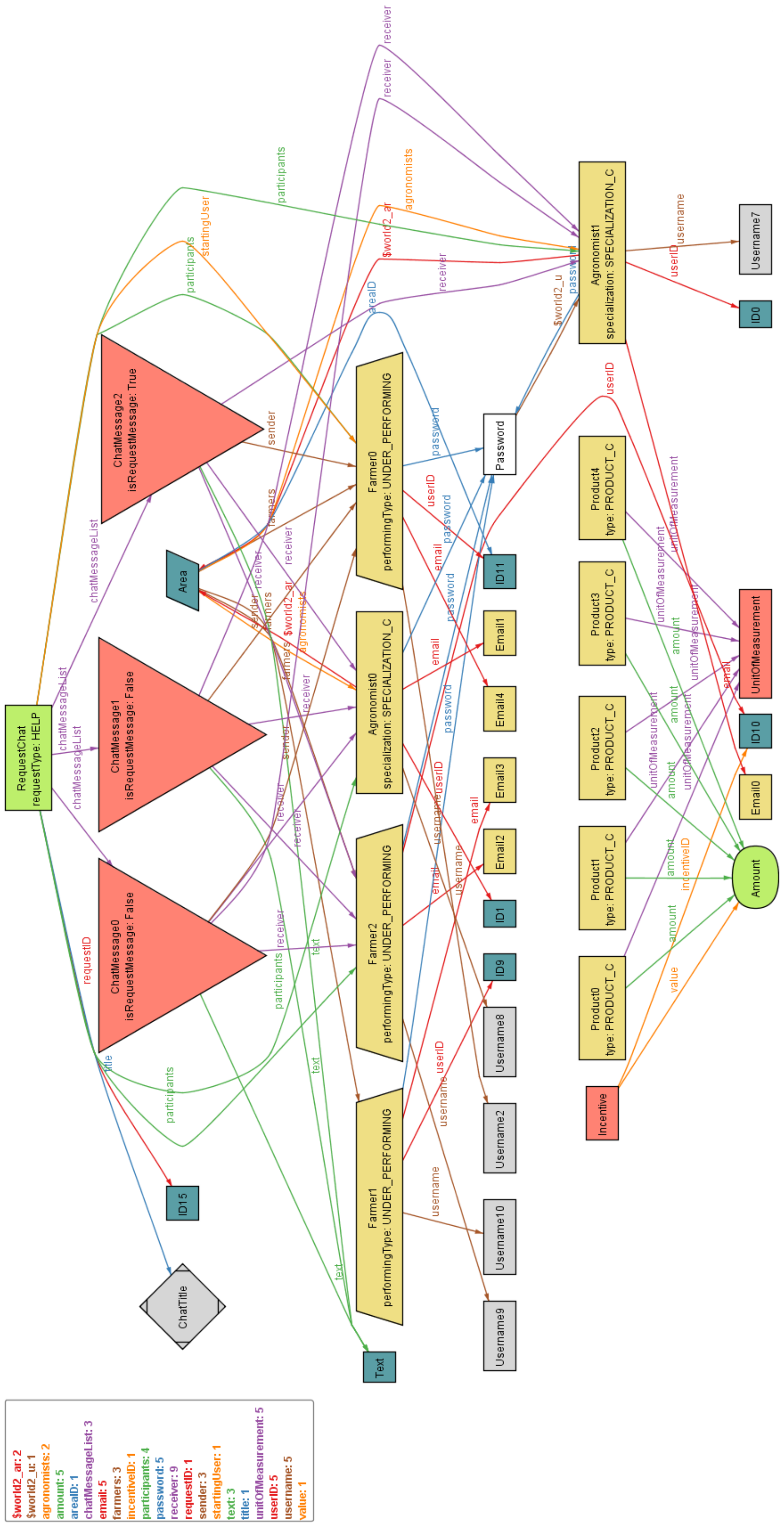


Figure 37: World focused on Requests

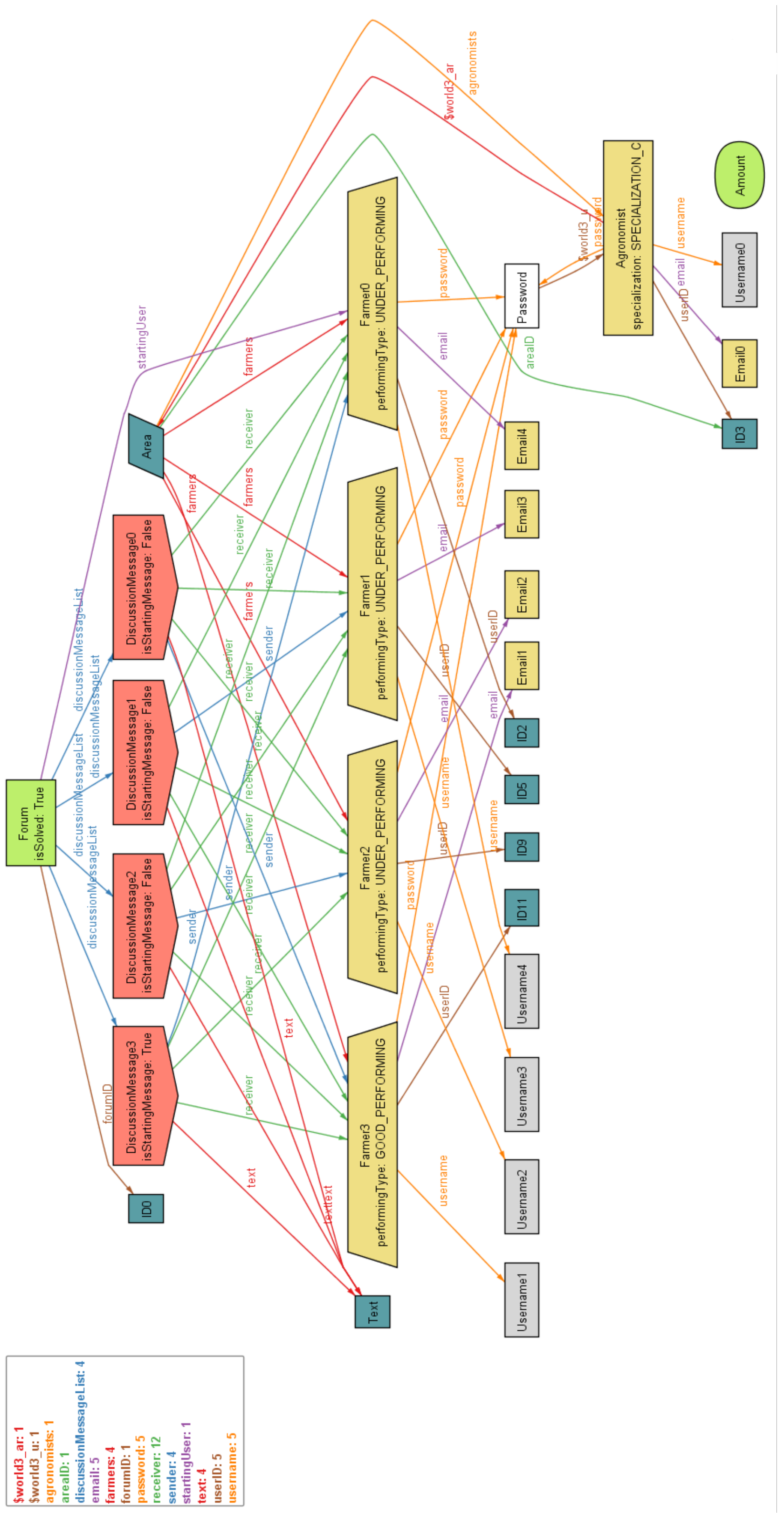


Figure 38: World focused on Forums

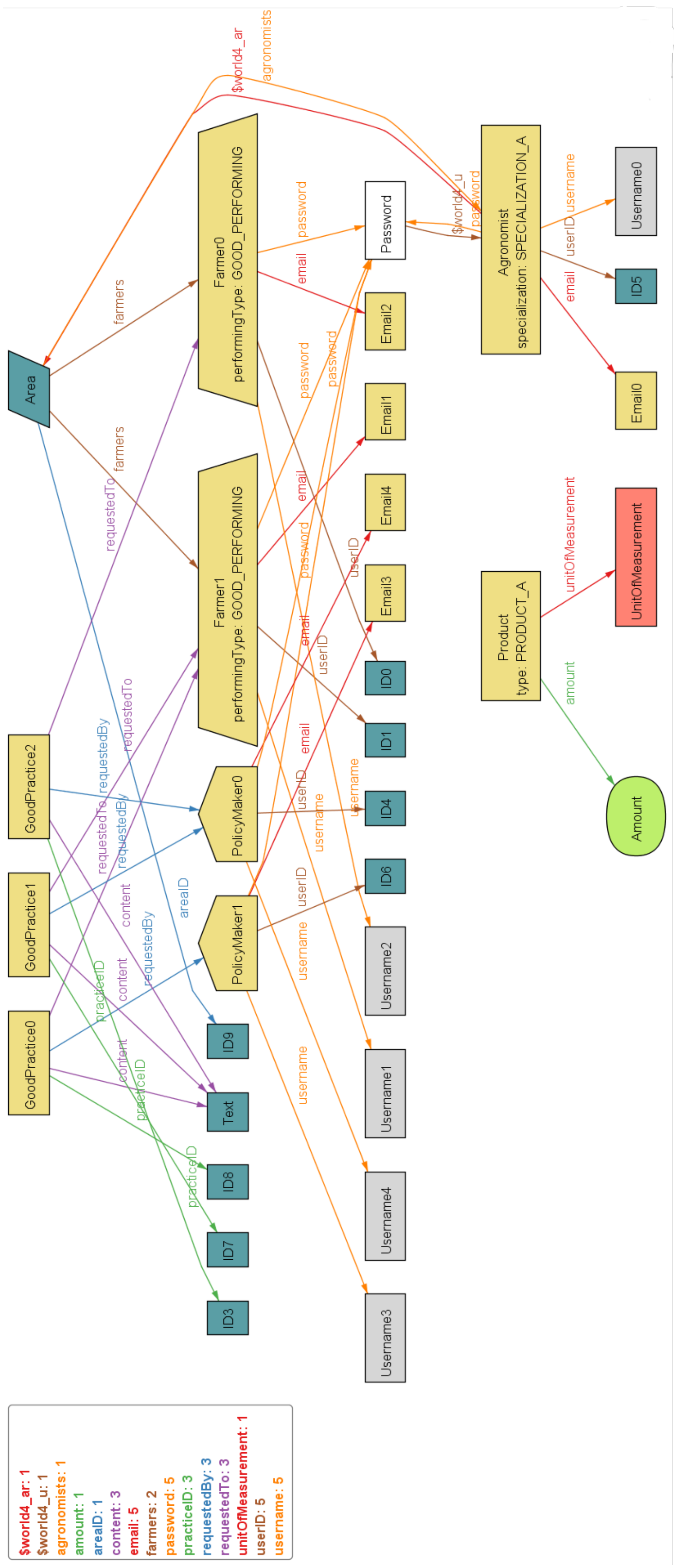


Figure 39: World focused on Good Practices

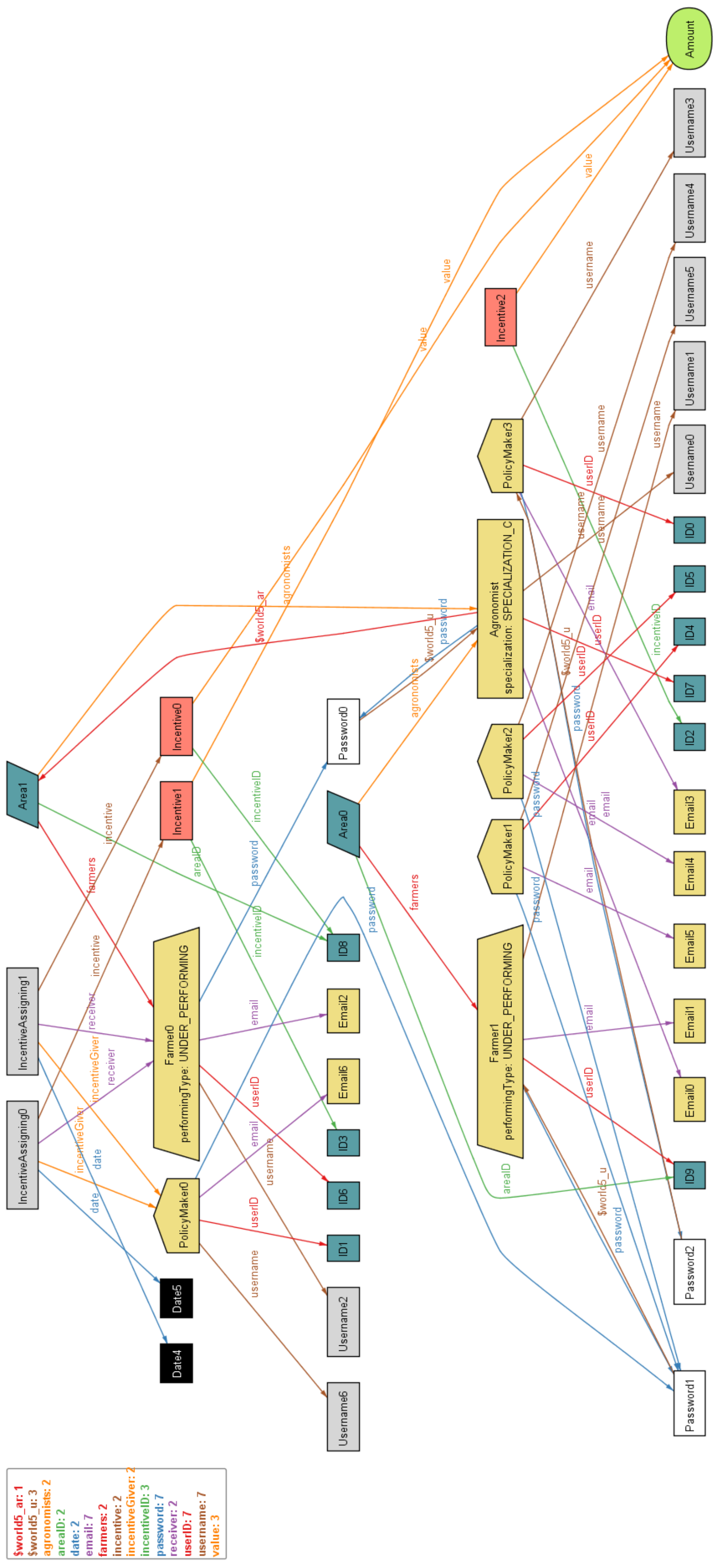


Figure 40: World focused on Incentives

5 Effort Spent

In this section we provide detailed information about how much effort each group member spent in working at this document. Further information about commits and updates is stored in the project GitHub repo. Furthermore also part of the Design Document effort is noted in the table below.

Section	Gori	Romanini	Watanabe
1.1	3,5		
1.2	4	2	2
1.3	2	2	4
1.4	6		
1.5	1		
1.6	3		
1.7	2		
2.1	11	8,5	9
2.2	3	2,5	5
2.3	1	2	
2.4	2	1	4
3.1			16
3.2	16	20,5	19,5
3.3		2	
3.4	3		
3.5		3	
4.1	4	15,5	
4.2		1	
4.3		2	
5.0	1	0,5	0,5
Whole document review	7	8	8
	69,5	70,5	68

Table 28: Table of efforts